TITLE: SOFTWARE ENGINEERING AND THE DEMING CYCLE
FROM ROBERT ADAMS
WEB-SITE www.whatifwe.com
EMAIL: robert.adams@whatifwe.com
DATE: MARCH 05, 2008

1.0 PROJECT GOALS

My overall goal of this project is to improve the quality of software that we use for critical applications in the United States. The rigorous application of the Deming Cycle to software engineering is critical to my pursuit of this goal

The best means of accomplishing this goal is to develop a serious international competitor to the United States based software industry. I believe that Japanese industry with considerable experience with the Deming Cycle can provide this competition.

2.0 HISTORICAL CONSIDERATIONS

2.1 JAPANESE-DEMING HISTORY

The Deming Cycle, also known as "Plan-Do-Check-Act" (PDCA) cycle can be briefly described as follows:

- **PLAN:** Establish methods and processes necessary to deliver results in accordance with customer requirements and the organization policies".
- **DO:** Implement the process.
- **CHECK:** Monitor and measure the processes and product against policies, objectives, and requirements for the product and report the results.
- **ACT:** Take action to continuously improve process performance.

Dr. W. Edwards Deming is known as the father of the Japanese post-war industrial revival and was regarded by many as the leading quality guru in the United States.

He was invited to Japan at the end of World War II by Japanese industrial leaders and engineers to assist them to shift the perception of the world from the existing paradigm that Japan produced cheap, shoddy imitations to one of producing innovative quality products.

The Japanese industries were successful.

2.2 ADAMS-SOFTWARE HISTORY[1]

Robert Adams began his quest for error-free software in the late 1960's. Major milestones in this pursuit are presented as follows:

- **Computer Language Compiler Writer (1971-1975)**: Used a Lockheed-developed simple precedence-based compiler writer to develop several error-free application programs.
- **Low Cost On-Site Custom Software Development (1992-1993):** Developed a Code Generator Program capable of rapidly developing a custom Medicare Progress Notes Editor. The primary challenge was verifying that these editors were error-free. .
- **Software Failure Modes (1993-2001):** Determined that software "breaks" irretrievably when it makes an erroneous write. Developed formal error detection methods to anticipate an erroneous write, prepare a report, and terminate execution.

[1] resume found on www.whatifwe.com

- **Programmable Code Generator (1993-2001)**. The PCG was developed to facilitate the implementation of the error anticipation and reporting methods. This program facilitated the development of standard error handling methods.
- **ISO-9001 Quality Standard Meeting (2003):** An ISO-9001 Meeting revealed that the PCG could satisfy the enforceable standard methods and procedures requirement of ISO-9001. It did not satisfy the ISO9001 quality evaluation requirements.
- **Programmable Monte-Carlo Test System (2003-2005):** A Monte-Carlo based test system was developed to satisfy the ISO-9001 requirement to evaluate the quality of the software development process.
- **Deming Cycle, Software Connection (2007):** Determined that a formal, enforceable application of the Deming Cycle to software engineering can be made by noting that the software product is a Deming Cycle planning document to be implemented at the customer site by a computer, not a human being.[2]

3.0 CURRENT BASELINE

3.1 RELEASED PRODUCTS[3]

3.1.1 PROGRAMMABLE CODE GENERATOR

The Programmable code generator (PCG) enables the provider to use enforceable standard methods and processes in the development of a software product. Specifically, the PCG redefines the programming effort as a formal two-step process:

- The development of a library of “software parts”.
- The use of this library in the development of product.

This two step operation gives management and the software engineering staff greater control over the quality of the product.

3.1.2 PROGRAMMABLE MONTE-CARLO TEST SYSTEM

It is not sufficient to demonstrate that a computer program will function as required when properly and reasonably used. To obtain high reliability, the product must be developed with the most error-resistant methods and tested over a broad spectrum of both reasonable and unreasonable circumstances.

The Programmable Monte-Carlo Test system (PMTS) was specifically developed to accomplish the task. This program explores the executable, module or software parts library with a user-defined test sequence process and records the results.

The use of a random generator to select the test conditions insures that a comprehensive exploration of the test conditions will be used; not just those that are reasonable or expected.

The Programmable Monte-Carlo Test System, when applied to a library of software parts can measure its resistance to programming error.

[2] Deming white paper found on www.whatifwe.com
[3] Open source products found on www.whatifwe.com

3.2 PROGRAMMABLE SOFTWARE DEVELOPMENT ENVIRONMENT

3.2.1 BASELINE PROPERTIES

The Programmable Code Generator and Monte-Carlo Test system have been combined into an integrated Programmable Software Development Environment which is in the final stage of checkout and documentation. Included in the upgrade are the following features

- **Software Kit Products:** The ability to combine multiple software kit libraries into a project library enables the division of a large project into manageable tasks that can be performed at different times and by different people. This capability will also enable independent organizations to develop and market software kit libraries.
- **User-Friendly Input Languages:** A user-defined text oriented input file and simple-precedence based operations provides the user with the capability of developing a custom input language for their specific needs

This system supports all aspects of the Deming Cycle and ISO9001 but does not currently support the Unicode International character set.

3.3 INTERNATIONAL LANGUAGE CONSIDERATIONS

3.3.1 PRECEDENCE ANALYSIS

In the late 1960's, a methodology for designing error-free compilers was developed. Specifically, if the rules for writing statements satisfy the criteria for *"simple precedence",* the corresponding rules for reading statements will properly parse every correctly written statement no matter how complex and detect every error no matter how subtle.

The rules of *"simple precedence"* are defined independently of the character set, and there is considerable capability of handling contextual situations. Furthermore the development of a simple-precedence translator is quite easy if the language can be written down in ink as it is being spoken.

3.3.2 DEVELOPMENT SYSTEM INTERNATIONAL CAPABILITIES

The compilers, link-editors and other software development tools are based on the English-based ASCII character set. These systems have been extended to handle Unicode International Character Set from a data point-of-view; but not from a programming point of view.

Could a practical link between the statements in the international input language could be converted into a useful equivalent compiler friendly form?

A simple experiment revealed that Microsoft Visual Studio.net would accept a Unicode-based source file with comment statements written in an international language. Any well defined translator can produce statements that are acceptable to the compiler; and comments that are written in the original international input language.

Having both the international comment and the equivalent computer-friendly statement in close proximity to each-other in the source files will greatly facilitate the development process using the US development software tools,

4.0 PLANNED EXTENSION

4.1 UNICODE INTERNATIONAL CHARACTERS

The Programmable Development Environment, currently in the final stages of checkout, is being used to re-develop itself. Special attention is being given to a Unicode International version. Full Unicode capability will be provided to all text and string capabilities. Additional commands will be added as necessary to support the international character set.

4.2 INTERFACE PARTS

The development of a Japanese based computer development language will be developed as the first real test case of the International Programmable Software Development Environment. A Japanese-American Software Engineer and one or more Japanese Industrial Clients will be prime in this activity. Robert Adams will support their activities as needed to insure that the Programmable Software Development Environment is optimal to this activity,

5.0 RESOURCES REQUIRED

5.1 JAPANESE-AMERICAN SOFTWARE ENGINEER

A Japanese-American Software Engineering Professional or talented student is critical to the success of the project. Experience in the development of simple-precedence computer language translators would be very beneficial. The individual should have equal understanding of the Japanese and American cultures and be proficient in both languages. It is expected that this individual would be the son or daughter of an immigrant from Japan.

5.2 JAPANESE CLIENTS

To properly develop a useful Japanese-American Computer Development Interface, one or more Japanese Industries (located in Japan) interested in its success must be found. These organizations must be sufficiently interested to provide advice and council as necessary to render it applicable to their industry.

5.3 BUSINESS MODULE

The business model is “to be determined”. It could start as a non-profit in which the initial benefactors would fund the project. It might eventually be transformed into a franchise in which a number of Japanese-American software engineers would be the “franchisees”. American funding would also be beneficial

6.0 CONCLUDING REMARKS

After World War II, the United States ignored the potential benefits of the Deming Cycle and the Japanese took advantage of them. As a consequence, the United States lost its leading position several industries including the automotive industry.

If Japanese Industry successfully made the practical connection between Software Engineering and the Deming Cycle, history would repeat itself. It would be very beneficial to the whole computing industry to develop some serious competition for Microsoft. With the right competitive stimulus, perhaps the American Industry, this time, will learn from the Japanese.