

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

1.0 OVERVIEW

Only a few years ago, a person living the American Dream had a good job and lived in an upscale neighborhood. Usually there was a significant commute between his house and job. Today, the declining job security and the rising fuel costs have rendered the "American Dream" a distant memory.

The restoration of the American Dream is tightly coupled to a significant reduction in the consumption of gasoline. Eventually, there will be alternate sources of fuel and transportation; however, these will not occur in the immediate future.

A significant reduction in fuel consumption will occur if everyone who can work at home did so. Many jobs cannot be done at home and not all people can perform the professional activities from their house. Those who can will reap the benefits from the reduction in the need to commute to work.

Software Engineering is a profession that can be done in most cases from home.

2.0 INTRODUCTION

I have been a software engineer since 1966. I was:

- Employed by Lockheed California Company from 1961 to 1975.
- Employed by Ocean Technology from 1975 to 1992
- Self-employed from 1992 to the present

I started working at home in 1980 when I purchased my first Personal Computer. For the remainder of my employment at Ocean Technology, I continued to work at home whenever possible. When I became self-employed, I provided my software development services for a number of off-site fixed price projects.

3.0 EMPLOYMENT RELATIONS

When I started working at home in 1980, it quickly apparent that my relation with my employer, Ocean Technology, would significantly change.

When I was working at the plant, my value to the company was not only defined by my professional performance, but also by my social skills, selection of clothing, reporting on time every morning and other non-performance related criteria.

When I was working at home, my value to my employer was determined exclusively by performance. In essence, I needed to perform all of my tasks successfully and one-time. It was like being a subcontractor instead of being an employee.

4.0 WORK AT HOME CHALLENGE

The need to perform all of my tasks successfully and on time caused me to re-examine my software development methods. Specifically, I needed to find economical means of developing error-free software that I could use as an individual. Of particular importance was the recognition that the client (or employer) is always an unwilling member of the test team.

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

5.0 MY RESPONSE

5.1 HISTORICAL CONSIDERATIONS

I learned very early in my career that the development of error-free software was feasible.

When I first joined the Scientific Computer Division at Lockheed in 1966, FORTRAN was the language of choice. The Fortran Compiler had a large never-ending bug list; new bugs were discovered as fast as IBM could fix them.

In the late 1960's, IBM introduced PL/I, a new computing language that combined the best of both FORTRAN and COBALT. They claimed that because the PL/I language was in strict compliance with the principles of "simple precedence", the compiler would be bug-free. It was bug free.

Bob Prince, a staff member of the Scientific Computer Division, developed a compiler writer based on the principles of "simple precedence". I used this program in a number of applications, some of which were not compilers. All of these programs were error-free.

5.2 HOW DOES SOFTWARE BREAK

We all know that software does not break. However, if I could find a hardware analog, I might be able to take advantage of my Ocean Technology experience in the development of Performance Monitoring and Fault Location sub-systems used to troubleshoot critical systems.

I found a practical analogy. Software breaks when it performs an illegal write. Furthermore, this failure is cannot be corrected after it occurs. Consequently, it must be prevented.

5.3 REVISED CODING METHODS

5.3.1 GENERAL CONSIDERATIONS

When I started to work at home, I realized very quickly that I would be responsible for any error that was present in my completed tasks. I could not pass the blame to another team member; even if I used his work-products to complete my task.

I needed a fairly formal development methodology that prevent and detect errors at many levels

5.3.2 THE IMPORTANCE OF A PLAN

My desire to provide my services off-site for a fixed price demanded that I make a plan. Most of these plans involved multiple tasks each of which resulted in a deliverable work product.

My latest contract was done predominately on-site at an hourly rate. The plan, in this case, was a detailed description of my software product. I then proceeded to strictly follow my plan. This action pleasantly surprised management; they were able to track my efforts with the confidence that I would complete my tasks ahead of schedule and under budget.

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

5.3.3 VISUALLY CONSIDERATE METHODS

It was obvious to me that working at home would significantly reduce the informal communication between me and the other team members. Consequently, it might be more difficult for my fellow employees to understand and use my work-products.

To facilitate this communication challenge, I did everything that I could to develop my work products from a set of small, self-documenting source files. Variable Names were not abbreviated. Comments were liberally used and the files were kept very small (three pages or less).

I found that this basic principle not helped my fellow employees and clients; it helped me also. I was always able to quickly answer a question over the phone even if a year or more had past since I had last worked on the software product.

5.3.4 SOFTWARE PERFORMANCE MONITORING AND FAULT LOCATION

High reliability military systems usually have some performance monitoring and fault location hardware to aid the user to quickly find and fix a problem. "A seventeen year old sailor with a fifth grade education had to be able to find and repair the fault within 15-minutes" is an important requirement on a submarine.

I needed to be able to quickly find and repair errors in my software products during execution (Please note that I assumed that I would make errors that I would not find). Furthermore, I needed to minimize the impact of my errors on my employer or client. I developed a collection of code segments that would efficiently perform this task.

5.3.5 MEMORY CYCLING

On my last contract, the principles of "Memory Cycling" greatly assisted me in rapidly debugging and testing my software product. "Memory Cycling" is a simple and very practical means of reducing the opportunity for an illegal write. A data element usually does not use all possible values. Specifically:

- A value that is not used is specified as the "empty" value. Writing to this data element is permitted only if the current value is the "empty" value.
- Reading from this data element is permitted only if the current value is not the "empty" value
- An error condition results when the above conditions are not met.

5.3.6 USE OF STATE MACHINES

Bob Prince at Lockheed used a state machine to develop his first compiler writer based on the principles of "simple precedence". As a consequence, I connected precedence analysis to the development of state machines. State machines tended to be self-documenting.

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

I chose extensive use of state machines in my last contract for the following reasons:

- Errors were generally in the form of missing processing paths which were discovered early in checkout.
- The state machine could be made rigorously compliant to the detailed description of the project contained in my plan
- The self-documenting nature of the state-machine made it easy to relate the source code to the plan.

5.3.7 MONTE-CARLO BASED TESTING

One of my last contract products was a controller of legacy hardware. A single hardware system was available as the Engineering Design Model. I was concerned about breaking it during checkout. Consequently I decided to develop a simulator for the hardware so that most of the checkout could be performed without access to the hardware.

I decided to build Monte-Carlo based simulator/stimulator test system. I could not only send a random sequence of commands to my product but also simulate both correct and errant hardware responses. The random nature of the Monte-Carlo method gave me a thorough exploration of the product. This enabled me to render the product error-free before testing it on the actual hardware.

5.4 ISO-9001 AND THE DEMING CYCLE

5.4.1 MOMENT OF DISCOVERY

On my last contract, I was exposed to ISO-9001 and the Deming Cycle. The client was considering getting ISO-9001 certified and it was quite apparent that their software group was going to have difficulties relating to this quality program.

I needed my own personal ISO-9001 program. The Deming cycle provided me with the means of accomplishing this goal. The four steps of the Deming Cycle are:

- Plan: Make a plan,
- Do: Execute the plan
- Check Evaluate your effort
- Analyze: Modify your plan as appropriate to improve your performance.

It was quite apparent that the Deming Cycle was a concise definition of common-sense. I could recall a number of unconscious Deming Cycle applications in my life. I found that being consciously aware of these principles was beneficial.

5.4.2 RECORD KEEPING

Very early in my career, I learned the value of keeping a log. Originally, it was an important expansion of the time-card in the Scientific Computer Division at Lockheed. Later, it provided management with needed historical project management data. On the last contract, it provided a weekly report that provided my client with a detailed status of my efforts.

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

To implement my own personal ISO-9001 program, I found it necessary to improve not only my record keeping process but also my planning process without greatly increasing the associated effort. An inexpensive Daily Organizer Program came to my rescue. It provided me with a planning document, a daily record, and a log of accomplishments. Furthermore, I improved my planning efforts to provide me each day with a list of significant accomplishments.

5.4.3 VALUE OF SPECIAL TOOLS

5.4.3.1 PROGRAMMABLE CODE GENERATOR

ISO-9001 requires that standard methods and processes be used. I first benefited from the use of standard methods developing software with a macro assembler. The macro command provided a means of extending the instruction set with commonly used groups of code. My collection of macros became my first set of standard methods; once I developed these macros, I never had the need to change them.

When I started using the C computing language, I discovered that the corresponding preprocessor define statement was significantly less competent than macro statement of the assembler. To rectify this problem, I developed a preprocessor application program. This program provided an equivalent macro capability to the C development environment. However, it did not extend the C language. As a consequent, the use of these macros was mandatory not optional. This property provided me with the capability of developing enforceable coding standards as required by ISO-9001,

5.4.3.2 PROGRAMMABLE MONTE-CARLO TEST SYSTEM

For most of my professional years, I tested my software products by the checkout process used by most software engineers. I never had a personal formal testing policy.

On my last contract, I needed to significantly raise my checkout standards. I developed the Monte-Carlo stimulator / simulator test system to thoroughly test my product as an integral part of my checkout process. This test system enabled me to deliver an error-free product ahead of schedule,

During this contract, I had a significant discussion with the engineer responsible for software quality regarding the ability of various coding methods to resist error. When the ISO-9001 meetings occurred, it became obvious that I needed to evaluate my software standards relative to their ability to resist error.

To satisfy this requirement, I developed a Programmable Monte-Carlo Test system. This program had the additional benefit of being able to test software work products from the individual collections of macros to the executable.

This program greatly raised the bar relative to error-resistant methods of the work products. Furthermore, most of the errors were detected during the early stages of development.

6.0 SUMMARY

In a normal employment relation, management defines the goals of the company, defines the tasks by which the goals are accomplished, and directs the workers to perform the tasks. The worker is not involved in any significant way in the development of the goals or the sequence of tasks needed to accomplish the goals.

TITLE SOFTWARE ENGINEERING FROM THE HOME
WRITTEN BY: ROBERT ADAMS
DATE: JUNE 11, 2008
WEB-SITE: www.whatifwe.com
EMAIL: robert.adams@whatifwe.com

In manufacturing companies, the application of quality programs such as ISO-9001 is strictly a management effort. College courses that address ISO-9001 and the Deming Cycle are offered in Business Administration schools. They are not offered in engineering schools as a technical subject.

Software Engineering is an important exception to the traditional application of the Deming Cycle. The computer program is a Deming plan to be done at the client site by a computer, not a human being. The Programmer is a quality manager.

The software engineering employee that works at home must think of himself as a sub-contractor. His employer must trust him to deliver an error-free work-product that supports the company goals within budget and schedule every time. He must develop a level of technical competence that renders the execution of his responsibilities routine not challenging.

If he can gain the trust of his employer, both he and his employer will reap many benefits. For example

- **Considerably less commuting costs:** During the last two years at Ocean Technology, I commuted to the plant once a week on the average.
- **Greater Availability to the family:** My kids always had a parent at home. My mere presence was of considerable benefit to the growth of my kids.
- **Better Health for me and my fellow employees:** The office is a breeding ground for many common illnesses. Usually, it takes two weeks to recover from the flu or the common cold. A full recovery will exhaust the employer's yearly sick-leave allowance. As a consequence, the employee comes back to work before he is fully recovered and either has a relapse or spreads the illness to someone else. When I got sick, I would call my employer and go on a reduced daily schedule. I was able to still get my job done, get the needed rest, and not contribute to the spread of the illness at the plant.