

The following is an excerpt from “Emergent Design: The Evolutionary Nature of Professional Software Development”, by Scott Bain, published by Addison Wesley/Pearson Education.

# Chapter 1

## *Software as a Profession*

In this chapter we will investigate what I hope is an interesting set of questions. Sometimes the real value of a question is not so much in finding a specific answer, but in allowing the investigation of the question to lead you to other, perhaps even more valuable questions. Also, sometimes discovering a question that is not being asked, or not asked very often, can help us to see opportunities that we are missing, which can also lead us to further, valuable discoveries.

I’ve been “in software” for a long time, but it occurs to me that we’ve reached a point in our industry when a little “stepping back” to see the nature of what we’re doing might be a useful thing.

---

### *How Long Have Human Beings Been Making Software?*

The answer to that question, like so many others, is "it depends."

What do we include in the notion of "making software?" Do we include the very early days when "programming" consisted of wire-wrapping PC boards and exchanging tubes? What about the Jacquard loom?

Maybe not... but then, should we include the days of "Data Processing": punch cards and mainframes and waiting overnight to see if your program ran properly, input-by-punch card or tape, output by printout, no interaction, no VDTs?

For my purposes here, I think we should start in the mid-to-late 70's, when small desktop systems originally emerged, and we began to develop interactive software for people to use directly.

It is not that I consider data processing to be less important / interesting / complex, it is just that the differences are so great between what they<sup>1</sup> did then and what we now do, that I don't see much wisdom carrying over. It is like riding a horse versus driving a car: they are both complex activities that require knowledge and skill, but knowing how to do one of them really does not help you to do the other well.

But wait! What about the emergence of object-oriented languages and techniques, as opposed to the procedural languages like C and Pascal that used to hold sway? Should the pre-OO, procedural era also be considered a separate kind of activity?

I don't think so. I think that, as it turns out, much of object orientation grew out of the maturation of procedural programming – that, in essence, the best practices we used to make procedural languages work well led to the concepts that got "built in" to object-oriented languages.

In fact, I think Object Orientation started as a "way of programming" before there was even an object-oriented language at all. There's also more to say about this, but for now, and in short, I think the object-oriented paradigm shares a lot of wisdom with the procedural one.

---

<sup>1</sup> Well, "we", I guess. When I got my start, I keyed my punch-cards on a keypunch machine that required you to file notches in a lead rod to change the tab stops. I don't miss those days at all...

I've got more to say on this (and will, in Chapter 4, "Evolution in Code, Stage 1"), but assuming we do or will agree, then I'll include the era of "structured programming" as part of our history.

That's my opinion, anyway. So, I would say we have been "making software" in this sense for 30-35 years.

Okay, another question.

---

### *What Sort of Activity Is Software Development?*

Is it a job, a craft, a profession, or what? Some would say it's an art or a science. Some would say it's engineering, some would say a branch of deductive logic. The real value, in my opinion, of asking this question is not so much finding the answer, but rather following it to the other questions which may arise by asking it.

We have lots of words for the things people do to make a living, and allow them to contribute what they are and know and can do to the world around them: job, work, trade, craft, avocation, métier, pursuit, art, career, profession, and so forth. Different people will use these words in different ways, but there are indicators, distinctions, expectations, and assumptions accompanying them that are fairly universal.

Take training, for instance. Most people would attach the concept of "extensive training required" to the notion of "profession", or "trade", and less so to "work" or "job."<sup>2</sup> This is not to say that all jobs are easy, but certainly there are rigors associated with entering a profession or trade that one does not expect to have to go through for a job.

---

<sup>2</sup> I am making a distinction here in using the term "job". Naturally, anything one does that involves work and produces value can be termed a "job", even if your job is being a doctor. I suppose what I mean here most people would call a "job-job", meaning a job you do because you need money, primarily, and really do not expect to be a long-term thing. I just always feel funny writing "job-job"...

Another distinction is licensing and oversight. Doctors, lawyers, accountants, contractors, and the like are licensed by the state, to ensure that they are competent, that their skills and training meet a minimal standard. This is generally because of the extensive damage they can do to us if they are not. This also, of course, implies that there is an agreed-upon set of minimal standards that these professionals can be held to, and that they are the right ones.

Furthermore, because of the rigors and standards expected of them, professionals (and craftspersons, too) have tended to form supportive organisms -- guilds and unions and professional organizations -- to support them and promote their success. These organizations also provide a focused place for shared wisdom and accumulated knowledge to persist and then to be handed down from one generation to the next.

There is an advantage, when one is engaging in complex and sometimes life-critical activities, to having this kind of support. It tells the professional if she is “up to snuff” on the critical aspects of her profession, and what to do if she is not. To those served by the professional, the advantage is also pretty clear: it is important to me that my doctor knows what he is doing, and that someone who can tell whether he does or not (better than I) is confirming this.

Also, professions tend to develop specific, highly specialized languages. Doctors, for instance, call removing your appendix an “appendectomy,” while removing your colon is a “colostomy.” Why is one an “ectomy” and the other an “ostomy”? I don’t know, but doctors do, and I’ll bet there is a lot to the distinction that is important to them. All we know is something of ours is leaving our bodies.

So, I’m going to somewhat arbitrarily settle on the word “profession” to indicate a vocation that requires extensive, lengthy training, has very specialized language to describe what the professionals do and to allow them to communicate with high fidelity from to one another, and usually has a professional organization supporting them. Professions have extensive traditions

behind them, well-defined communities of practice and support, and collected wisdom that is shared among all its members. A profession provides access to peer-review on activities and ideas, and a licensure or certification process, usually as part of an academic discipline or disciplines, to give confidence both to the professionals and those they serve.

Also, there is a very clearly-defined path to follow if you want to become a doctor, or a lawyer, or a master cabinetmaker. When society needs more of them, we know what process to take individuals through to increase their numbers, and we know how to tell young people to prepare for a given professional career.

So where does software development fit?

Software development is certainly very complex, and certainly requires extensive training and experience to do it well. Not everyone can do it: not because it is fundamentally beyond some people, but really because not everyone would *want* to do it, or would keep doing it for very long. It requires a certain temperament, a technical bent, and a fundamental love of problem-solving that does not exist in everyone.

Also, there are certainly supportive organizations and groups that have formed over time, and in these organizations and groups we software developers seek to share what we know, and how we do things. The entire open-source movement is, among other things, an example of collegiality among software developers. So are “Java User Groups” and “.Net Developer Study Groups.” Developers almost always support these efforts on their own time, because they feel them to be valuable.

So, I would say that software development is, by nature, a professional activity. Whether or not we have been conducting it as a profession, we should be doing so, in my opinion. I’m going to set aside the question of whether or not we should be regulated by the government, because my focus here is to discover those things that a “software profession” would provide that would be of

direct benefit to developers, and the quality of the software they produce (and, of course, therefore of benefit to those who own and use the software).

If you'll allow me this, then consider the following:

We have not been doing it terribly long. Most professions and crafts are hundreds or even thousands of years old, and so they have had a long time to work out the fundamentals that underlie them.

Imagine what medicine, as an example, was like when it was 35 years old. I imagine there was a lot of the attaching-of-leeches, and the shaking-of-rattles, and the letting-of-blood. Interestingly, medicine in that form *did* enjoy a certain degree of success. It turns out that bleeding a person can reduce a fever, though it is not a very good way of doing so (sometimes you end up reducing it *all the way*). It turns out, even today, that a fair percentage of the good a doctor can do for you has to do with your confidence in medicine, and your belief that the treatments you are being given will help to make you feel better.

So, witch-doctors did have some success. Just not very often, certainly not often enough, and the process was not terribly reliable, dependable, or predictable. Sound familiar?

Also, within this 35-year period that comprises our history, how much has changed, in terms of computing power, the relationship of business and daily life to software, and so forth? The human body is still pretty much what it was when medicine began, so doctors can lean very heavily on their past traditions and discoveries, while they learn new techniques and move their profession forward. We don't have that luxury. Computers are fundamentally different than they were even just a few years ago.

They are orders of magnitude faster, for example. Also, when I began writing software critical resources like disk storage and memory were not only very expensive, but also quite limited (there was no such thing as a 100 gigabyte hard drive, at any price). If you go back far enough (not that far, really), there was a time when the only individuals who interacted with

computers at all were trained, highly skilled operators. These are big changes, incremental changes yes, but change is the fundamental reality over the span of even one person's career.

So, what is the chance we are doing it completely right, this "making software" stuff? Pretty slim, I would say. In fact, I think it is rather amazing that we can do it at all, given that we are probably still at the leech-and-rattle stage, at least to some degree.

So, I am saying that I think software *should be* conducted as a professional activity, but that it *isn't yet*. Not quite.

---

## *What is Missing?*

Let's compare software development to what other professions typically have:

- ***Specialized language.*** In software development we have some of this, but it has always tended toward implementation specifics. Words like "loop", "switch", "break" and "exception" are specialized but very low-level. A master carpenter does not talk in terms of angles and inches when describing how you make a "mortise-and-tenon". The term itself captures all those specifics, and also communicates *why* you'd do it this way or that way, what you will likely run into in terms of *difficulties and opportunities*, and what you'll *gain or lose* as a result of choosing this path. Specialized language at this higher level helps professionals to make decisions better, and often to make them together, as colleagues.
  - Part of my thesis here is that the Design Patterns movement is, among other things, an attempt to add this specific, high-level language to what we do, to promote our level of professionalism. I don't say that this was necessarily the intention behind the movement but only what may have

turned out to be a significant, if unintended, contribution.

- I also think this may, in part, explain the popularity of patterns and pattern books. I think a lot of us have had the feeling that something is missing in what we do, and how we think and talk about it. In listening to this instinct, I am suggesting we are maturing toward professionalism.
- ***A clear path to entry.*** If someone asks you what they should do to "become" a software developer, what do you tell them? Go to college? Read a given set of books? Get a vendor-supported certification? Just get a job and fake it 'till you make it?
  - I can tell you, at least in general terms, what you should do if you want to become a lawyer. Go to college, then to law school, then intern at a law firm, pass the bar, become an associate, and work toward becoming a partner. That is to become a civil lawyer. Want to become a criminal lawyer? There are similar, but different steps to that goal, and they too, are well defined.
  - A colleague of mine, Adam Milligan, put it this way one day over lunch: "We have med school and law school... where is the dev school?" There isn't one. Hospitals hire residents in order to allow seasoned doctors to train them; they understand that medicine must work this way if they are to produce the new doctors we all need, at the level of quality that's essential for patient health and well-being. The software development business has not bought into this yet, and they need to.
- ***Peer-review.*** Doctors and lawyers have peer-reviewed journals and other mechanisms for support among professionals. These mechanisms allow the professionals to move the profession forward, to the benefit of all, and allow for a

reality check on new practices and procedures.

- We have some of these things in software development, but they are very *ad-hoc*, and not well organized. You can "Google a problem" and scan Usenet for answers, but your results will be spotty. Many of the user and study groups I mentioned earlier are, I think, grassroots attempts to create peer-review in software. The same certainly can be said about the extensive use of blogging to share ideas and wisdom among software; there is lots of great stuff out there, but it is very chaotic.
- ***Standards and practices.*** There are certain things a doctor simply knows to do, always, and that by doing so is guaranteeing for herself a certain level of success, or at least, by doing certain things she is avoiding a guarantee of failure. For instance, doctors know that cleanliness is very important. They sterilize instruments and wash their hands carefully before interacting with a patient.
  - Doctors did not always know this, by the way. Up until the mid to late 1800's, surgeons did not bother to even wash their hands before cutting into patients, and as a result their failure rate (dead patients) was much higher than it should have been. Hungarian-born doctor Ignaz Semmelweis, while working at a birth clinic in Vienna in 1860, suggested that there were these tiny, invisible, microscopic things making people sick, and that doctors should wash them off of their hands and instruments before treating patients.
  - At the time, he might as well have been blaming ghosts or evil spirits. He was laughed at, almost lost his license to practice three times, and finally had to intentionally infect himself to make his point. Only after Dr. Semmelweis' death was the germ theory of disease developed, and he

is now recognized as a pioneer of antiseptic policy and the prevention of infection.<sup>3</sup>

- *All* doctors know this now. They do not have to rediscover this basic truth of medical procedure, their profession "knows" this, and therefore so do they. They wash their hands before they even meet informally with a new patient.

In software development, we have had a general tendency to expect each developer to discover such things on their own, to make the same mistakes we have all made, re-solve the same problems, and build from the ground up. It is actually been part of the culture of "the cowboy coder" that developers would rather outdo each other than support each other.<sup>4</sup> This problem is often further exacerbated by the tendency in many organizations to move senior developers into management roles, rather than keeping them available to the junior folks. It seems as if the value of "those who know" is seen as secondary.

And it is not just what we know or don't know. It is the things we have learned to pay attention to that are relatively more important than other things: what you must pay attention to now and what you can safely ignore or worry about later.

A profession is an organism. There has been "medicine" for thousands of years, but no particular doctor has been around that long. The doctors practicing today gain support from this organism, and therefore can continue to benefit from the hard work and sacrifices of those like Dr. Simmelweis, even though he is long gone.

---

<sup>3</sup> This is not an isolated incident of being considered insane because of a keen insight. Marconi was institutionalized for believing you could transmit magic waves through the air that could transmit information.

<sup>4</sup> Actually, it's worse than this. In our "profession," there is little agreement about what is right. So we each learn our own way, and go our own way.

---

## *Who is Responsible?*

Generally speaking, one does not visit the doctor and say "Y'know Doc, I'm thinking I might like an appendectomy." You tell the doctor what's wrong, and the doctor tells you what the right thing to do is.

In software development, since we have not really thought of ourselves as professionals, neither have the people we develop software for been thinking of us this way. The stakeholders in a project will often set the timelines, impose the process, and track our progress.

If we were a profession, this would be up to us, or at least heavily influenced by us. We would be responsible for these decisions, and would be able to tell our customers what to expect, and when to expect it, with a high degree of reliability. The current process is like going to the doctor for your operation and telling the surgeon – “ya’ got an hour, I’ve got to be on the golf course by 11”.

This involves a shift, not only for us, but for them. This involves building trust, changing the basic nature of the relationship between clients and developers, and proving our value through an increased rate of success.

This is going to be *hard*, I think. But it has to happen.

Software development, in my view, is not only a practice that should be conducted as a profession, but could be considered the most important profession of all as we move into the 21<sup>st</sup> century.

Why?

All the other professions use software, after all, and are therefore vulnerable to its cost and quality. When you fly on an airplane, you are flying on software, more and more. When the doctor scans your abdomen for tumors and looks at the display from the ultrasound machine, she

is relying on software to tell her how to help you. When your identity gets stolen, that's somebody's software, somewhere, exposing you to risk.

And this does tie back, at least to some degree, to the question of regulation and certification. I am still formulating my opinion on what is appropriate for our profession here, but I have a concern. Over the coming years, as software penetrates more and more of the daily lives of ordinary people, failures of that software will become more and more a part of the social consciousness. If we continue to expose people to identity theft, medical and transportation accidents, viruses, Y2K type (or daylight savings) type bugs, etc... eventually the citizens are going to ask someone to do something about this.

They will turn to the government, of course. I don't want to hit this too hard or come off as an alarmist, but frankly if the US House of Representatives sets the standard for what constitutes professional software development, I don't think we're in for a good result. At all.

We have to do it ourselves, and I think we'd better get going. There is a lot to do, and I cannot claim to be doing it all here (I don't think I should, even if I thought I could. It should be a community-based, grassroots effort). But I hope this book will contribute some specific clarity on the key issues and encourage more of the same.

---

## *Uniqueness*

Let's be clear: I am not saying software development is like medicine or law or any other profession. Medicine is not like the law either. Doctors are not carpenters.

Professions are unique. They have derived their own unique processes that are highly appropriate to their nature. These processes come from a fundamental understanding of that nature, built incrementally over time by the communities themselves. Medicine is practiced the way it is because of the experiences of medical professionals over the history of the profession.

Part of the problem in the current practice of software development is that it is based, largely, on a fundamentally flawed principle: that it is very much like something else. That is what the next two chapters are all about.

## NET OBJECTIVES' LEAN-AGILE APPROACH

### INTEGRATED AND COHESIVE

All of our trainers, consultants, and coaches follow a consistent Lean-Agile approach to sustainable product development. By providing services at all of these levels, we provide you teams and management with a consistent message.

### PROCESS EXECUTION

Net Objectives helps you initiate Agile adoption across teams and management with process training and follow-on coaching to accelerate and ease the transition to Lean-Agile practices.

### SKILLS & COMPETENCIES

Both technical and process skills and competencies are essential for effective Agile software development. Net Objectives provides your teams with the knowledge and understanding required to build the right functionality in the right way to provide the greatest value and build a sustainable development environment.

### ENTERPRISE STRATEGIES

Enterprise Agility requires a perspective of software development that embraces Lean principles as well as Agile methodologies. Our experienced consultants can help you develop a realistic strategy to leverage the benefits of Agile development within your organization.



Contact Us:  
sales@netobjectives.com  
1-888-LEAN-244 (1-888-532-6244)

*We deliver unique solutions  
that lead to tangible improvements in software development  
for your business, organization and teams.*

# Services Overview

## TRAINING FOR AGILE DEVELOPERS AND MANAGERS

Net Objectives provides essential Lean-Agile technical and process training to organizations, teams and individuals through in-house course delivery worldwide, and public course offerings across the US.

## CURRICULA — CUSTOM COURSES AND PROGRAMS

Our Lean-Agile Core Curriculum provides the foundation for Agile Teams to succeed.

- ✓ Lean Software Development
- ✓ Agile Enterprise Release Planning
- ✓ Agile Estimation With User Stories
- ✓ Implementing Scrum for your Team
- ✓ Sustainable Test-Driven Development
- ✓ Design Patterns

In addition, we offer the most comprehensive technical and process training for Agile professionals in the industry as well as our own Certifications for Scrum Master and Product Champion.

## PROCESS AND TECHNICAL TEAM COACHING

Our coaches facilitate your teams with their experience and wisdom by providing guidance, direction and motivation to quickly put their newly acquired competencies to work. Coaching ensures immediate knowledge transfer while working on your problem domain.

## LEAN-AGILE ASSESSMENTS

Understand what Agility means to your organization and how best to implement your initiative by utilizing our Assessment services that include value mapping, strategic planning and execution. Our consultants will define an actionable plan that best fits your needs.

## LEAN-AGILE CONSULTING

Seasoned Lean-Agile consultants provide you with an outside view to see what structural and cultural changes need to be made in order to create an organization that fosters effective Agile development that best serves your business and deliver value to your customers.

# Free Information

## CONTACT US FOR A FREE CONSULTATION

Receive a free no-obligation consultation to discuss your needs, requirements and objectives. Learn about our courses, curricula, coaching and consulting services. We'll arrange a free consultation with instructors or consultants most qualified to answer all of your questions.

Call us toll free at 1-888-LEAN-244 (1-888-532-6244) or email [sales@netobjectives.com](mailto:sales@netobjectives.com)

## REGISTER FOR ACCESS TO PROFESSIONAL LEAN-AGILE RESOURCES

Visit our website and register for access to professional Lean-Agile resources for management and developers. Enjoy access to webinars, podcasts, blogs, whitepapers, articles and more to help you become more Agile. Register at: <http://www.netobjectives.com/user/register>