Document Concordance Generator

Requirements Specification

Eric Brickner

Chris Blanchard

20-260-495
Software Engineering Lab
Professor Carter
Winter, 1992

1. Introduction

Requirements analysis is a software engineering ask that bridges the gap between system level software allocation and software design. Requirements analysis enables the system engineer to specify software and performance, indicate software's interface with other system elements, and establish design constraints that the software must meet. Requirements analysis allows the software engineer to refine the software allocation and build models of the processes, data, and behavioral domains that will be treated by the software. Requirements analysis provides the software designer with a representation of information and function that can be translated to data architectural, and procedural design. Finally the requirements specification provides the developer and the customer with the means to assess quality once the software is built.

-Roger S. Pressman

1.1 System Reference

The customer has not specified the context within which the Document Concordance Generator product executes. Thus, we define the system reference to be a standalone program executing on a non-realtime processor.

1.2 Overall Description

The Document Concordance Generator is intended as an aid in the study of large textual works.

Given the name of a file containing a document of an output file, the Document Concordance Generator produces a concordance of the document in the output file. The concordance output file contains a heading at the top of each page, a footer at the bottom of each page with page number, and a title page showing the name of the input file. Provision is made for the exclusion of certain words from the generated concordance.

1.3 Software Project Constraints

Software constraints include the following items:

1.) Finite quantity of secondary storage for input/output files.
2.) Finite quantity of RAM that may be incorporated into internal data structures.

2. Information Description

2.1 Information Flow Representation

Within the Document Concordance Generator, information flows in the following ways:

1.) The User Command Line flows from the Console to the User Command Parser, which then sends an Input Filename to the Line/Word Parser. Error Messages may also be sent to the Display. This module also sends the Output Filename to the Output Formatter.

2.) The Line/Word Parser accepts Input Text from the Input Text Document, and send Words to the Skip-Word Module and the Structure Builder, as well as Error Messages to the Display.

3.) The Skip-Word Module receives Words from the Line/Word Parser Module, and returns Status to that module.

4.) The Structure Builder sends Error Messages to the Display, and sends a Concordance Structure

to the Output Formatter Module.

5.) The Output Formatter Module sends Error Messages to the Display, and Output Text to the Output Text Document.

All symbols used in the data flow diagrams and control flow diagrams adhere to the standards presented in Figure 7.2 of the text.

2.1.1 Data Flow

For all Data Flow Diagrams of the DCG, please refer to figures 2.1 and 2.2.

Data Flow Diagram - Level 0

µ §

Figure 2.1

Data Flow Diagram - Level 1

µ §

Figure 2.2

2.2.2 Control Flow

For all Control Flow Diagrams, please refer to figure 2.2 and 2.3.

Control Flow Diagram - Level 0

µ §

Figure 2.3

Control Flow Diagram - Level 1

µ §

Figure 2.4

2.2 Information Content Representation

name: **Concordance Structure**
alias: CS

where used:
1.) The Structure Builder places Words and their Coordinates of Appearance into the CS.
2.) The Output Formatter retrieves Words and their Coordinates of Appearance from the CS.

how used:
Used to hold all Words and their Coordinates of Appearance.

content description:
A list of all non-skip Words within the Input Text Document and their associated Coordinates of Appearance

supplementary information:
none

name: **Coordinates of Appearance**
alias: none

where used:
1.) Passed from the Line/Word Parser Module to the Structure Builder.
2.) Passed from the Structure Builder to the Concordance Structure.
3.) Retrieved from the Concordance Structure by the Output Formatter.

how used:
Used to specify the location (Page,Line) of a Word within the Text Input Document..

content description:
1.) Page Number
2.) Line Number

supplementary information:
none

name: **Error Message**
alias: Error Msg

where used:
1.) Passed from the User Command Parser to the Display.
2.) Passed from the Line/Word Parser to the Display.
3.) Passed from the Structure Builder to the Display.
4.) Passed from the Output Formatter to the Display.

how used:
Used to communicate error conditions to the user.

content description:
A string of characters.

supplementary information:
none

name: **Input Filename**
alias: none

where used:
1.) Passed from the User Command Parser to the Line/Word Parser.

how used:
Used to specify which file to open for Input Text.

content description:
A string of characters.

supplementary information:
none

name: **Input Text**
alias: none

where used:
1.) Passed from the Input Text Document to the Line/Word Parser.

how used:
Used as input to the Line/Word Parser. The concordance is based on the contents of the Input Text.

content description:

A stream of ASCII characters.

supplementary information:
none

name: **Output Filename**
alias: none

where used:
1.) Passed from the User Command Parser to the Output Formatter.

how used:
Used to specify which file to open/create to receive the Output Text.

content description:
A string of characters.

supplementary information:
none

name: **Output Text**
alias: none

where used:
1.) Passed from the Output Formatter to the Output Text Document

how used:
Used to represent the contents of the Concordance Structure.

content description:
Lines of text, delimited by carriage returns, containing the Words and Coordinates of Appearance of the Words

supplementary information:
none

name: **Status**
alias: none

where used:
1.) Passed from the Skip-Word Module to the Line/Word Parser Module.

how used:
Used to specify if a given word is a member of the Skip-Word List.

content description:
Boolean

supplementary information:
none

name: **User Command Line**
alias: UCL

where used:
1.) Passed from the Console to the User Command Parser.

how used:
Used to invoke the Document Concordance Generator and to specify an Input Filename and Output Filename

content description:
DCG <Input Filename> <Output Filename>

supplementary information:
The elements within the UCL must be delimited by a blank space.

name: **Words**
alias: none

where used:
1.) Passed from the Line/Word Parser to the Skip-Word Module.
2.) Passed from the Line/Word Parser to the Structure Builder.
3.) Passed from the Structure Builder to the Concordance Structure.
4.) Retrieved from the Concordance Structure by the Output Formatter.

how used:
Used to specify the words of the concordance.

content description:
A string of characters delimited by white space.

supplementary information:
none

2.3 Standard Interface Description

The Document Concordance Generator operates in the command line mode from a SunOS command prompt. All error/status messaging is displayed to the command window. The input and resulting output files reside in the same directory as the Document Concordance Generator executable file.

3. Functional Description

Upon invoking the DCG, and specifying an input and an output file, the text document is read and parsed into its constituent words. Each word is checked against a list of skip-words. Any word appearing in the skip-word list will not appear in the final concordance. If the word is valid, it is added to the concordance structure, along with its coordinates of appearance. Once all words have been read and have all been placed into the concordance structure, the output document is then generated. The output document consists of a tabular presentation of each valid word along with its coordinates of appearance.

3.1 Functional Partitioning

The Document Concordance Generator has been partitioned into five functional modules.

User Command Parser (UCP) - This module accepts the User Command Line and parses it into an Input Filename and an Output Filename. It then functions to check the access rights of the specified files to ascertain whether they are correct, and notifies the user of any errors.

Line/Word Parser (LWP) - This module functions to parse the Input Text Document into its constituent Words. If any errors are encountered, they are reported to the user.

Skip-Word Module (SWM) - This module functions to determine if a given Word should be omitted from the concordance structure, and returns this information to the LWP.

Structure Builder (CS) - Given a Word and its Coordinates of Appearance from the LWP, this module functions to build the Concordance Structure.

Output Formatter (OF) - This module functions to generate the Output Text Document, given the Words and Coordinates of Appearance from the Concordance Structure.

3.2 Functional Description

3.2.1 Processing Narrative

User Command Parser - Upon invocation of the DCG, the UCP parses the User Command Line to determine if two filenames have been specified by the user. If not, an Error Message is issued. The validity of the Input Filename is then checked. If the file specified by the Input Filename does not exist or is write-only, an Error Message is issued. The validity of the Output Filename is then checked. If this file already exists, the user will be prompted to enter whether that file may be over-written. If the user responds affirmatively, the file is checked for its access rights. If the access is read-only, an Error Message is issued. Once all these steps have completed, the Input Filename is passed to the Line/Word Parser.

Line/Word Parser - Given the Input Filename, the LWP then opens the Input Text Document and parses the Input Text Document to determine its constituent words, as well as its Coordinates of Appearance. In turn, each Word is passed to the SWM for the purpose of having its skip-status checked. If the SWM returns a positive status for that Word, then that Word and its Coordinates of Appearance is passed to the SB. This process continues until each Word in the Input Text Document has been examined. If any errors occur during this process, an Error Message is displayed.

Skip-Word Module - As each Word is passed in, it is checked against the list of words that are to be excluded from the Concordance Structure. The Status of each Word is returned to the LWP.

Structure Builder - As each Word and its Coordinates of Appearance are passed in, the Concordance Structure is searched for the correct spot in which to insert the Word. The Coordinates of Appearance are then added to this structure. If any errors occur during this process, an Error Message is displayed. If successful, control is returned to the LWP.

Output Formatter - Once the SB has successfully completed its duties for each Word, control is turned over to the OF. The OF then works its way down the Concordance Structure, generating lines of Output Text to send to the Output Text Document. Each line of Output Text consists of the

Word and each of its Coordinates of Appearance. Once the entire Concordance Structure has been worked over, the Output Text Document is closed and the DCG terminates normally. If the OF is unable to complete its duties, an Error Message is displayed.

3.2.2 Restrictions and Limitations

User Command Parser - This module is limited to parsing filenames that are delimited by blank spaces. Any legal Input Text Document will be assumed to contain Input Text that is legal for processing by the LWP. That is, no check on the content of the file will be made by the UCP.

Line/Word Parser Module - This module is limited to parsing lines delimited by carriage returns, and to words delimited by blank spaces.

Skip-Word Module - No restrictions or limitations apply to the SWM.

Concordance Structure Builder Module - The maximum size of the concordance data structure is ultimately determined by the hardware configuration of the particular installation environment.

Output Format Module - No restrictions or limitations apply to the OFM.

3.2.3 Performance Requirements

There are no timing, delay, or other performance constraints on any of the modules.

3.2.4 Design Constraints

User Command Parser - There are no design constraints on the UCP.

Line/Word Parser - There are no design constraints on the LWP.

Skip-Word Module - As the number of skip-words increases, so does the need for an efficient searching algorithm.

Structure Builder - The Concordance Structure shall be designed to consume a minimum amount of the memory resource. In addition, as the number of Words in this structure increases, so does the need for an efficient searching algorithm.

Output Formatter - There are no design constraints on the OF.

3.2.5 Supporting Diagrams

Data Flow Diagrams (DFD)

Module Name	Data Item	In/Out	Source	Destination
User Command Parser	User Command Line	Input	Console	
	Input Filename	Output		LWP
	Error Message	Output		Display
Line/Word Parser	Input Text	Input	Input Text	

			Document	
	Status	Input	SWM	
	Input Filename	Input	UCP	
	Error Message	Output		Display
	Words	Output		SWM
	Words	Output		SWM
	Coordinates of Appearance	Output		SB
Skip-Word Module	Words	Input	LWP	
	Status	Output		LWP
Structure Builder	Words	Input	LWP	
	Coordinates of Appearance	Input	LWP	
	Error Message	Output		Display
	Words	Output		Concordance Structure
	Coordinates of Appearance	Output		Concordance Structure
Output Formatter	Words	Input	Output Text Document	
	Coordinates of Appearance	Input	Output Text Document	
	Error Message	Output		Display
	Output Text	Output		Output Text Document

Level 0 - In the level 0 DFD, the DCG is shown as the centerpiece of the system. Input in the form of a User Command Line enters the DCG from the console, while Input Text enters the DCG from the Input Text Document. Output consists of Error Messages that are sent to a display, and a Output Text sent to the Output Text Document.

Level 1 - In the level 1 DFD, the DCG has been subdivided into five parts, the User Command Parser, the Skip-Word Module, the Line/Word Parser, the Structure Builder, and the Output Formatter.

- The User Command Parser handles the User Command Line, and passes the Input Filename along to the Line/Word Parser. The UCP also may send an Error Message to the Display.

- The Line/Word Parser sends Words to the Skip-Word Module, and the Skip-Word returns the Status of that word. In addition, an Error Messages flow from the LWP to the Display. If all goes well, the LWP then sends the Word and its Coordinates of Appearance to the SB.

- The Structure Builder receives Words and Coordinates of Appearance from the LWP, and places them in the Concordance Structure. This module may send an Error Message to the Display.

- The Output Formatter accepts input in the form of Words and Coordinates of Appearance from the Concordance Structure, and sends output in the form of Status Messages and Output Text to the Display and the Output Text Document, respectively.

Control Flow Diagrams (CFD)

User Command Parser - This is the initial state. If the opening of the Input Text Document is a success, then control is handed to Line/Word Parser. If unsuccessful, an Error Message is issued and the DCG terminates.

Line/Word Parser - It is necessary to check each Word to see if it is to be excluded from the Concordance Structure. Therefore, as each Word is parsed, control is passed over to the Skip-Word Module. Next, control is passed to the Structure Builder. Once all the Words have been parsed, control is handed over to the Structure Builder. If any errors occur, an Error Message is issued, and the DCG terminates.

Skip-Word Module - After each word is checked against the skip-word list, control is handed back to the Line/Word Parser.

Structure Builder - As each Word and its Coordinates of Appearance have been added to the Concordance Structure, control is returned to the LWP. Once the Concordance Structure has been completely built, control is passed to the Output Formatter.

Output Format Module - When this module has finished sending Output Text to the Output Text Document, and no errors have occurred, the DCG terminates normally. Otherwise an Error Message is issued, and the DCG terminates.

3.3 Control Description

3.3.1 Control Specification

User Command Parser
Control Inputs: Start DCG
Control Outputs: UCP Done

Line/Word Parser

Control Inputs: UCP Done
Control Outputs: LWP Done, Check Skip-Word

Skip-Word Module
Control Inputs: Check Skip-Word
Control Outputs: SWM Done

Structure Builder
Control Inputs: LWP Done
Control Outputs: SB Done

Output Formatter
Control Inputs: SB Done
Control Outputs: terminate DCG.

3.3.2 Design Constraints

In as far as control description is concerned, no design constraints have been identified for any of the modules.

4. Behavioral Description

4.1 System States

User Command Parser - This is the initial state. If the opening of the Input Text Document is a success, then control is handed to Line/Word Parser. If unsuccessful, an Error Message is issued and the DCG terminates.

Line/Word Parser - It is necessary to check each Word to see if it is to be excluded from the Concordance Structure. Therefore, as each Word is parsed, control is passed over to the Skip-Word Module. Next, control is passed to the Structure Builder. Once all the Words have been parsed, control is handed over to the Structure Builder. If any errors occur, an Error Message is issued, and the DCG terminates.

Skip-Word Module - After each word is checked against the skip-word list, control is handed back to the Line/Word Parser.

Structure Builder - As each Word and its Coordinates of Appearance have been added to the Concordance Structure, control is returned to the LWP. Once the Concordance Structure has been completely built, control is passed to the Output Formatter.

Output Format Module - When this module has finished sending Output Text to the Output Text Document, and no errors have occurred, the DCG terminates normally. Otherwise an Error Message is issued, and the DCG terminates.

4.2 Events and Actions

Although the DCG is not event-driven, there are a few significant events which occur during normal operation.

1.) Program Initiation
Event: the DCG is invoked with and input and output filenames.
Action: the UCP determines if the files are valid for the DCG.

2.) Parser Go-Ahead
Event: the UCP passes the LWP the input text file stream.
Action: the LWP begins to parse the input text file.

3.) Check Skip-Word
Action: the LWP has found a word.
Event: the Skip-Word Module checks if it is a skip-word.

4.) Output Formatting Initiation
Event: the SB finishes building the concordance data structure.
Action: the OFM begins to generate the output text file.

5.) Program Termination
Event: the OFM finishes generating the output text file.
Action: the DCG terminates normally.

5. Validation Criteria

5.1 Performance Bounds

The performance expectations of the Document Concordance Generator are rather informal and intuitive in nature. The length of the runtime of the DCG shall be directly proportional to the length of its input text file. In addition, the accuracy of the DCG shall be 100%.

Breakdown by module:

User Command Parser - This module shall correctly determine the status of the input text file, i.e. whether it can be found, if it can be opened, and if it can be read. In addition, this module shall determine the status of the output file, i.e. whether it already exists, if it can be created, if it may be written to. Any error conditions shall be reported by this module.

Line/Word Parser Module - This module shall correctly parse the entire input text file into individual words. Any errors occurring during this process shall be reported to the user.

Skip-Word Module - This module shall correctly identify if a given word is or is not a member of its Skip-Word List. The status of a give word shall be returned to the Line/Word Parser Module.

Concordance Structure Builder Module - This module shall correctly determine the placement of the given word within the Concordance Data Structure. Any errors in the construction of this data structure shall be reported to the user.

Output Format Module - This module shall correctly format the output text file given a Concordance Data Structure. In addition, any errors that occur during the creation of the output text file shall be reported to the user.

5.2 Classes of Test

Tests shall be performed on each of the modules to determine it operates within the above specified performance bounds.

Test Class	Description of Test
User Command Parser Tests	Ensure that all two filenames are specified, and that all files may be correctly accessed.

Line/Word Parser	Ensure that this module can correctly distinguish among lines and words.
Skip-Word Module	Ensure that skip-words are handled appropriately.
Structure Builder	Ensure that the data structure is complete and correct.
Output Formatter	Ensure that the Output Text Document is complete and correct.

5.3 Expected Software Response

The above-specified classes of test shall illicit the following responses.

Test Class	Expected Software Response
User Command Parser Tests	An error message shall be issued if less than two filenames are specified, and if either file may not be correctly accessed.
Line/Word Parser	The LWP shall display the correct number of lines within the Input Text Document, as well as the correct number of lines within each line.
Skip-Word Module	The SWM shall correctly identify each member of the skip-word list.
Structure Builder	An error message shall be issued if this module is unable to complete the data structure.
Output Formatter	An error message shall be issued if this module is unable to complete the Output Text Document.

6. Bibliography

Carter, H. Software Engineering Laboratory - Projects and Guides. Unpublished. 1992.

Carter, H. Software Engineering - Lesson Plan and Student Workbook. Unpublished. 1992.

Pressman, Roger S. Software Engineering - A Practitioner's Approach. McGraw-Hill, New York. 1992.

7. Appendix