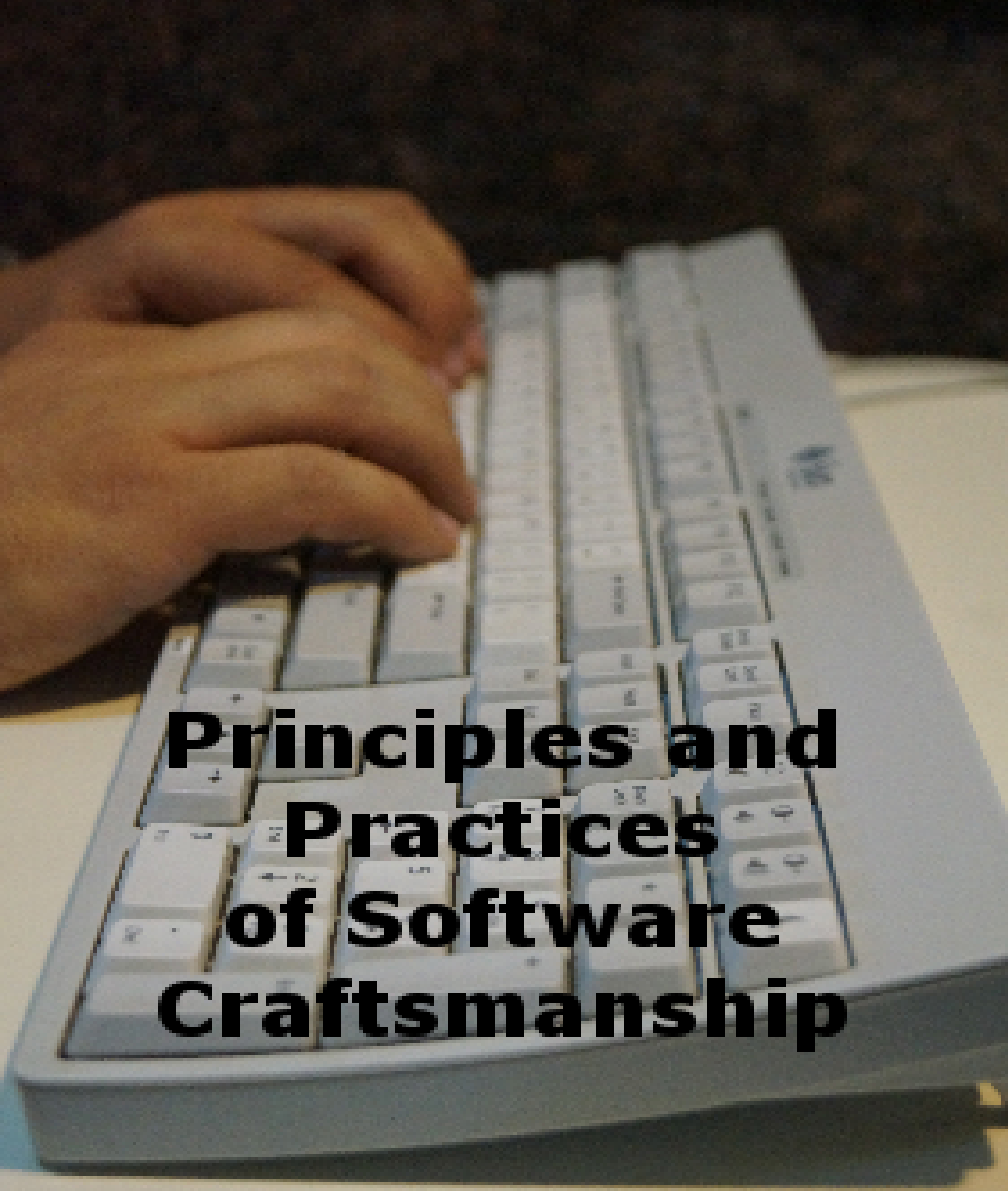


Principles and Practices of Software Craftsmanship

Michael Nash

Principles and Practices of Software Craftsmanship

Michael Nash

This book is for sale at http://leanpub.com/principlesandpractices

This version was published on 2014-11-21

Contents

Principle: Responsibility

> "The price of greatness is responsibility."
>
> *Winston Churchill*

I specifically chose to make the principle of *Responsibility* the first one we examine as it underlies many other principles shared by craftsmen.

Responsibility means being accountable for a thing or an act, being the person who will be called upon to take the credit or the blame.

In this context, we mean the responsibility to produce the *right* software at the *right* time. This is more than just producing what was specified according to the estimate – much more.

The right software means understanding what the customer wants, to the depth of having confidence that what we're building will in fact solve their problem, even if that means helping the customer discover the true need, as opposed to the expressed need.

The right time is more than delivering on schedule – it is more about delivering value sustainably at a predictable pace, not just aiming for a date on the calendar and declaring completion.

Where the Buck Stops

The craftsmen therefore takes *responsibility* for the whole process of delivering value. He doesn't hide behind excuses, and never says

- "The software does what you asked for" even when the understanding of the problem has evolved.

- "We'll have that for you by Friday" when he knows full well that is not possible.
- "We're done!" when he knows quality is below what his own professional ethics demands.
- "It works" when he knows there are not sufficient tests to be *sure* it works.
- "I know how" when he does not.

Holistic Responsibility

The craftsman understands that the scope of this responsibility is the whole of the deliverable *value*. This does not mean he is the only individual involved in delivery that value – indeed, there may well be a whole team, and each person in that team feels the same responsibility, it is not concentrated in any one person.

What it does mean though, is that the responsibility for value doesn't stop with the completion of development. If the product isn't delivered, deployed, configured, taught… the value hasn't occurred.

Customers don't pay for software – they pay for solving a problem, fulfilling a need, reducing a pain. If the need isn't fulfilled, it doesn't matter if the software does what the spec said it should, the value still hasn't happened.

Never Blocked

Responsibility is often seen in situations where a group of craftsmen are working on a project, and one of their number finishes their current task.

When he's ready, this person would normally pick up the next available task, but he sees that there are no additional tasks defined yet (perhaps the team hasn't planned out the next breakdown of stories yet).

The craftsman demonstrates responsibility in what he does next – he doesn't just hang out until somebody else finds what he's supposed to do next, he finds a task that will move the project forward, without conflicting with anyone else and without interrupting their work – and dives in.

The recently-coined term "DevOps" has been used to describe teams where the development responsibilities and the operations responsibility are contained in a single team, rather than different teams for each of these areas.

This is a natural way of working for the craftsman, as he sees it as a requirement to be able to deliver business value to have the full span of capabilities. Otherwise it becomes too easy to think something is the other guy's job.

Many craftsmen have been one-man teams at some point in their career, so being responsible for the whole of the delivery process is usually nothing new to them in any case.

Being able to always be contributing, always moving the project forward requires a commitment on each person's part to understand in detail the business value they are working towards. If you

only understand the bit you're working on, you can't easily fill in when that bit for some reason becomes unavailable – you'll flounder looking for something productive to do.

This is just the externalization of his understanding that it is the whole teams responsibility to deliver the functionality on time, on spec, and with a quality they can all be proud of. That's not his boss's responsibility, or some manager somewhere – it's *personal*.

When Things go Wrong

Responsibility doesn't mean that the craftsman will fall on his sword if things don't go as planned.

The responsible professional will ensure that all stake-holders know when a problem is happening (not long afterwards), and will describe what's being done to solve it.

The customer doesn't need someone to take the blame, he needs a *solution*, so when things go wrong the craftsman admits it (even, in fact, especially if it's a mistake he made), describes what he's going to do to fix it, and gets on with it.

It's when things don't go right that you see the difference between true responsibility and face-saving.

It's Never SEP

A final form of responsibility that the craftsman appreciates is the responsibility to initiate action when it's necessary. Sometimes circumstances conspire to make it easier to just go along for the ride, not to say anything when it's clear something should be changed or done differently. The craftsman doesn't take that easy way out.

The craftsman understands that it is never "someone elses' problem" (SEP), it's his. This is the essence of responsibility.

Steer your own Course

The craftsman also takes full responsibility for his own career.

It's not his employer's job to tell him what to learn, to buy him books or send him on training courses.

It's not someone elses job to tell him what the latest industry standards and best practices are, or that he should start using them on his project – these things are the craftsman 's responsibility alone.

If his employer wants to help or make suggestions, all well and good, but that doesn't mean the craftsman isn't steering the boat.

Tale from the Bilge

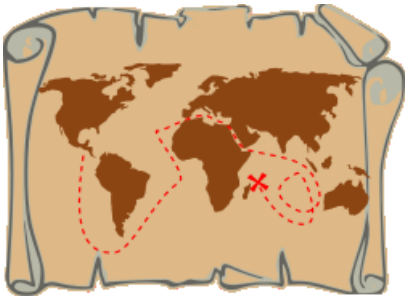
Responsibility is sometimes obvious, sometimes subtle. On a project a while back for a client in Texas, a team member - let's call him Dave - was tasked with investigating a new technology (OSGi, as it happens). He dug in over a period of a couple of weeks, in between his other development work, found out what he needed to, and brought his findings back to the team, with a recommendation that we give it a try.

His findings were communicated clearly, and the team agreed that OSGi was going to bring us significant benefit on this project, and we should give it a crack.

At a glance, you'd say that Dave's responsibility was fulfilled, right? He would not have agreed, and in fact, he didn't. Dave went on to build a sample project using OSGi, and to pair with another developer, Joe, on his own initiative and time to integrate a small service into the sample project. Joe was enthusiastic, and went back to the team with good things to say about OSGi and what it had done for their sample. Joe then paired with another team member, as did Dave, and pretty soon the whole project was smoothly integrated with OSGi.

Dave's task wasn't to go find out about OSGi and report back, even though that's what he was asked to do. What the team really wanted was the *value* that OSGi brought to our project, so that's what Dave *really* delivered by going the extra mile to build knowledge an experience about what he'd learned, and to share the enthusiasm he had for it.

That's responsibility.



- Figure out what the value to be delivered is, and take responsibility to deliver it.
- Consider that there's no such thing as "not your job" when it comes to delivering value.
- Learn about DevOps, and how it promotes a responsibility culture.