Tao of Programming

Translated By Geoffrey James
Transcribed By Seth Robertson

Table of Contents

The Silent Void
Book One

Thus spake the master programmer:

"When you have learned to snatch the error code from the trap frame, it will be time for you to leave."

1.1

Something mysterious is formed, born in the silent void. Waiting alone and unmoving, it is at once still and yet in constant motion. It is the source of all programs. I do not know its name, so I will call it the Tao of Programming.

If the Tao is great, then the operating system is great. If the operating system is great, then the compiler is great. If the compiler is greater, then the applications is great. The user is pleased and there is harmony in the world.

The Tao of Programming flows far away and returns on the wind of morning.

1.2

The Tao gave birth to machine language. Machine language gave birth to the assembler.

The assembler gave birth to the compiler. Now there are ten thousand languages.

Each language has its purpose, however humble. Each language expresses the Yin and Yang of software. Each language has its place within the Tao.

But do not program in COBOL if you can avoid it.

1.3

In the beginning was the Tao. The Tao gave birth to Space and Time. Therefore, Space and Time are the Yin and Yang of programming.

Programmers that do not comprehend the Tao are always running out of time and space for their programs. Programmers that comprehend the Tao always have enough time and space to accomplish their goals.

How could it be otherwise?

1.4

The wise programmer is told about the Tao and follows it. The average programmer is told about the Tao and searches for it. The foolish programmer is told about the Tao and laughs at it.

If it were not for laughter, there would be no Tao.

The highest sounds are the hardest to hear. Going forward is a way to retreat. Greater talent shows itself late in life. Even a perfect program still has bugs.

The Ancient Masters
Book Two

Thus spake the master programmer:

"After three days without programming, life becomes meaningless."

2.1

The programmers of old were mysterious and profound. We cannot fathom their thoughts, so all we do is describe their appearance.

Aware, like a fox crossing the water. Alert, like a general on the battlefield. Kind, like a hostess greeting her guests. Simple, like uncarved blocks of wood. Opaque, like black pools in darkened caves.

Who can tell the secrets of their hearts and minds?

The answer exists only in the Tao.

2.2

Grand Master Turing once dreamed that he was a machine. When he awoke he exclaimed:

> "I don't know whether I am Turing dreaming that I am a machine, or a machine dreaming that I am Turing!"

2.3

A programmer from a very large computer company went to a software conference and then returned to report to his manager, saying: "What sort of programmers work for other companies? They behaved badly and were unconcerned with appearances. Their hair was long and unkempt and their clothes were wrinkled and old. They crashed out hospitality suites and they made rude noises during my presentation."

The manager said: "I should have never sent you to the conference. Those programmers live beyond the physical world. They consider life

absurd, an accidental coincidence. They come and go without knowing limitations. Without a care, they live only for their programs. Why should they bother with social conventions?"

"They are alive within the Tao."

2.4

A novice asked the Master: "Here is a programmer that never designs, documents, or tests his programs. Yet all who know him consider him one of the best programmers in the world. Why is this?"

The Master replies: "That programmer has mastered the Tao. He has gone beyond the need for design; he does not become angry when the system crashes, but accepts the universe without concern. He has gone beyond the need for documentation; he no longer cares if anyone else sees his code. He has gone beyond the need for testing; each of his programs are perfect within themselves, serene and elegant, their purpose self-evident. Truly, he has entered the mystery of the Tao."

Design
Book Three

Thus spake the master programmer:

"When program is being tested, it is too late to make design changes."

3.1

There once was a man who went to a computer trade show. Each day as he entered, the man told the guard at the door:

> "I am a great thief, renowned for my feats of shoplifting. Be forewarned, for this trade show shall not escape unplundered."

This speech disturbed the guard greatly, because there were millions of dollars of computer equipment inside, so he watched the man carefully. But the man merely wandered from booth to booth, humming quietly to himself.

When the man left, the guard took him aside and searched his clothes, but nothing was to be found.

On the next day of the trade show, the man returned and chided the guard saying: "I escaped with a vast booty yesterday, but today will be even better." So the guard watched him ever more closely, but to no avail.

On the final day of the trade show, the guard could restrain his curiosity no longer. "Sir Thief," he said, "I am so perplexed, I cannot live in peace. Please enlighten me. What is it that you are stealing?"

The man smiled. "I am stealing ideas," he said.

3.2

There once was a master programmer who wrote unstructured programs. A novice programmer, seeking to imitate him, also began to write unstructured programs. When the novice asked the master to evaluate his progress, the master criticized him for writing unstructured

programs, saying: "What is appropriate for the master is not appropriate for the novice. You must understand the Tao before transcending structure."

3.3

There was once a programmer who was attached to the court of the warlord of Wu. The warlord asked the programmer: "Which is easier to design: an accounting package or an operating system?"

"An operating system," replied the programmer.

The warlord uttered an exclamation of disbelief. "Surely an accounting package is trivial next to the complexity of an operating system," he said.

"Not so," said the programmer, "when designing an accounting package, the programmer operates as a mediator between people having different ideas: how it must operate, how its reports must appear, and how it must conform to the tax laws. By contrast, an operating system is not limited my outside appearances. When designing an operating system, the programmer seeks the simplest harmony between machine and ideas. This is why an operating system is easier to design."

The warlord of Wu nodded and smiled. "That is all good and well, but which is easier to debug?"

The programmer made no reply.

3.4

A manager went to the master programmer and showed him the requirements document for a new application. The manager asked the master: "How long will it take to design this system if I assign five programmers to it?"

"It will take one year," said the master promptly.

"But we need this system immediately or even sooner! How long will it take it I assign ten programmers to it?"

The master programmer frowned. "In that case, it will take two years."

"And what if I assign a hundred programmers to it?"

The master programmer shrugged. "Then the design will never be completed," he said.

Coding
Book Four

Thus spake the master programmer:

"A well-written program is its own heaven;
a poorly-written program is its own hell."

4.1

A program should be light and agile, its subroutines connected like a strings of pearls. The spirit and intent of the program should be retained throughout. There should be neither too little nor too much,

neither needless loops nor useless variables, neither lack of structure nor overwhelming rigidity.

A program should follow the 'Law of Least Astonishment'. What is this law? It is simply that the program should always respond to the user in the way that astonishes him least.

A program, no matter how complex, should act as a single unit. The program should be directed by the logic within rather than by outward appearances.

If the program fails in these requirements, it will be in a state of disorder and confusion. The only way to correct this is to rewrite the program.

4.2

A novice asked the master: "I have a program that sometimes runs and sometimes aborts. I have followed the rules of programming, yet I am totally baffled. What is the reason for this?"

The master replied: "You are confused because you do not understand the Tao. Only a fool expects rational behavior from his fellow humans. Why do you expect it from a machine that humans have constructed? Computers simulate determinism; only the Tao is perfect.

The rules of programming are transitory; only the Tao is eternal. Therefore you must contemplate the Tao before you receive enlightenment."

"But how will I know when I have received enlightenment?" asked the novice.

"Your program will then run correctly," replied the master.

4.3

A master was explaining the nature of the Tao to one of his novices, "The Tao is embodied in all software -- regardless of how insignificant," said the master.

"Is the Tao in a hand-held calculator?" asked the novice.

"It is," came the reply.

"Is the Tao in a video game?" continued the novice.

"It is even in a video game," said the master.

"And is the Tao in the DOS for a personal computer?"

The master coughed and shifted his position slightly. "The lesson is over for today," he said.

4.4

Price Wang's programmer was coding software. His fingers danced upon the keyboard. The program compiled without an error message, and the program ran like a gentle wind.

Excellent!" the Price exclaimed, "Your technique is faultless!"

"Technique?" said the programmer, turning from his terminal, "What I follow is the Tao -- beyond all technique. When I first began to program I would see before me the whole program in one mass. After three years I no longer saw this mass. Instead, I used subroutines. But now I see nothing. My whole being exists in a formless void. My senses are idle. My spirit, free to work without a plan, follows its own instinct. In short, my program writes itself. True, sometimes there are difficult problems. I see them coming, I slow down, I watch silently. Then I change a single line of code and the difficulties vanish like puffs of idle smoke. I then compile the program. I sit still and let the joy of the work fill my being. I close my eyes for a moment and then log off."

Price Wang said, "Would that all of my programmers were as wise!"

Maintenance
Book Five

Thus spake the master programmer:

"Though a program be but three lines long,
someday it will have to be maintained."

5.1

A well-used door needs no oil on its hinges.
A swift-flowing steam does no grow stagnant.
Neither sound nor thoughts can travel through a vacuum.
Software rots if not used.

These are great mysteries.

5.2

A manager asked a programmer how long it would take him to finish the program on which he was working. "I will be finished tomorrow," the programmer promptly replied.

"I think you are being unrealistic," said the manager. "Truthfully, how long will it take?"

The programmer thought for a moment. "I have some features that I wish to add. This will take at least two weeks," he finally said.

"Even that is too much to expect," insisted the manager, "I will be satisfied if you simply tell me when the program is complete."

The programmer agreed to this.

Several years slated, the manager retired. On the way to his retirement lunch, he discovered the programmer asleep at his terminal. He had been programming all night.

5.3

A novice programmer was once assigned to code a simple financial package.

The novice worked furiously for many days, but when his master reviewed his program, he discovered that it contained a screen editor, a set of

generalized graphics routines, and artificial intelligence interface, but not the slightest mention of anything financial.

When the master asked about this, the novice became indignant. "Don't be so impatient," he said, "I'll put the financial stuff in eventually."

5.4

Does a good farmer neglect a crop he has planted?
Does a good teacher overlook even the most humble student?
Does a good father allow a single child to starve?
Does a good programmer refuse to maintain his code?

Management
Book Six

Thus spake the master programmer:

"Let the programmer be many and the managers
few -- then all will be productive."

6.1

When managers hold endless meetings, the programmers write games. When accountants talk of quarterly profits, the development budget is about to be cut. When senior scientists talk blue sky, the clouds are about to roll in.

Truly, this is not the Tao of Programming.

When managers make commitments, game programs are ignored. When accountants make long-range plans, harmony and order are about to be restored. When senior scientists address the problems at hand, the problems will soon be solved.

Truly, this is the Tao of Programming.

6.2

Why are programmers non-productive?
Because their time is wasted in meetings.

Why are programmers rebellious?
Because the management interferes too much.

Why are the programmers resigning one by one?
Because they are burnt out.

Having worked for poor management, they no longer value their jobs.

6.3

A manager was about to be fired, but a programmer who worked for him invented a new program that became popular and sold well. As a result, the manager retained his job.

The manager tried to give the programmer a bonus, but the programmer refused it, saying, "I wrote the program because I though it was an interesting concept, and thus I expect no reward."

The manager, upon hearing this, remarked, "This programmer, though he holds a position of small esteem, understands well the proper duty of an employee. Lets promote him to the exalted position of management consultant!"

But when told this, the programmer once more refused, saying, "I exist so that I can program. If I were promoted, I would do nothing but waste everyone's time. Can I go now? I have a program that I'm working one."

6.4

A manger went to his programmers and told them: "As regards to your work hours: you are going to have to come in at nine in the morning and leave at five in the afternoon." At this, all of them became angry and several resigned on the spot.

So the manager said: "All right, in that case you may set your own working hours, as long as you finish your projects on schedule." The programmers, now satisfied, began to come in a noon and work to the wee hours of the morning.

Corporate Wisdom
Book Seven

Thus spake the master programmer:

"You can demonstrate a program for a corporate executive, but you can't make him computer literate."

7.1

A novice asked the master: "In the east there is a great tree-structure that men call 'Corporate Headquarters'. It is bloated out of shape with vice-presidents and accountants. It issues a multitude of memos, each saying 'Go, Hence!' or 'Go, Hither!' and nobody knows what is meant. Every year new names are put onto the branches, but all to no avail. How can such an unnatural entity exist?"

The master replies: "You perceive this immense structure and are disturbed that it has no rational purpose. Can you not take amusement from its endless gyrations? Do you not enjoy the untroubled ease of programming beneath its sheltering branches? Why are you bothered by its uselessness?"

7.2

In the east there is a shark which is larger than all other fish. It changes into a bird whose winds are like clouds filling the sky. When this bird moves across the land, it brings a message from Corporate Headquarters. This message it drops into the midst of the programmers, like a seagull making its mark upon the beach. Then the bird mounts on the wind and, with the blue sky at its back, returns home.

The novice programmer stares in wonder at the bird, for he understands it not. The average programmer dreads the coming of the bird, for he fears its message. The master programmer continues to work at his terminal, for he does not know that the bird has come and gone.

7.3

The Magician of the Ivory Tower brought his latest invention for the master programmer to examine. The magician wheeled a large black box into the master's office while the master waited in silence.

"This is an integrated, distributed, general-purpose workstation," began the magician, "ergonomically designed with a proprietary operating system, sixth generation languages, and multiple state of the art user interfaces. It took my assistants several hundred man years to construct. Is it not amazing?"

The master raised his eyebrows slightly. "It is indeed amazing," he said.

"Corporate Headquarters has commanded," continued the magician, "that everyone use this workstation as a platform for new programs. Do you agree to this?"

"Certainly," replied the master, "I will have it transported to the data center immediately!" And the magician returned to his tower, well pleased.

Several days later, a novice wandered into the office of the master programmer and said, "I cannot find the listing for my new program. Do you know where it might be?"

"Yes," replied the master, "the listings are stacked on the platform in the data center."

7.4

The master programmer moves from program to program without fear. No change in management can harm him. He will not be fired, even if the project is canceled. Why is this? He is filled with the Tao.

Hardware and Software
Book Eight

Thus spake the master programmer:

"Without the wind, the grass does not move.
Without software, hardware is useless."

8.1

A novice asked the master: "I perceive that one computer company is much larger than all others. It towers above its competition like a giant among dwarfs. Any one of its divisions could comprise an entire business. Why is this so?"

The master replied, "Why do you ask such foolish questions? That company is large because it is so large. If it only made hardware, nobody would buy it. If it only maintained systems, people would treat it like a servant. But because it combines all of these things, people think it one of the gods! By not seeking to strive, it conquers without effort."

8.2

A master programmer passed a novice programmer one day. The master

noted the novice's preoccupation with a hand-held computer game. "Excuse me", he said, "may I examine it?"

The novice bolted to attention and handed the device to the master. "I see that the device claims to have three levels of play: Easy, Medium, and Hard", said the master. "Yet every such device has another level of play, where the device seeks not to conquer the human, nor to be conquered by the human."

"Pray, great master," implored the novice, "how does one find this mysterious setting?"

The master dropped the device to the ground and crushed it under foot. And suddenly the novice was enlightened.

8.3

There was once a programmer who worked upon microprocessors. "Look at how well off I am here," he said to a mainframe programmer who came to visit, "I have my own operating system and file storage device. I do not have to share my resources with anyone. The software is self-consistent and easy-to-use. Why do you not quit your present job and join me here?"

The mainframe programmer then began to describe his system to his friend, saying: "The mainframe sits like an ancient sage meditating in the midst of the data center. Its disk drives lie end-to-end like a great ocean of machinery. The software is a multi-faceted as a diamond and as convoluted as a primeval jungle. The programs, each unique, move through the system like a swift-flowing river. That is why I am happy where I am."

The microcomputer programmer, upon hearing this, fell silent. But the two programmers remained friends until the end of their days.

8.4

Hardware met Software on the road to Changtse. Software said: "You are the Yin and I am the Yang. If we travel together we will become famous and earn vast sums of money." And so the pair set forth together, thinking to conquer the world.

Presently, they met Firmware, who was dressed in tattered rags, and hobbled along propped on a thorny stick. Firmware said to them: "The Tao lies beyond Yin and Yang. It is silent and still as a pool of water. It does not seek fame, therefore nobody knows its presence. It does not seeks fortune, for it is complete within itself. It exists beyond space and time."

Software and Hardware, ashamed, returned to their homes.

Epilogue
Book Nine

Thus spake the master programmer:

"Time for you to leave."

The Tao of Programming flows far away and returns on the wind of morning. 1.2. The Tao gave birth to machine language. Machine language gave birth to the assembler. The assembler gave birth to the compiler.Â Programmers that do not comprehend the Tao are always running out of time and space for their programs. Programmers that comprehend the Tao always have enough time and space to accomplish their goals. How could it be otherwise? 1.4. The Tao of Programming flows far away and returns on the wind of morning. 1.2. The Tao gave birth to machine language. Machine language gave birth to the assembler. The assembler gave birth to the compiler. Now their are ten thousand languages. Each language has its purpose, however humble. Each language expresses the Yin and Yang of software.Â Programmers that comprehend the Tao always have enough time and space to accomplish their goals. How could it be otherwise? 1.4. The wise programmer is told about Tao and follows it. The average programmer is told about Tao and searches for it. The foolish programmer is told about Tao and laughs at is. If it were not for laughter, there would be no Tao. The highest sounds are hardest to hear. The Tao of Programming flows far away and returns on the wind of morning. 1.2. The Tao gave birth to machine language. Machine language gave birth to the assembler. The assembler gave birth to the compiler.Â A program should be light and agile, its subroutines connected like a string of pearls. The spirit and intent of the program should be retained throughout. There should be neither too little or too much, neither needless loops nor useless variables, neither lack of structure nor overwhelming rigidity. A program should follow the â€˜Law of Least Astonishment.â€™ What is this law? It is simply that the program should always respond to the user in the way that astonishes him least. A program, no matter how complex, should act as a single unit. Programmers that comprehend the Tao always have enough time and space to accomplish their goals. How could it be otherwise?Â Truly, this is not the Tao of Programming. When managers make commitments, game programs are ignored. When accountants make long-range plans, harmony and order are about to be restored. When senior scientists address the problems at hand, the problems will soon be solved. Truly, this is the Tao of Programming. 6.2. Why are programmers non-productive? The Tao of Programming book. Read 25 reviews from the world's largest community for readers.Â Goodreads helps you keep track of books you want to read. Start by marking â€œThe Tao of Programmingâ€ as Want to Read: Want to Read savingâ€¦ Want to Read. Currently Reading. Read. Other editions. Enlarge cover.