

Sword and Spirit

The eJournal of Itten Dojo

July 2021



— Why Budo? —

Regardless of the times in which you live, or the circumstances of your life, success largely depends on things you can control:

- **Building a foundation of strong relationships in a community of mutual support and achievement.**
- **Forging a disciplined and positive mindset.**
- **Enhancing your physical health and capabilities.**

These are exactly the benefits membership in a dojo provides.

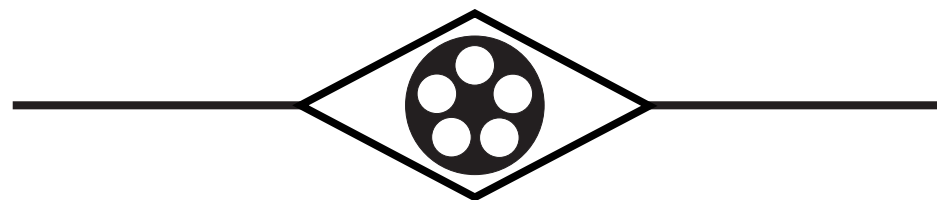
Copyright 2021 Itten Dojo, Inc.
701 W Simpson Street, Suite C
Mechanicsburg, PA 17055-3716
www.ittendojo.org

Problem Solving in Programming and Budo

Many people will tell you that computer programming is abundantly complicated to learn, and you have to be highly intelligent to be successful. Although some God-given intelligence does help with learning to program, it is not a requirement. Let it be known that anyone can learn to code. The same people will tell you that martial arts are abundantly complicated and difficult to learn, and you have to be physically gifted to make any progress. Similarly, starting physically gifted is not the measure of whether a student will be successful or not. A more accurate measure of whether someone can learn to code or is able to practice martial arts successfully is if they are willing to put in the effort necessary and the time required. Most skills are much the same, no matter if it is playing guitar, lifting weights, or writing a paper; anyone can learn and become better if effort and practice are put forth. Humans are remarkable when it comes to learning new skills, and it is simply a matter of honing your skills over time.

I have learned a great deal in my short time practicing martial arts. Some lessons were physical in nature, while others were more ideas or principle-based. The real secret is that the majority of what I learned was directly applicable to my life as much as it was to my martial arts practice. Training has directly influenced and illuminated other areas of everyday life—some of these lessons I learned from martial arts are straightforward and non-esoteric principles. Take, for example, learning to keep better posture, have special awareness, or maintaining focus under pressure. This essay will address a seemingly more esoteric application of martial arts principles, comparing problem-solving as it is found in martial arts to problem-solving as it pertains to computer science and the principles that connect them, as they relate to efficiency, nomenclature, timing, and ongoing training and development.

Sun Tzu's *The Art of War* has this maxim: "When you engage in actual fighting, if victory is long in coming, then men's weapons will grow dull, and their ardor will be damped." (Tzu, 5) One of the many ways to interpret this



quote from Sun Tzu would be an exercise in efficiency. If an army or state can win a fight without fighting, it would be wise to do so, as not having to fight or go to war would save time, money, and resources. Sun Tzu's maxim applies not only to states engaged in combat and to an individual's own martial arts practice, but it can also apply to coding. It is important not to reinvent the wheel every time a programmer needs a program to function a certain way. Often, it is better to call in an Application Programming Interface (API), a set of already written functions or use existing and tested code rather than writing your own. The principle is only to do what is needed to obtain the desired outcome, and the simplest solution may not always be taking physical action.

The efficiency of movement could be described as taking out extra steps. Efficiently using one's body could be described as creating a structure that leverages the supplied mechanical power. Efficiently reacting could be described as training an effective response to minimize reaction time. In computer programming, overly complex code with redundancies can most often be solved in a similar vein to movement efficiency in martial arts by "taking out the extra steps" (Foote, 269). Cleverly using computer resources can solve many common bugs and errors in a manner similar to using one's body efficiently.



When coding, it is also essential to maintain system resources such as memory—memory leaks exemplify this. A memory leak is when a programmer creates a memory store in a heap but does not delete it, like an expired encyclopedia taking up space on a shelf. A programmer should always release memory when it is no longer needed (*What is Memory Leak? How can we avoid?*). This is similar to creating a structure that uses mechanical power in your body. A close inspection of the timing of a function can make all the difference in a computer program being able to run (Foote, 233), similar to reacting efficiently.

This could be as simple as using an if-then statement, "If X then do Y." In other cases, a programmer may need to create event listeners, where if the user clicks in a certain spot then the computer does X. In programming and martial arts, the way to make kata or a program better is the same. Use only what you need, use what you have as efficiently and effectively as possible, and plan responses beforehand.

In an essay titled "What's The Point?," in the September 2016 issue of this journal, Robert Wolfe details what area of the foot to put your weight on to perform a turn in order to protect your knees and move efficiently. According to Mr. Wolfe, there are three points to consider: The point inside the ball of the foot under the big toe, the point outside the ball of the foot under the little toe, and the heel of the foot. Where do you put your weight, when? Your choice matters. Mr. Wolfe is talking about preventing mechanical failure without sacrificing efficiency and accuracy in movement. He does so by making clear and meaningful distinctions between the parts of the foot.

The governing principle of Mr. Wolfe's movement drills is similar to the one expressed in Robert C. Martin's book *Clean Code*, by making meaningful distinctions a clear rule. Martin recommends that when naming a variable, use a full meaning name to avoid confusion and minimize possible failure (Martin, 20). For example, by calling a variable "var1," the only information another programmer could gain from this variable name is that it is a variable and was most likely the first one. A variable called "CarsOnTheRoad" would give another programmer much more detail. Using this name, an



unfamiliar programmer could guess what type of information the variable would hold and the variable's data type. It is not good enough just to move from one place or stance to another with no regard for how you get there—in martial arts, specificity matters. How you move and the choices you make are as important as the destination. Similarly, when coding, it is not sufficient to satisfy the compiler for the program to run, rather, the choices you make in how you get there can make all the difference. The principle to know is to be specific.

Another way to solve coding problems by not adding unnecessary bugs in a large coding project would be to use design patterns (Freeman). It is unnecessary to reinvent the wheel for each project. Instead, a programmer's time can be better invested by reusing code and solving similar problems in similar ways. This is also true in martial arts. Take, for example, sheathing the sword, or *noto*. It would not serve a martial artist to sheath a sword in many different ways without any thought as to the reason for doing so. A student should learn when and why they sheath a sword with one approach over another, instead. An example of this in coding is the Strategy pattern. The Strategy pattern encapsulates a family of algorithms so

they can vary independently from the clients, the part of a program using the algorithms (Freeman, 24). Simply put, it allows a programmer to update an algorithm in one place instead of having to update the algorithm in every place that it is used. In both martial arts and coding, the principle is the same: Use already tested responses instead of reinventing the wheel each time.


A student doing kata without regard for the timing of the movements isn't practicing nearly as effectively as they could. Consider *Ki-musubi* ("Spirits-tied"), a fundamental, paired sword-form found in Sanshu-ho Aikibudo. In *Ki-musubi*, there are four distinct speeds or tempos that need to be incorporated for the kata to be correct: Slow, Quick, Steady, and Surge. These tempos are just as important to the kata as the overall pattern and steps themselves. Timing is also important for a programmer, as failing to consider when a line of code is executed can lead to issues. If the code makes a call to an external source, it may be essential to halt other lines of code until a response is received. In JavaScript, when calling an external API, one way is to use a "wait function" that halts the code until the call has received a response.



The order that lines of code will be executed must also be considered to avoid errors while coding. If a line of code needs a resource, the currently running line it is using must wait for the resource to be released. This is similar to the idea of *Go no Sen* in martial arts, where one waits for the opponent to attack and then responds (Weisgard). Or maybe a programmer needs a line of code to execute before any other line; to a martial artist, this would be *Sen no Sen*, “attack the attack” or respond faster than your opponent (Weisgard). A programmer may even need a line to be loaded before a program starts by declaring a variable to be static. *Sen Sen No Sen*, or taking the initiative and attacking before the attack (Weisgard).

Looking broadly, the stages of software development are Requirements Gathering, Software Design, Software Development, Test and Integration, Deployment, and Operationalization and Maintenance (Jachja). It is worth detailing the last step, Operationalization and Maintenance, where there is never a point where a project is good enough. Instead, there is always more work to do, be that security, fixing bugs, or continued optimization. Both software and martial arts must continue to evolve. The last principle here can be summed up as, “You are either growing or dying.”

Practicing martial arts has changed perspectives I once took for granted. Getting out of the mindset that makes coding difficult is also practiced in the martial arts. Will martial arts make you a great programmer? Maybe not, but it will make you a more well-rounded person with skills that can be applied to both areas. Computer programming may be done quickly and easily by just slapping down what you want a computer to do without concern for all of the coding processes outlined in this essay. A poorly designed program may compile or even run, but it will be full of bugs and security issues if the program has any bit of complexity to it. For the majority of programming issues to be avoided, all a programmer has to do is put in the work and follow best practices.

When we practice martial arts, we are fundamentally studying how to put in the time and work that will allow us to continue to be more efficient and evolve. Too often, people give up when learning gets difficult, or they look for easy solutions. Training in martial arts helps us avoid the mentality that some things are too hard to do. A better way to think of approaching new skills would be deciding if it is worth the effort and time to understand and apply in multiple ways. This is how martial arts fundamentally relates to computer programming and life in general. 

Foot, S. (2015). *Learning to program*. Addison-Wesley.

Freeman, E., Freeman, E., & Sierra, K. (2014). *Head first design patterns: a brain-friendly guide*. O'Reilly.

Jachja, T., & Tiffany Jachja Tiffany Jachja is an Evangelist at Harness. Prior to joining Harness. (2021, May 4). *Understanding the Phases of the Software Development Life Cycle (SDLC)*. Harness. <https://harness.io/blog/devops/software-development-life-cycle/>.

Martin, R. C. (2010). *Clean code a handbook of agile software craftsmanship*. Upper Saddle River etc.: Prentice Hall.

Sanoulla, M. (2012, March 27). *Redundancy: An Open Enemy to Writing Good Code - DZone Java*. [dzone.com. https://dzone.com/articles/redundancy-open-enemy-writing](https://dzone.com/articles/redundancy-open-enemy-writing).

Tzu, S., & Giles, L. (2017). *The art of war*. Place of publication not identified: Greyhound Press.

Weisgard, ethan monnot. (n.d.). <http://www.aiki-shuren-dojo.com/pdf/Go%20no%20sen.pdf>. <http://www.aiki-shuren-dojo.com/>. <http://www.aiki-shuren-dojo.com/pdf/Go%20no%20sen.pdf>.

Wolfe, R. (2019, September). What's the Point? *Sword and Spirit*, 3-4.

What is Memory Leak? How can we avoid? GeeksforGeeks. (2017, July 14). <https://www.geeksforgeeks.org/what-is-memory-leak-how-can-we-avoid/>.

Charles Hudson joined Itten Dojo in 2012 but his training was interrupted by moves to Florida and Washington. Since returning he's become an indispensable member of the dojo and tested for *shodan* (first-degree black-belt) in May 2021. Mr. Hudson received a Bachelor of Science degree in computer science from Fort Hayes State University and works as a web developer with expertise in web design, development, and languages such as PHP, JavaScript, HTML, and CSS.

