System Analysis: A Comparison of Two Methods

An Honors Thesis (ID 499)

by

Larry R. Bracken

Thesis Director

William F. Brown

Ball State University
Muncie, Indiana
May 1988

(Spring/88)
# Table of Contents

I. Introduction
   - Purpose ............................................................ 3
   - Method Used ............................................................ 3
   - Outline ............................................................ 4

II. Terminology
   - Software Engineering ........................................ 5
   - Project Life Cycle ........................................ 5
   - Structured Analysis ........................................ 6
   - Work product ........................................ 7
   - CS 497 ........................................ 7

III. CS 497 Analysis Phase ........................................ 9

IV. Fairley's Analysis Phase ...................................... 12

V. Examples of Fairley's Work Products ....................... 14

VI. Comparison of CS 497 to Fairley .......................... 124

VII. Recommendations .............................................. 130

   - Conclusion .............................................................. 134
SYSTEM ANALYSIS: A COMPARISON OF TWO METHODS

I. Introduction

Purpose

This paper will present the system analysis phase of two different software engineering methods—the method used in the CS 497 course (BSU86) and the method proposed by Richard E. Fairley (FAI85). The purpose of studying and using the analysis phase of these different software engineering methods is to gain an understanding of the difference between the types of software engineering methods that can be applied to the same life cycle. The analysis phase of the software engineering method discussed by Fairley will be compared to the CS 497 software engineering method to evaluate the differences between the two methodologies. The differences will be used to formulate possible improvements to the CS 497 Software Engineering I Systems Analysis course that would help make it more beneficial to computer science students.

Method Used

I am able to adequately compare the analysis phase of the two methods because of the practical experience I gained during the CS 497 Software Engineering I System Analysis course (Winter Quarter, 1986-87). This course required working in a team environment and applying the system analysis phase, as defined in the CS 497 standards manual, to a "live" project. To gain an understanding of the system analysis activities of the Fairley approach to software engineering, I applied the analysis
phase of the phased life cycle model to a small software project. This software project involved the planning of the development process and the definition of the requirements of a system that would assist my client in the production of a special type of literature for the deaf. I followed the activities required by Fairley and prepared the appropriate work products. These documents are the System Definition, the Project Plan, the Preliminary User Manual, and the System Requirements. After the completion of the analysis activities and the compilation of the work products, the work products obtained from Fairley's analysis phase were compared to the work product obtained from the CS 497 analysis phase. This comparison will be accomplished by presenting the activities of the analysis phase of each method separately and then discussing the major differences between them. The results of this comparison will be used to formulate recommendations to improve the CS 497 course.

Outline

This paper will be presented in the following format. To acquaint the reader with some of the major terms, I have defined the following words: software engineering, project life cycle, structured specifications, work products, and the CS 497 software engineering course. After the terminology is presented, the activities of the analysis phase of the two software engineering methods will be discussed separately. The major emphasis of the methods, the procedures that one must follow, and the resulting work products will be described, and the two methods will then be compared and contrasted. In conclusion, I will offer my recommendations concerning the CS 497 course.
II. Terminology

**Software Engineering**

Before presenting the methods of software engineering, a few terms should be defined for clarification. First, what exactly is software engineering? The term software engineering has been used to describe the process of developing software products with the use of formalized procedures and management controls. The use of these formalized procedures and controls leads to improved product quality and increased productivity by the software engineers. Software engineering differs greatly from the "traditional" form of computer programming because it utilizes techniques that are similar to ones used in other engineering disciplines. It also involves the use of management skills and techniques to control the project. According to Fairley (FAI85), "engineering problem solving techniques provide the basis for project planning, project management, systematic analysis, methodical design, careful fabrication, extensive validation, and ongoing maintenance activities." He also states that software engineering is a combination of the "concepts from computer science, management science, economics, and communication skills."

**Project life cycle**

One of the tools used by the software engineer is the project life cycle. This term refers to the stages that a software project passes through from its inception to its final completion. A project life cycle is
a general outline of the activities that will occur during a project. The life cycle provides an organization a basis for consistency in the development of software products. This consistency is insured by the introduction of checkpoints that are used for management control and evaluation. These checkpoints are placed throughout the life cycle—usually between the different phases that have been defined.

A typical life cycle consists of the activities to "define, develop, test, deliver, operate, and maintain a software product." (FAI85) The number and type of activities that are included in a life cycle will vary with the type and size of the project being developed. The smaller projects will not require the extensive management control that larger projects with more developers do. And though management control and checkpoints are critical to a project, a small project may only have one checkpoint (at the end of the project), but a large project would have several intermediate checkpoints to determine if the project is behind schedule and if more resources need to be allocated. A life cycle should be defined and stated for every project since it helps to organize the activities and makes sure that problems are addressed at the appropriate times.

Structured Analysis

Structured analysis is a technique used during the analysis phase of a software life cycle. Structured analysis introduces the use of graphic documentation tools to produce a structured specification document. The tools used by structured analysis are data flow diagrams, data dictionaries, decision trees, and decision tables. The data flow diagrams
are "a network of related functions showing (the) interfaces between (the) components." (DEM79) The data flow diagrams are a pictorial representation of the flow of data through the system. The diagrams are composed of bubbles that represent processes, rectangles which represent external sources of information or repositories of information, straight lines that represent files, and arrows that represent the flow of information. The data flow diagrams become a useful tool after which to model the system and also a way to partition the system. The data dictionary contains the definition of every data flow and data file that appears in the data flow diagrams. In this dictionary, the procedures to be followed in the processes of the system are described using Structured English, which is a special type of narrative text. Decision tables and decision trees are used in place of Structured English when they would describe the processes more effectively.

**Work Product**

Work products are the documents produced from the different phases of the life cycle. The documents produced from one phase becomes the input documents of the next phase. The number and type of work products produced depends on the type of life cycle being applied to the project and the type and size of the project.

**CS 497 Software Engineering course**

The CS 497 course is the first part of a two term course sequence involving the subject of software engineering. The analysis phase of the
life cycle is the subject addressed by the CS 497 course. This course is the first environment for the students to ever experience working in groups and applying a software engineering method to a "live" project. The main emphasis of this course is the development of a major software project that is decided upon by each team. Each team of usually three to five members obtains a sponsor that has a need for a software package and is willing to allow the team to work on rectifying the problem. The team receives approval of the project from their professor and then begin to plan the development process. CS 497 utilizes a set of guidelines and standards that must be followed by the team in completing the requirements and specification activities during the system analysis phase of the software life cycle. The activities that occur in the course are the drafting of a system project proposal, documentation of the existing system, and defining the system's functions and user requirements. Each member of the team is required to present a milestone report that involves describing the project and the progress that has been made thus far. The final work product is a Systems Proposal Report that contains a complete definition of the problem, a description of the system, an analysis of the system, a proposal of alternative solutions, and reasons why the recommended solution was selected. The teams are required to follow the standards in the manual, but have freedom when it comes to the actual details of the activities. The teams are required to use the following documents for the management control of the project: weekly team report, a meeting log, a project task schedule, and weekly progress reports.
The CS 497 course involves the application of a software life cycle that is based on DeMarco's structured analysis techniques. The analysis phase used in the CS 497 course places emphasis on the rigorous study and documentation of the current system environment. The client's current environment is studied to obtain a working understanding of the type of procedures that are performed by the personnel of the system. From the representation of the client's system, actual problems and system requirements are obtained, and the activities used to obtain a model of the current system are covered. The information received from the interviews of the client are used to model the current system. Other information that is pertinent are the roles and functions performed by the personnel.

The boundary or scope of the project should be defined toward the beginning of the analysis phase. The scope of the project shows the departments and personnel affected by the modification of the current system. The inputs and the outputs of the area that will be changed should be noted as well as the procedures that are performed. From all of the previous information collected, a detailed description of the current system is prepared by using the tools of structured analysis. All of the interviews of the client become the basis for developing a narrative description of the current system and other supporting documents. An organizational chart that presents the employees and their positions in a hierarchical form is prepared. A physical layout should be drafted to show the type of space that is used to perform the variety of tasks of the job.
There should be a description of all of the forms that are used and then the data flow diagrams that show the flow of information from one place to another are prepared. Data structure diagrams are used to describe any complex files of the system, such as databases. Along with describing the current environment, the limitations that are imposed by the existing system must be considered. The standards and policies of the business and the mission and goals of the organization should be studied as ways to begin compiling limitations. Other things that are considered are the procedures required to perform the job or function, the limitation of space, the cost of the current system, the error rate in performing the jobs, the time taken to perform the job, and the time delays that occur in the system. After all of the data has been compiled on the current system, an exact statement of the problem is then developed.

The next activity in the analysis phase of the CS 497 software life cycle is the analysis of the problem statement. The documentation and model of the current system are studied to find if there are any particular problem areas that need to be solved. The effect of the proposed modification must also be considered if such a change should take place. The problem statement is then expanded to encompass the goals that were previously stated. A data dictionary is created that contains the definition of all the components of the data flow diagrams.

An interview with the client will help answer questions concerning the new system, such as the type of functions that should be performed. A tentative data flow diagram of the new system is then prepared. After
drafting several alternative solutions with the appropriate data flow diagrams, a cost and benefit analysis is performed to find the most feasible solution. New structured specifications are produced once a strategy is selected for recommendation.

The last activity that is performed is the production of the Systems Proposal Report. The System Proposal Report is the work product from the analysis phase and will be used as the input in the next phase of the life cycle. The Systems Proposal Report will contain "a complete description of the problem, a description of the system, an analysis of the system showing how each part performs its task, a proposal of alternatives, and reasons why the recommended system was selected." (BSU86) A summary is included in the front of the report with conclusions concerning the system selected. The works cited are included plus any other supporting information, which is placed in the appendix.

Along with the required activities, certain management tools are implemented and required to keep control of the project. A weekly progress report is required including a team report, a meeting log, and a project task schedule. These tools help the top management (the professor and the team leader) to stay abreast of any major problems that might have occurred and the progress that is being made in the development of the project.
IV. Fairley’s Analysis Phase

The analysis phase of the life cycle defined by Richard E. Fairley encompasses a greater amount of preliminary planning than the CS 497 method. Fairley’s method of developing software requires more extensive planning at the beginning of the phased life cycle model. The activities of the method begin with the definition of the problem. First, a statement of the problem is defined by describing the present situation, documenting the constraints that exist, and documenting the goals to be achieved. The client is interviewed until adequate information is obtained to understand the current environment. The second step is to determine whether or not a computerized solution is appropriate. Sometimes the problem can be rectified not by introducing a totally new system, but by some small refinement of the current system. For a computerized system to be acceptable, the new system must use less resources and provide the same or more functions. The next step is to state the functions that will be provided, and the constraints on the hardware, software, and the people. The high level goals and acceptance criteria are then formalized.

The next activity is the development of a solution strategy. To obtain the best strategy, one must brainstorm without regard to any constraints of the system. A feasibility test is then performed on all of the strategies to find the best one. A list of product characteristics is then prioritized according to whether they are essential, less important, or “nice-if” features. The third step is the planning of the development process. This is where a life cycle is selected and its activities modified.
to fit the particular project. The life cycle that will be used on the project is defined, including the milestones that will occur and the final work products of each phase. An organizational structure of the management and personnel of the project is then constructed, including staff and resource schedules. A development schedule is developed and other considerations must be planned at this time. After the planning of the process is complete, the activity of defining the requirements is then commenced. The next step is to develop a model of the new system using the structured analysis techniques, develop the functional specifications, and the performance requirements.

There are a total of four manuals that are the work products of the Fairley method. The results of the planning activities are the System Definition, the Project Plan, and the Preliminary User's Manual. The System Requirement Specification is the product from the requirement definition activity. The checkpoints for this method are reviews that occur at strategic points throughout the life cycle. A product feasibility review is held after the problem definition and project planning activities. The System Definition and the Project Plan are reviewed and the project is evaluated to assess whether or not to continue, redirect, or terminate the project. The other checkpoint that occurs in the analysis phase is the software requirements review. This review is held for the purpose of making sure all the documentation from the analysis phase is adequate for the continuation of the project through its life cycle.
V. Examples of Fairley's work products

The results of applying the Fairley method to the analysis of a small project is shown in this section. The activities of Fairley's analysis phase were followed and the four required work products were produced. The manuals obtained from the development planning and the requirement definition activities are the System Definition, the Project Plan, the User's Manual, and the System Requirements manuals.
# SYSTEM DEFINITION

## Table of Contents

<table>
<thead>
<tr>
<th>SECTION</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Problem Definition</td>
<td>16</td>
</tr>
<tr>
<td>II</td>
<td>System Justification</td>
<td>24</td>
</tr>
<tr>
<td>III</td>
<td>Goals for the System and Project</td>
<td>26</td>
</tr>
<tr>
<td>IV</td>
<td>Constraints on the System and Project</td>
<td>27</td>
</tr>
<tr>
<td>V</td>
<td>Functions to be Provided</td>
<td>30</td>
</tr>
<tr>
<td>VI</td>
<td>User Characteristics</td>
<td>33</td>
</tr>
<tr>
<td>VII</td>
<td>Development/Operating/Maintenance Environments</td>
<td>34</td>
</tr>
<tr>
<td>VIII</td>
<td>Solution Strategy</td>
<td>36</td>
</tr>
<tr>
<td>IX</td>
<td>Priorities for the System Features</td>
<td>38</td>
</tr>
<tr>
<td>X</td>
<td>System Acceptance criteria</td>
<td>40</td>
</tr>
<tr>
<td>XI</td>
<td>Sources of Information</td>
<td>41</td>
</tr>
<tr>
<td>XII</td>
<td>Glossary of Terms</td>
<td>42</td>
</tr>
</tbody>
</table>
SECTION I: Problem Definition

1.0 Client Information

The client for this project is Kay Hoke of Goshen, Indiana, who is planning to start a business in the near future. Mrs. Hoke is a qualified instructor of the American Sign Language (ASL). Along with working with people who are deaf, Mrs. Hoke instructs people with normal hearing to become fluent enough with ASL that they will be able to converse and become leaders of classes or activities for the deaf. Mrs. Hoke has entrepreneurial tendencies and has several small ventures that are directly or indirectly related to her activities with the deaf community. At the moment, one of her activities is the production of figurines, necklaces, earrings, and other jewelry from molded resin. These products are produced in religious and secular shapes and in a myriad number of colors. Mrs. Hoke also has other products that she makes from her home and sells at places like bazaars.

Kay Hoke would now like to set up a business that will produce a special type of literature. This type of literature will be used as a tool for educators to help deaf people make a connection between the ASL hand signs and written ASL. There are similar products on the market at the moment, but they use a special sign language system that uses written English as its basis. These books have the story in written English with the hand sign equivalent placed above it. The sign is usually shown as a picture of a person actually performing the required hand configuration and movement of the hand sign. The pictures of the handsigns have been rendered by a professional illustrator. Mrs. Hoke would like to produce a comparable product that would involve a minimum of operating expenses. To aid in accomplishing this task, she would like a software package to be developed that will allow her to enter the manuscript text and have the computer translate it into the appropriate hand signs.

Kay Hoke has envisioned the type of hardware and software system that might be required to accomplish her business goal. Since Mrs. Hoke will be stationed out of her home, she does not want a computer system that will consume a large amount of her living space. Thus, she is
leaning toward a microcomputer that might utilize a hard disk system. She would compose the manuscript in the written form of ASL using a text editor. The text will then be stored for future editing, additions, or printing. This manuscript file will then be translated by the computer software, which will match the appropriate graphic picture of the hand sign to the proper ASL word. A second manuscript file will be created that is a combination of the ASL text and the corresponding ASL handsigns. This final document can then be displayed on the screen or printed on the printer. She will also need a way to create new handsigns if they have not been previously defined in the dictionary.

Kay Hoke has asked that an analysis of the software required to accomplish her goal be performed. She would like to be able to start the business in the next three (3) years depending on if the hardware and software needed is not exorbitantly expensive.

1.1 Product Background: American Sign Language

The American Sign Language (known as ASL or Ameslan) is a communication language used by deaf, mute, and hearing impaired individuals. Even though it is used by the majority of the deaf in the United States, ASL is only one example of the numerous manual and visual languages in existence. Not only are different sign language systems found outside of the United States, but there are many local variations in sign languages. Thomas Gallaudet, who had a profound influence on the evolution of education for the deaf, introduced ASL to the United States in 1816. The American Sign Language system is derived from combining the "French Sign Language with various U.S. local languages." [Friedman77]

The majority of the general populace holds the notion that sign language is a "one-to-one correspondence to English." [Rittenhouse83] This means that for every word of English there is a handsign. This notion is absolutely in error. The American Sign Language is a totally separate language with properties that are "different from English and other spoken languages." [Rittenhouse83] As spoken languages are not compatible with other spoken languages, ASL is not compatible with other sign languages.
The following is a list of other characteristics of the American Sign Language:

- its governing grammatical principles are different both in form and content
- there are a finite set of syntactic rules from which an infinite number of different sentences can be generated
- there are sign-formation rules analogous to pronunciation rules in spoken language
- ASL can report the past or the future. [Rittenhouse83]

There are many ways that ASL differs from the English language. A few of these differences are that ASL has no articles (i.e. *a, an, the*), no inflections, no signs for irregular words, no passive voice, and no prepositions other than individual modifiers for location (such as *on, over, etc.*). ASL also differs from English in its structure and its syntax. [Rittenhouse83]

ASL is a language that forms words by combining individual components (morphemes) together. In ASL, the meaning is not changed by the order that the signs are placed in a sentence, but by modifying the individual signs themselves. ASL signs are composed from a finite set of handsign components. There are four types of components, the hand-shape, the movement of the hand-shape, the place of articulation, and the orientation of the hands in relation to the body of the person. These components are called the ASL parameters of articulation because the modification of at least one of the four components will change the meaning of the handsign. All the handsigns are performed inside a special area called the articulation space. This area is limited to the front of the
body that the other person can see. The articulation area is a rectangle with an upper boundary located six (6) inches above the head and a lower boundary located at the waistline. The width of the articulation area is defined by the length of the person's arms bent at the elbows held at shoulder height.

First, the hand-shape component of the handsign will be addressed. There are fifty-four basic hand-shapes in the American Sign Language. These hand-shapes are categorized into three groups, with several individual hand-shapes appearing in more than one group. The first group is the American Manual Alphabet. There are twenty-six hand-shapes in this group that correspond one-to-one with the English alphabet. These signs are used for fingerspelling people's names, names of places, and other words that are not defined by an ASL handsign. The second group of handsigns are called classifiers. Classifiers are used to describe, identify, or quantify an object. The third group is a collection of forty other commonly used hand-shapes. These hand-shapes are then combined with the other components to produce the wide range of ASL handsigns.

Another component that can be used to change the meaning of the handsign is the type of movement. The movement component can be divided into four aspects:

- the number of hands that are articulators (i.e. that move); if both hands move. Do they perform the same type of movement or do they interact with each other?
- the body contact used for the execution of the handsign
- the direction of the movement
- the manner of the movement. [Friedman77]

If only one of the hands is the articulator, then the other hand must either be the place of articulation or do nothing at all. The only other choice is for both of the hands to be articulators. If the hands are both articulators and interact with each other, there are only five (5) ways that the hands may interact. These types are mutually exclusive and can only happen without any body contact (i.e. the signs are performed in the
neutral space in front of the body). One way that the hands can interact is that they can alternate movement in any of the three (3) axis (two horizontal and one vertical). The hands may alternately move from one side to another side, move toward and away from the person making the handsign, or move up and down. These are all straight movements. Other types of interactions are for the hands to approach each other on the horizontal axis, to separate and move away from each other, to cross over each other, and for the hands to link fingers or grasp each other. [Friedman77]

If the hands make contact with any part of the body, there are certain areas inside of the articulation space that are used. A list of the parts of the body that are usually used in making ASL signs can be found at the end of this section. There are a total of six (6) types of body contact that could occur when executing a handsign. There may be continuous contact as the hand that is the articulator moves from its starting point to the place-of-articulation. A second type of contact is called holding contact. This is when the articulator stays in contact while it is not moving toward the place-of-articulation. End-contact occurs when there is contact made at the end of the handsign, but not at the beginning of the handsign. Beginning-contact is just the opposite of end-contact. The fifth type of contact is called double-contact. This is when separate contact is made at two different places-of-articulation. The last type is when there is no contact during the execution of the handsign.

There are several different directions used when signing ASL. The possible directions fall on the three axis, vertical, horizontal-depth, horizontal-width. The types of direction that fall on the vertical axis are the motions of up, down, and up-and-down. The motions related to the horizontal-depth axis are the movement toward the signer, the movement away from the signer, and the movement to-and-fro. The last axis has the motions of right, left, and side-to-side. The last type of direction is for the hands to move in a non-directional way.

The manner of movement in an ASL handsign may be one of the following: it may be a straight motion, which involves the hand following a straight path from one point to another, it may be a twisting of the wrist, or it may be a circular motion of the hand or arm. The other manner of movement relates to the movement of the hand itself. The movements are
the bending of the fingers, the bending of the knuckles, the bending of the wrist, the opening of the hand, the closing of the hand, and the wiggling of the fingers.

The last component of the hand-shape is the orientation of the hand. The hand can be oriented toward the signer, away from the signer, up and down, or to the right or left of the signer.
Movement features [Friedman77]

INTERACTING MOVEMENTS:
- Alternate
- Separate
- Link
- Approach
- Cross
- Noninteracting

TYPES OF CONTACT:
- Continuous
- End
- Double
- Holding
- Beginning
- Noncontact

DIRECTIONS OF MOVEMENT:
Vertical
- Upward
- Up-and-down
- Downward

Horizontal-width
- Right
- Side--to-side
- Left

Horizontal-depth
- Toward
- To-and-from
- Away

Nondirectional

MANNER OF MOVEMENT:
- Straight
- Circular
- Bend--fingers
- Bend-wrist
- Close hand
- Twist
- Bend-knuckles
- Open hand
- Wiggle
PLACES OF ARTICULATION [Friedman77]

NEUTRAL SPACE

HEAD

- Whole face - whole face / Top of head
- Upper face - forehead / side-forehead / eyes / side-eyes
- Nose - center / under / side
- Lower face - chin / mouth / under-chin
- Cheek - center / side / ear
- Neck - center / side

TRUNK

- Chest - center / stomach / waist
- Shoulder

ARM

- Upperarm
- Elbow
- Dorsal forearm
- Ventral forearm
- Dorsal wrist
- Ventral wrist

Hand
SECTION II: System Justification

2.0 Product Justification

The type of literature that Kay Hoke proposes to produce is an important tool used to help the deaf people to connect the ASL handsigns to written ASL. There is similar literature already on the market at this time. The literature available are educational tools to help young, hearing impaired individuals develop important skills in reading and vocalizing English. The tool used to accomplish this development is called Signed English. Signed English is a type of sign language system that was developed for the sole purpose of providing the means for beginners to learn quickly and easily. Signed English is composed of easily learned signs that correspond one-to-one with the English language. The lexical system of Signed English is the same as written English in such a way that a direct translation is possible. At present, Signed English has not become popular because there are a limited number of people using it and it has a limited vocabulary of handsigns. If a person only learned Signed English, they would not be able to communicate with the vast majority of the deaf populace. Kay Hoke feels that there is a need for her proposed product to give educators another option in helping the development of their deaf pupils.

2.1 Project justification

Kay Hoke would like to keep the operating costs at a minimal level while still having a quality product produced through a straightforward procedure. The hardware and software system that will be proposed will achieve these goals. If Mrs. Hoke attempted to create the literature without the aid of a computer, she would be able to produce the text of the manuscript from any typewriter, but she would not be able to render the pictures of the handsigns herself. The services of a professional illustrator would have to be purchased to produce the drawings for the manuscript. The cost of production would be greater since the illustrator would have to be used for every piece of literature. There would also be a
lag time between when the manuscript text was complete and when the illustrator completed their work. The illustrator would also need to be provided with enough information regarding each handsign to be able to create a correct rendering of the ASL-sign. This scenario has potential for the occurrence of many delays due to lack of communication between Mrs. Hoke and the illustrator. If the illustrator is not given enough information about each picture, the result may not be an adequate representation of the proper handsign.

If the whole procedure was computerized, almost all of the problems that could occur without having it computerized would not have a probability of ever happening. The software system would allow Mrs. Hoke to compose directly on the computer using a text editor. There would be a spellchecker and other functions available to assist in the proofreading and editing of the draft copy of the manuscript. This will save time and effort on Mrs. Hoke's part. After she completes the manuscript draft to the way she wants it, the computer will then translate the ASL-words by matching them to the proper handsign pictures that are found in its dictionary. The final manuscript will then be displayed on the terminal screen for viewing and the last proofreading. The final manuscript can then be printed on the printer. Having the process computerized allows Mrs. Hoke to have the flexibility of obtaining the final manuscript anytime she requires it. The middle-man (the illustrator) is replaced by the computer system which makes the process completely self-contained. It would be very cost-effective to computerize this activity.
SECTION III : Goals for the system and project

3.0 Goals for the project

- The system should be deliverable in three (3) years.
- The project will not exceed the budget as set forth by the client after the design and implementation phase has been approved. There is no cost incurred by the client for the analysis phase of the project lifecycle but monetary consideration will be received for the design and implementation of the software.
- Documentation standards will be followed by producing clear, concise, and understandable manuals.
- The client will be consulted before each new phase of the software life cycle is started.
- Meet the performance requirements.
- A usable system will be produced.
- To train the end users to use the completed project.

3.1 Goals of the system

- The system will provide adequate storage for storing the draft and finished manuscripts, the component dictionary, and the ASL-word dictionary.
- All the ASL-components will be pre-defined and placed in the dictionary.
- The search of the dictionaries should not exceed sixty (60) seconds.
- The system will allow easy procedures for defining new signs.
- The system should produce clear and detailed graphic pictures.
- The printing of the manuscripts should take less than ten (10) minutes per page of text and graphics.
- The system will be able to be used to translate any written language to the appropriate handsign.
SECTION IV: Constraints on the System

4.0 Cost constraint

The total hardware required for the development and operation of the system should not exceed ten thousand dollars (plus or minus two thousand dollars).

4.1 Number of Individual Hardware Units

A. Minimum requirements:
The system will require the main processing unit of the computer, a keyboard and mouse for the input of the manuscript text and for selecting the options from the screen, two (2) floppy disk drives for the storage of the written text and the storage and retrieval of dictionary information, and a printer that can reproduce the graphic pictures of the ASL signs in the proper resolution quality.

B. Maximum requirements:
The maximum number of units that would be required would be the main processing unit of the computer, a keyboard and mouse for input, a 20 megabyte hard disk drive, one floppy disk drive, and a laser printer to produce a publishable form of the manuscript.

4.2 Developmental facilities required:

- computer system chosen by the client
- language editor, compiler, and debugger utilities
- reference manuals for the hardware of the system
- reference manuals for the programming language(s)
- reference materials for the ASL handsign components
- A printer for the output of high-resolution graphic pictures
- disk storage (floppy or hard disk)
4.3 Proprietary software to be used:

The software that will be more cost effective to purchase will be the language compilers and the text editor to be used. It may also be beneficial to purchase a graphic package to help create the handsign components to be placed in the component-dictionary. The graphic package can also be used to set up the appropriate screen formats needed. The only precaution that must be taken is to ensure that the picture that the package produces can then be manipulated through programming.

4.4 Hardware interfaces:

NONE

4.5 Software interfaces:

- Structure of the file produced by the text editor
- The required programming to use the pictures produced with the graphic package.

4.6 Data Interfaces:

- Data Interfaces: component-dictionary, ASL-sign-dictionary
- Data Formats:
  - The options available to the user must be selected from menus and/or windows.
  - The output must have the pictorial representation of the handsign placed above the text that it corresponds. For the output, use six blocks across and four blocks down.

- Validity Checks Required:
  - If an ASL-word has been previously defined
  - If a proper selection option has been chosen
4.7 Techniques/tools, algorithms, etc.

- Use bit-mapping to display the graphic representation of the handsign.

4.8 Environmental constraints:

- The electrical outlet needed for the system should not exceed 15 amperes, 125 voltage. This should be a grounded outlet with some type of overcurrent protection device.

- Temperature should be kept in the range of 65°F - 85°F.
SECTION V : Functions to be Provided

5.0 User functions:

A. The user will be required to be knowledgeable about ASL and its structure and syntax. The text that will be entered using the text editor will be in written ASL.

B. It will be the responsibility of the user to proofread the manuscript that has been entered to be sure that the proper ASL syntax has been followed.

C. If an ASL-word has not been previously defined, the user must create the definition of the new ASL-word by selecting the appropriate components from the ones available. As the components are selected, the user must then be sure that the pictorial representation of the handsign is accurate. This means checking to see if the components have been placed in the proper areas, the proper hand orientation has been selected, the direction is correct, and the proper type of motion has been selected.

D. The user must select the proper handsign if there are more than one handsign that is associated with a particular ASL word. For example, the word "if" has fifteen (15) different signs that define the word. There is a handsign for each of the different usages of the word. When this happens, all the handsigns in the dictionary that define the word will be displayed to the user. The user will be allowed to select one of the choices, or define a new handsign if it has not been previously placed in the dictionary.

5.1 Hardware functions:

A. The original ASL-text entered by the user will be stored on the secondary storage (floppy or hard disk).

B. The ASL-component-dictionary will be stored on floppy disks or a hard disk.

C. The ASL-code-dictionary will be stored on floppy disks or a hard disk.
disks.

D. The keyboard and mouse will allow easy input of text and option selections.

E. The printer will provide a relatively fast output of the final manuscript.

5.2 Software functions:

A. The text editor will be used to enter and edit the ASL text. The text editor will have the following capabilities:

- entering of text
- editing functions: cut/paste, delete/insert
- spellcheck utility
- store manuscript text to a file
- retrieve a manuscript text from a file

B. A dictionary will be provided which contains the components that can be used to define the different handsigns. The dictionary will also be used to match the proper handsign with the ASL-word being translated. The following functions will be provided for the maintenance of the dictionary:

- allow definition and addition of new ASL signs
- allow deletion of a definition
- allow validation of the definitions through listing capabilities.

C. If a particular ASL word has not been used before, the definition of the ASL-sign will not be found in the dictionary. The user must then create the definition and the software will provide the following functions to accomplish this task:

- all the selection options will be on menus
- allow the definition of a handsign by the retrieval of a previously defined handsign and modifying it
- allow the selection of particular components to define a handsign (manner of contact, type of movement, and orientation of hands)
- Allow the recording of the new handsign into the dictionary
- Allow the removal of the defined handsign before recording it.
D. The system will take the text file created by the text editor and translate the ASL-words to obtain the final manuscript. The translation process will involve the following functions:

- search the dictionary for the proper handsign of the ASL-word
- display all of the available ASL-signs that correspond to the particular ASL-word. If there are more than one handsign that corresponds to the word, the user will be required to select the appropriate one.
- prompt the user if the word is not found in the dictionary and needs to be defined
- save the ASL-word and its corresponding handsign to a manuscript file.

E. The software will allow the user to either display the final manuscript on the screen or print a hardcopy printout on the printer.
SECTION VI : User characteristics

6.0 Skills and abilities

• The user is not fully computer literate

• The user is not a programmer.

• The user is not an artist.
  The computer will provide an easy way for the user to define the required handsigns. All the components of the handsign, such as the pictures of the different hand-shapes and positions, will already be defined.

• The user is not familiar with the target computer.
  The user will not have to be familiar with the chosen target computer since the developer will provide the necessary training for the user to become acquainted with the hardware and software system.

• The user has typing skills that will be used when composing the manuscript on the computer with the text editor.

• The user is very knowledgeable about the ASL sign language, both the manual and written form.

• The user is able to write ASL in its proper syntax and is able to recognize and correct structure errors. The system will not be able to check for ASL syntax and structure errors in the manuscript typed in by the user.
SECTION VII: Development/operating/maintenance Environments

7.0 Development environment

The development of the system will be performed on the same type of computer configuration that will be used by the client. By using the same type of computer environment for development as the target computer, conversion and/or limiting the functions provided will not occur. This only occurs if a program is developed on a larger computer and then scaled down for one that is smaller. Because of the differences in the specifications of different microcomputers, it would be best to stay with ones that are compatible with the target computer. The developer will have access to the required language editors for writing the source code for the program, and the appropriate language compilers and linkers to execute the program. The environment will include all the manuals necessary for the completion of the project. The manuals that will be available will be the following:

- the language reference manuals
- the compiler reference manuals
- the reference manuals for the hardware
- the debugger reference manuals
- the manuals for any proprietary software that is being used

The system will be developed using a high-level computer language and any assembler modules needed for the execution of the graphic applications. Any graphic packages needed will be obtained, including the possibility of an electronic scanner. The creation of the different components of the handsign could be made more efficient by having access to either a graphics tablet or a scanner. This would allow the developer to quickly obtain the digitized image of the required handsign component. Otherwise, the developer will have to tediously create every one of the components from scratch.

The developer must also have access to the type of printer that the client will be actually using. This allows the developer to make sure that the proper control codes, and other machine dependent processes will
work.

7.1 Operating environment

The operating environment of the system will be the same microcomputer system that was used in the development process. During the operation of the software, the user will have access to the appropriate user manuals for all of the hardware of the system and the necessary user manuals for any proprietary software used as part of the system and the manuals for the system. The operating environment will consist of the microcomputer with two floppy disks or a hard disk and a floppy disk for its secondary storage, the keyboard and mouse, and the printer.

7.2 Maintenance environment

All maintenance will be performed on the same target-computer that was used during development and operation. The person maintaining the software will have access to all the documentation that pertains to the software of the system. The maintainer will also have access to all of the compilers, debuggers, and utility programs used during the development process.
Section VIII: Solution Strategy

8.0 Hardware proposed

- microcomputer (leaning toward the Macintosh Plus)
- hard disk drive or two(2) floppy disk drives.
- high quality dot matrix printer (Imagewriter)
- mouse for input

8.1 Method of text input and translation: Batch style

The user will enter the text with a text editor that is available for the target computer. Once all the text has been entered, proofread, and corrected, the text file will be translated. This method will be simpler than other methods to design and implement. Also, this method will allow a proprietary editor to be used. This will help minimize the cost of the project.

8.2 Translation of the text: coded sequence

The text will be matched with a dictionary that holds the words that have been previously defined by the user. The code structure for that word is obtained from the appropriate dictionary. This code contains the required information to describe the type of hand-shapes, type of movement, type of contact, etc. of the handsign. The code is then used to recreate the graphic picture that will be displayed on the screen or printed to the printer. The use of secondary storage resources is minimized by using the coded sequence for the storage of the graphic picture of the handsign. If the actual graphic picture was stored, the data for this bit-mapped picture would consume large amounts of storage area. This means that more floppy disks would be needed for storing the dictionary. Also, the search time during the translation phase would be greater.
8.3 Definition of ASL signs for the dictionary

A handsign will be defined by allowing the user to select the proper components from the component-dictionary. Each handsign will be coded by a string of bytes. Each byte will represent one type of component (hand-shape, hand movement, etc.) of the ASL-sign being defined. All the possible components must be developed by the designer and placed in the component-dictionary.

8.4 Development of the components

The components can be developed using graph paper with the following proportions:

- Hand : not more than 20 screen dots wide and 25 screen dots high
- Head/torso : not more than 100 x 100 screen dots
- Arrows (direction) : not more than 5 screen dots wide.

The bit representation of the graphed picture is then computed.

The following should be included:

- All the hand configurations used in ASL. This includes the view of the back and the palm side, and the different views of the hand when it is held in different directions and positions.

- Arrows denoting the different movement features

- Head/torso of a person used as the background and places of articulation

- Arms of the person which are attached to the torso depending on where the hands are located in the handsign.

There is a finite number of places in the articulation space that the components can be placed. This will allow the user to select the placement of the components from a menu. It will also make it possible to
set up a code string for the ASL-sign.

SECTION IX: Priorities for the system features

9.0 Essential Features:

• Option selection is made through menus and windows.

• The final manuscript can be displayed on the screen or printed on a printer.

• The definition of the handsigns by the user should be through the use of menus and the mouse.

• A word may have several different signs associated with it.

• Allow the user to select the proper handsign for the word if there is more than one handsign associated with it. If the proper usage is not shown, then the user will be allowed to define another handsign for that word.

• Allow the storage of the original text file and the manuscript file to be on a hard disk or floppy disk.

• A components dictionary that holds the graphic pictures of the units that are placed together to make a graphic picture of an ASL handsign.

• Displays all the signs that correspond to a particular ASL-word.

• Text is entered using a general text editor that allows the user to enter, proofread, and edit the text. The translation of the text does not commence until the user tells the computer to do so.

• If the target computer has a hard disk, the dictionaries will be stored on them.

• The final manuscript that contains the text and the pictures of the handsigns will be displayed on the screen before printing it. This allows the user to proofread the final document and to scrap it if there is any errors.

• A search in the ASL-word dictionary is made for each word of text to see if is defined.
9.1 Less important features:

* A mouse is used for the majority of the option selection.
* The system will be easy to use and be "idiot proof".
* The system will allow the text of the final manuscript to be in different text fonts and sizes.
* Allow the editing and modification of previously defined handsigns.

9.2 "Nice If" features:

- Allow the user to type in English text which will then be translated into written ASL and then the appropriate ASL handsigns.

- Allow the user to change the page format of the final manuscript. This means changing the number of ASL-word and ASL-sign combinations that are printed on a line and the number of lines to be allowed per page.

- Be able to print the final manuscript in either manuscript form or in book form.

- Allow the listing of the contents of the dictionaries and the cross-references.

- Have a tutorial disk as a system orientation tool for new users.
SECTION X : System acceptance criteria

10.0 Hardware acceptance criteria:

- meets the storage needs of the system and also meets the access speed for the dictionary searches.

- if the system hardware is usable for the purpose that it was intended.

- The printer must meet the resolution requirements for printing the graphics.

- The printer must meet the speed requirement for printing the graphics.

10.1 Software acceptance criteria:

- The software is reliable and efficiently uses the resources (memory).

- The software must be error free and be "idiot proof".

- The software must be easy to load, execute, and use.

- There must be adequate documentation of the software.

- The software must be modifiable without expensive effort.

- The software must meet all the performance requirements.

- The software must perform the required functions and tasks.

10.2 Documentation criteria:

- The user will receive a detailed user manual and other documentation required to execute the program and modify it later.

- There will be cross-referencing of the documentation.

- The documentation shall be concise, accurate, and complete.
SECTION 11 : Sources of Information


SECTION XII : Glossary of terms

American Manual Alphabet: A set of twenty-six handsigns that represent the letters of the English alphabet.

American Sign Language: A sign language system used in the USA

Articulation space: The area in front of the signer that the person receiving the handsign can easily see.

ASL: American Sign Language.

ASL-component: The units that are combined together to produce an ASL-sign.

ASL-sign: The manual handsign that is used in the American Sign Language system as the basis for communication.

ASL-word: The smallest unit of written ASL.

Bit mapping: The definition of a graphic picture through the manipulation of the bits that correspond to the screen.

Classifiers: A set of handsigns that are used to describe, identify, or quantify an object.

Code-dictionary: A file that holds the coded sequences that describe the handsigns that have been defined.

Component-dictionary: A file that holds the pictures of the ASL-components.

Debugger: A developmental tool that helps the programmer find and correct the errors in the programs being developed.

Fingerspelling: The act of using the American Manual Alphabet to spell words that are not defined by any of the ASL signs.

hand-shape: The shape of the hand required for performing the ASL-sign.

Hard disk: Secondary storage (disk drive) that has the disk permanently encased.

Language compilers: Software that converts the source code of a high-level computer language to the language understood by the computer.

Linkers: Software that sets up the required addresses in memory to allow the compiled program to be executed by the computer.

Manuscript file: A file that stores the original text entered by the user.
Microcomputer: Any of the small computers.

Morpheme: The minimal grammatical units of a language (such as a word) that can not be divided into separate grammatical parts.

Neutral space: The space directly in front and away from the body of the signer.

Place-of-articulation: The place where contact is made during the execution of an ASL handsign.

Proprietary Software: Software that is purchased from a vendor.

Signed English: A sign language system that uses simple hand signs that correspond to written English.

Spellchecker: A utility to find spelling errors in text documents.

Software lifecycle: All the activities required in a software development process.

Source code: The high-level language command statements that must then be compiled to obtain the language that the computer understands.

Target computer: The computer system of the end user.

Text editor: A tool that allows a document to be entered in the computer and then easily modified.
# PROJECT PLAN

Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Life cycle model</td>
<td>44</td>
</tr>
<tr>
<td>II</td>
<td>Organizational structure</td>
<td>47</td>
</tr>
<tr>
<td>III</td>
<td>Preliminary staffing requirements</td>
<td>50</td>
</tr>
<tr>
<td>IV</td>
<td>Preliminary development schedule</td>
<td>54</td>
</tr>
<tr>
<td>V</td>
<td>Preliminary cost estimate</td>
<td>56</td>
</tr>
<tr>
<td>VI</td>
<td>Project monitoring and control mechanisms</td>
<td>57</td>
</tr>
<tr>
<td>VII</td>
<td>Tools and techniques to be used</td>
<td>58</td>
</tr>
<tr>
<td>VIII</td>
<td>Programming languages</td>
<td>59</td>
</tr>
<tr>
<td>IX</td>
<td>Testing requirements</td>
<td>60</td>
</tr>
<tr>
<td>X</td>
<td>Supporting documents required</td>
<td>61</td>
</tr>
<tr>
<td>XI</td>
<td>Manner of demonstration and delivery</td>
<td>62</td>
</tr>
<tr>
<td>XII</td>
<td>Training schedule and materials</td>
<td>63</td>
</tr>
<tr>
<td>XIII</td>
<td>Installation Plan</td>
<td>64</td>
</tr>
<tr>
<td>XIV</td>
<td>Maintenance considerations</td>
<td>65</td>
</tr>
<tr>
<td>XV</td>
<td>Method and time of delivery</td>
<td>66</td>
</tr>
<tr>
<td>XVI</td>
<td>Method and time of payment</td>
<td>67</td>
</tr>
<tr>
<td>XVII</td>
<td>Sources of information</td>
<td>68</td>
</tr>
</tbody>
</table>
SECTION I: Life cycle model

1.0 Project life cycle used

This project will be developed following the phased life cycle model as defined by Richard Fairley [Fairley85]. The phased model consists of the following phases:

• Analysis phase
• Design phase
• Implementation phase
• Testing phase
• Operation and Maintenance phase

The phases are performed in succession with one phase being completed before beginning the next phase of the life cycle. During each phase, the work products from previous phases are reviewed.

1.1 Analysis phase

The first phase of the model is the analysis phase. This phase consists of the two activities of planning the development process and defining the requirements.

The planning activity involves the definition of the customer's problem, development of a solution strategy after performing feasibility studies on the different alternatives available, determining the acceptance criteria, and planning the development process. The work products produced from activity are the System Definition and the Project Plan.

The activity of defining the requirements is concerned with finding the functions that will be performed by the
components of the system and the constraints under which they must perform their functions. The products from this activity will be the Preliminary User Manual and the System Requirements.

A review of the work products from the analysis phase will occur at the end of this phase. This will be a feasibility analysis to see whether the client would like to invest resources into developing the software package.

1.2 Design phase

The design phase is concerned with recognizing the components of the software being proposed, "specifying relationships among components, specifying software structure, ..and providing a blueprint for the implementation phase" of the life cycle. [Fairley85] The design phase involves two activities, architectural design and detailed analysis. Architectural design is the process of identification, decoupling and decomposing the software components into processing modules. This is achieved through the technique of structured analysis. Detailed design involves looking at how to package the processing modules, implement the processing algorithms, data structures, and the interconnections among the modules and the data structures.

The work products produced from the design phase are the Architectural design specification and the Detailed design specification. The work products are then reviewed after this phase is completed.

1.3 Implementation phase

The implementation phase of the life cycle involves taking the design specifications and using them to produce source code. The tasks of debugging the source code, writing the necessary documentation, and the unit testing of each of the modules programmed are also part of the activities of the
implementation phase. During the implementation phase, the finished modules will be reviewed and as the system evolves, the system will be shown to the client for approval. The work products prepared during this phase of the life cycle include the final User's Manual, the Installation and Training Plans, and the software maintenance plan.

1.4 System testing phase

There are two types of testing involved in this phase, life cycle verification and formal verification. First, the work products are reviewed to find out the degree in which they fulfill the requirements as stated in the specifications. Formal verification involves the rigorous testing of the source code through mathematical means to ensure that it meets the requirements.

1.5 Operations / maintenance phase

There are two activities associated with this phase of the life cycle. The first activity is one of correcting the errors that occur during the first months of usage by the client. The software will require modification and then be tested again. The second type of activity of the operations and maintenance phase is one of enhancement of the system. This may include adding new and improved functions to the system, improving the user menus and display formats, or increasing the performance of the system.
SECTION II: Organizational structure

2.0 Management structure

The project format of management will be followed for this project. Due to the small size of this project and the limited resources available to develop and implement the system, the project will be "conducted ... from start to finish" by the same team that is selected for the project. This type of management will be used for the project definition, the design of the system, the implementation and the maintenance.

2.1 Team structure

Because of the limited resources at the client's disposal, there will only be two individuals that will be working on the project. The individuals will not be working during the same phases of the life cycle. The following parts will delineate the functions that each of the individuals will be responsible.

2.2 Work Breakdown Structure
2.3 Statements of work (WBS)

1000 Analysis phase
   1100 Analysis management
   1200 Produce System Definition
   1300 Produce Project Plan
   1400 Produce Preliminary User's Manual
   1500 Produce System Specification
   1600 Perform System modelling

2000 Design phase
   2100 Design management
   2200 Produce Architectural Design Specification
   2300 Produce Detailed Design Specification
   2400 Produce Acceptance Test Plan
   2500 Refine Project Plan
   2600 Refine SYstem Models
3000 Implementation phase
   3100 Implementation management
   3200 Produce System for trial
   3300 Prepare User documentation
   3400 Prepare Installation and Training plans
   3500 Prepare Software Maintenance Plan
   3600 Refine Project Plan

4000 System testing
   4100 System testing management
   4200 Perform system tests and acceptance
   4300 Prepare Software Verification Summary
   4400 Install system

5000 Maintenance phase
   5100 Maintenance management
   5200 Fix errors in the system
   5300 System enhancement
   5400 Adapt the system to environmental changes

6000 Project management
   6100 General management
   6200 Liaison for client and suppliers
   6300 Provide training
SECTION III: Preliminary staffing and resource requirements

3.0 Preliminary staffing requirements

The staffing requirement was estimated using the COCOMO model as described by Fairley [Fairley85] and Londeix [Londeix87]. The project has tight constraints and the problem being solved has not been dealt with previously. To estimate the staffing, the embedded mode of the COCOMO model will be used. We will estimate on the conservative side and say that the project will have up to five thousand source instructions that the developer will produce. Thus the cost equation is given by

\[
PM = 2.4 \times (KDSI)^{1.20} \\
= 16.556756
\]

and the development time by:

\[
TDEV = 2.5 \times (PM)^{0.32} \\
= 6.1377909
\]

The cost equation (PM) is then multiplied an adjustment is the product from the multiplication of the values of the fifteen cost drivers.
The following multiplier values were used:

<table>
<thead>
<tr>
<th>COST DRIVER</th>
<th>RATING</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required Software Reliability</td>
<td>Very Low</td>
<td>0.75</td>
</tr>
<tr>
<td>Data base size</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>Product complexity</td>
<td>High</td>
<td>1.15</td>
</tr>
<tr>
<td>Execution time constraint</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>Main storage constraint</td>
<td>High</td>
<td>1.06</td>
</tr>
<tr>
<td>Virtual machine volatility</td>
<td>Low</td>
<td>0.87</td>
</tr>
<tr>
<td>Computer turnaround time</td>
<td>Low</td>
<td>0.87</td>
</tr>
<tr>
<td>Analyst capability</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>Application experience</td>
<td>High</td>
<td>0.91</td>
</tr>
<tr>
<td>Programmer capability</td>
<td>High</td>
<td>0.86</td>
</tr>
<tr>
<td>Virtual machine experience</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>Programming language experience</td>
<td>Low</td>
<td>1.07</td>
</tr>
<tr>
<td>Modern programming practices</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>Use of software tools</td>
<td>Low</td>
<td>1.10</td>
</tr>
<tr>
<td>Schedule constraint</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
</tbody>
</table>

The above cost driver values multiplied together produces the adjustment of 0.63741. Thus the cost equation value used is the following:

\[
PM = (16.556756) \times 0.63741
\]

\[
= 10.553441
\]

The average number of personnel needed is obtained by dividing the number of programmer months (PM) by the time for development (TDEV).

\[
PM/TDEV = 10.553441 / 6.1377909
\]

\[
= 1.7194202
\]

A preliminary staffing schedule was developed from this data.
This schedule can be found on the next page.
PRELIMINARY STAFFING SCHEDULE
(based on 22 months of development 2*PM)

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>SCHEDULE</th>
<th>PERSONNEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plans and requirements</td>
<td>1.32 MO</td>
<td>1</td>
</tr>
<tr>
<td>Architectural Design</td>
<td>3.52 MO</td>
<td>1</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>5.28 MO</td>
<td>2</td>
</tr>
<tr>
<td>Implementation</td>
<td>8.36 MO</td>
<td>1</td>
</tr>
<tr>
<td>System test</td>
<td>4.84 MO</td>
<td>1</td>
</tr>
</tbody>
</table>
# System Development: Gantt Chart

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architectural</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding Top Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coding Bottom Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operations/Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error Correction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhancements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Key

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Estimate|   |   |   |   |   |   |   |   |   |   |   |   |
| Actual|   |   |   |   |   |   |   |   |   |   |   |   |   |
SECTION V : Preliminary cost estimate

5.0 Cost of analysis

The cost of the activities of planning the development of the project and the definition of the requirements are not considered since the client is not being charged for these services.

5.1 Personnel cost

The cost for the personnel developing the system can be estimated, but the amount of compensation for the system will be negotiated when the product is delivered.

5.2 Hardware costs

A microcomputer system will fall in the range of the following amounts: $2,500.00 - 5,600, depending on the type of hardware is purchased. It is almost impossible to estimate the amount that will be spent on the hardware due to the uncertainty of the type of target computer that will be chosen by the client.
SECTION VI: Project monitoring and control mechanisms

6.0 Project monitoring and control mechanisms

- The system will periodically be shown to the client for feedback on the functions, displays, and procedures being implemented.

- A Preliminary Design Review and a Critical Design Review will be used as milestones.

- The modules of the system will periodically be submitted to a walkthrough with the personnel responsible for the analysis phase.

- Integration testing will occur after a new module has been programmed.

- The documentation shall be kept current as the system is being implemented.

- A list of the errors that occur will be kept.

- A list of the problems encountered in the design and implementation phases will be documented.

- The work products of the project life cycle will be continuously reviewed and updated to the new information.
SECTION VII: Tools and techniques to be used

7.0 Requirement definition

- Structured analysis techniques
  - Data flow diagrams
  - Data dictionaries
  - Structured English
  - Relational specification

- Word processor for writing of manual
- Graphic program to develop mock screen formats

7.1 Design phase

- Top-down Structured design techniques
- Modularization
- Structured flowcharts
- Stepwise refinement
- Word processor for writing of documentation

7.2 Implementation phase

- Structured coding techniques
- Program unit notebooks
- Compiler, linkage editor, debugger
- Walkthroughs and inspections
SECTION VIII: Programming languages

8.0 Primary Language characteristics

- Compiled language
- User-defined data types
- Manipulation of graphic data
- Good file access capability

8.1 Possible languages:

- BASIC
- PASCAL
- ASSEMBLER
- C

The selection of the language will depend on which languages are available for the target computer chosen, and which languages are a compiled language instead of an interpreted programming language.
SECTION IX : Testing requirements

9.0 Functional tests

- Testing of all error handling routines
- Testing of all input drivers (Menu management)
- Testing of mouse selection of functions
- Testing all functions of the system
- Testing of file access routines
- Testing of all display formats
- Testing of all output formats and interfaces

9.1 Performance tests

- Testing of the speed of searching a dictionary
- Testing of file access speed
- Testing of ease of user definition of a handsign
- Testing of memory usage
- Testing of translation time
- Testing of printing final manuscript

9.2 Structure tests

- Follow all the paths of the system
- Verify that requirements have been followed
SECTION X : Supporting documents required

10.0 Design phase
• System Requirements document
• American Sign Language reference manual
• Structured analysis reference document
• Target computer reference manuals

10.1 Implementation phase
• Requirements specifications
• Design documents
• Test plans
• User's manuals
• Installation instructions
• Computer language reference manuals
• Text editor reference manual
• Target computer reference manuals
• Detailed design documentation
• Printer documentation

10.2 Testing phase
• Test Plan

10.3 Maintenance phase
• All system documentation
• All hardware documentation
• All previous design documentation
• Maintenance guide
• Test suite description
SECTION XI: Manner of demonstration and delivery

11.0 Demonstration

The demonstration of the system will be performed at either the client's home or the developer's home, depending on the agreed arrangement. The client will be allowed to go through the system with the help of the developer. The developer should have a demonstration plan before performing the actual demonstration. Notes should be taken of the criticisms of the system and ways that the client would like it to be improved. If the proposed changes fall under the "nice if" features, the developer may either decline the upgrade or negotiate a new contract.

11.1 Delivery

The system shall be personally delivered to the client by the developer to the client's home in Goshen, IN. The package to be delivered shall be composed of the following:

- the program of the system
- the system dictionaries
- the User Manual
- all other documentation.
12.0 Training Schedule

The schedule for training the user cannot be made at this time. The times for training, after the system has been accepted, will be left to the discretion of the client and the developer. This is being done because of the difficulty in setting of specific dates that would be convenient for both parties.
SECTION XIII: Installation plan

13.0 Installation plan

The installation of the system will occur after the system has been tested for major errors. It will be acceptable to install a prototype of the system that has the majority of the essential functions implemented. As the program is fixed or modified to include more features, the client can be given an upgrade of the system. No specific detail can be given for the installation of the system until the actual configuration of the final system is determined.
SECTION XIV : Maintenance considerations

14.0 Analysis

- Develop standards and guidelines
- Set milestones for the work products
- Prioritize and list likely enhancements of the system
- Determine resources needed for future maintenance

14.1 Design

- Design to allow future enhancements
- Use standard notation for describing data flows
- Specify likely exceptions and ways to handle them
- Provide cross reference directories

14.2 Implementation

- Use structured programming techniques
- Use a simple and clear coding style
- Use standard internal documentation

14.3 Other

- Develop a maintenance guide
- Develop a test suite
SECTION XV : Method and time of delivery

15.0 Method of delivery

The system and its dictionaries shall be stored on diskettes with duplicate copies. The diskettes will be delivered to the client's home in Goshen, IN by the developer. The system will be installed on a hard disk if the target computer has one. All the required documentation shall be given to the client.

15.1 Time of delivery

The system will be delivered after all the appropriate testing has been finished. It is not known at the present time when the design phase will commence.
SECTION XVI: Method and time of payment

16.0 Method of payment

A lump sum payment to the developer of the software will be made based on the agreed amount of consideration to be exchanged. The payment shall only be made when the product has been accepted by the client. Acceptance shall be based on the criteria stated in section 10 of the System Definition.

16.1 Time of payment

The payment of the agreed amount of consideration shall be made at a date after the software system has been delivered and accepted. The exact date shall be negotiated by the developer and the client.
SECTION XVII: Sources of information


<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Introduction</td>
<td>68</td>
</tr>
<tr>
<td>II</td>
<td>Getting started</td>
<td>72</td>
</tr>
<tr>
<td>III</td>
<td>Modes of operations</td>
<td>79</td>
</tr>
<tr>
<td>IV</td>
<td>Advanced features</td>
<td>92</td>
</tr>
<tr>
<td>V</td>
<td>Command syntax</td>
<td>93</td>
</tr>
</tbody>
</table>
SECTION I : Introduction

1.0 Product overview and rationale

- Translation of Macwrite text files of written ASL to graphic pictures of the handsigns corresponding to the text.
- Menu selection of options by the use of a mouse
- Dictionary of ASL-words and the corresponding handsigns
- User-definition of dictionary entries through a selection process
- Multiple handsigns corresponding to one ASL-word
- Printing of final translated document
- Dictionary management instructions
- Text can be in different fonts or sizes of letters

This product is for a person that is knowledgeable about written ASL and wants to produce the type of literature that this product creates. This product takes the text file that has been created by the MacWrite word processor and translates the text into the corresponding handsigns. A new text file is then created that has the handsign picture placed above the corresponding word. This new document is then outputted to a printer that can reproduce the graphics that the Macintosh produces.
1.2 Terminology and basic features

**TERMS**

Click: Moving the cursor arrow to the selected option and pressing the button on the mouse to register the selection.

Menu: A rectangular box that lists the options available to the user.

Mouse: the small rectangular box with a button that is used to control the cursor and to select options from the screen.

**Basic Features**

- File commands - save, load, delete
- Translation of file
- Dictionary management - search, define, delete
- Change the printer
- Print the document to the printer
1.3 Summary of display and report formats

The user displays follow the specifications used by the other Macintosh software. This means that all option selection are through menus and the manipulation of the mouse. Every screen has an instruction block at the top with the functions labeled on it. When you place the pointer on the desired function, a menu will pull down to show the available options of that particular function. The area below the instruction block will be framed by a colored rectangle. Here is an example of a typical screen.

The document obtained from the translation process will contain the
words of the written ASL that was in the MacWrite file designated to be translated. The computer will place the picture of a person performing the appropriate handsign above the corresponding word that is in the document. Each page of the translated document has a capacity of sixteen words of text and their handsigns.

1.4 Outline of this manual

The section Getting Started will be presented first. This section will cover the procedures that you should follow to begin the execution of the program. The type of assistance that is available to you when questions arise will be covered next. The section will end with a small example of how the translation function works. The procedures to follow will be explained and the output will be shown. This example will give you a start in using this system. Examples of using all the functions will NOT be shown in this sample run. The tutorial program should be used for examples of the usage of the other commands. Section three will cover all the types of commands that are available on this system. The name of the command will be in large outlined text with the display screen shown below it. The command is then explained following the example of the display. This section also explains the type of reports that the system will generate. Section four will list the advanced features of the system. The manual will conclude with section five that covers the syntax of the commands and the different system options.
SECTION II: Getting started

2.0 Sign-on

The following procedures should be followed to load and execute this system.

1. Turn on the Macintosh computer
2. Place the Macintosh system disk in the main disk drive to initialize the Macintosh.
3. Place the program disk in the external drive.
4. Select the program HANDSIGN with the pointer and click the button located on the mouse twice.
5. Wait a few moments and then the main menu will be displayed on the screen.

2.1 Help mode

A tutorial disk will be available for you to gain an understanding of the system by following some examples. The tutorial will guide you through the program start-up process and provide you with examples of the usage of all the commands and options that are available on the system. You will be given instruction on how to enter the text using the Macwrite program and to save it for future use. The steps of translating the previously entered program will be covered, from loading the text file, translating the file, and storing the translated file, and printing the translated document. Also, the steps involved in
the maintenance of the dictionary and creating the new handsign entries to the dictionary are covered. Along with this tutorial disk, the developer will be offering training sessions on this system.
2.2 Sample run

To help you to see how the translation of the text is accomplished with this system, a sample execution is being presented. A translation of only one word will be shown, since the procedures are the same for any size document.

First, we assume that a file has been created using the MacWrite program that contains the single word TALL. This file is stored under the name of File 2. Shutdown the computer and remove all the disks. We will now go through the procedure from initializing the computer to obtaining the final printout. The first thing to do is to start the execution of the program by following the procedures that are listed under the sign-on section. You will be shown the main menu screen which looks like the following:

```
MAIN | File  Translate Dictionary Font Options
```

![Diagram of main menu screen](image-url)
The next step is to retrieve the file that we had created with MacWrite that was explained in the paragraph above. To do this, move the cursor with the mouse until it points to the word FILE that is found on the instruction line. Click the mouse and a pull-down menu will appear that shows all the options available with that command.

This is how the screen will look:

```
MAIN File Translate Dictionary Font Options
Save Load Delete Eject
```

Move the cursor to the option LOAD and click the button on the mouse. An option screen will appear that allows the loading of the file that is selected. Move the cursor to the file name of FILE 2 and click the mouse button. The name will be highlighted and should look like this:
Move the cursor to the bubble that surrounds the word LOAD and click the mouse button. The file FILE 2 will be loaded and you will be placed back to the main menu.

The next step is to begin the translation process on the file. Move the cursor to the word TRANSLATION on the instruction line of the main menu and click the mouse button to select it. A pull-down menu will appear with the options listed. The screen will look like this: