Shu Ha Ri

image: shu-ha-ri.gif

See also the video FAQ Agile FAQVid Shu Ha Ri (discussion: Re: Vid of Alistair describing Shu Ha Ri), and see how Kokoro (discussion: Re: Shu Ha Ri Kokoro) 心 extends Shu, Ha, Ri. (added, 2015)

I've written about the Shu Ha Ri progression several places, on Ward's wiki and in my books and talks. Here are a couple.

From *Agile Software Development* first edition, 2001:

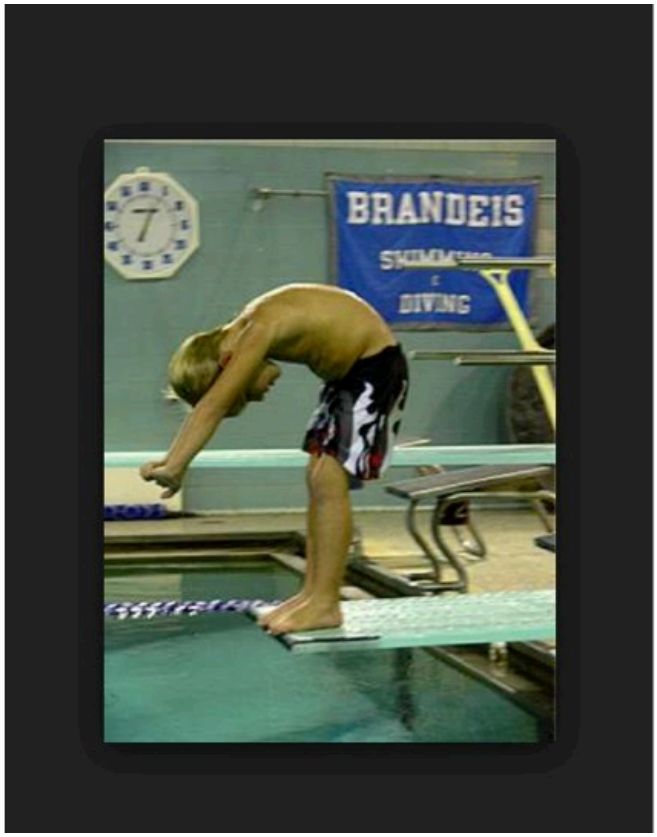
Three Levels of Listening

People who are learning and mastering new skills pass through three quite different stages of behavior: *following*, *detaching*, and *fluent*.

People in the *following* stage look for one procedure that works. Even if ten procedures could work, they can't learn ten at once. They need one to learn first, one that works. They copy it; they learn it. In this stage, practitioners measure success by (a) whether the procedure works and (b) how well they can carry out the procedure.

> **The 1708 Card Reader**
> We watched a Humanities major encountering the Univac 1708 card readers for the first time in her first programming class (this was 1974).
> Her short program didn't compile. Upset at this failure, she requested help from the student assistant. When the program failed to compile a second time, she became nearly hysterical, and shouted at the assistant in tears:
> "But you *promised* me it would work!"

Her reaction is typical of stage one learning. The reward for success in this first stage is the sense of, "at least this thing works," and "I can at least manage to accomplish that."



守

People moving to some new skill domain, whether software or some other, want explicit instructions. In terms of written software development methodologies, this means a thick, detailed manual. The thickness and the detail offer signs of safety for the learning.

In the *detaching*, or Level 2, stage, people locate the limitations of the single procedure and look for rules about when the procedure breaks down. They are actually in the first stage of a new learning; namely, learning the limits of the procedure. The person in the detaching stage learns to adapt the procedure to varying circumstances. She is now more interested in learning the ten alternative procedures, in learning when each is most applicable and when each breaks down.

A large-scale technique breakdown of this sort occurred in our industry when large software contracting firms, finely tuned to developing software using Information Engineering (IE) architectures, had to begin delivering object-oriented software. After years of unsuccessfully trying to adapt IE methods, they had to develop completely new development methodologies, often regressing through quite unstructured development before discovering new structures to support the new projects. Most of these organizations now have two methodologies, one for IE and another for object-oriented (OO) development.

In the third, *fluent* stage, it becomes irrelevant to the practitioner whether she is following any particular technique or not. Her knowledge has become integrated throughout a thousand thoughts and actions. Ask her if she is following a particular procedure, and she is likely to shrug her shoulders: It doesn't matter to her whether she is following a procedure, improvising around one, or making up a new one. She understands the desired end effect and simply makes her way to that end.

A team leader who has led a number of projects in different areas doesn't care about "methodology" any more: "Just leave us alone and we'll deliver it," she says. She simply observes and senses that more discipline is needed here, more freedom needed there, more communication needed in some other place. This is the Level 3 practitioner.

The Three Levels and Methodologies

The same three levels apply to listening, coaching, or reading about software development. It is important to respect all three levels, as the following story illustrates:

> **Level Mix-up with CRC Cards**
> Three of us, unaware of these levels of learning, accidentally crossed to the wrong level on our first design mentoring assignment. We decided to lead small design sessions using Class-Responsibility-Collaborator (CRC) cards. (See Beck, 1987.)
> The three of us worked slightly differently, which upset the designers, who were newcomers to object-oriented design. They said,
> "You are all doing something different! Which one of you is right, and why don't the others do that, too!"
> We tried saying, "It doesn't matter. They all work." But that did not help the beginners, who were confused: Should they hold the cards up or leave them on the table? Should they write down all the instance variables, or some, or none? And so on.
> We knew that the session could be made to work using any of those variants, but the beginners were still in Level 1 and needed one defined way of working that they could apply several times in a row.

A programming book aimed at the Level 1 audience would work to assure the reader that there really is a way of developing software that works, and that if the reader will just follow it, success will follow. Such a book might be called *The Science of Programming* (Gries 1983) or *The Discipline of Programming* (Humphreys 1991).

A methodology text aimed at the Level 1 audience describes processes, techniques, and standards in detail. The very detailed templates in the

Rational Unified Process (RUP) serve Level 1 practitioners. The big methodologies of Andersen Consulting, Ernst & Young, and the like fall into this category.

A programming book aimed at the Level 2 audience might be called *The Art of Programming* (Knuth 1997). It would show the reader several techniques for working, with examples and notes about when each is more useful.

A book aimed at combined Level 2 and Level 3 audiences might be called *The Laissez-Faire of Programming* (think of that as an alternate title for this book) or *The Pragmatic Programmer* (Hunt 2000). It would name issues to bear in mind and identify techniques that the practitioner might learn, pick up, and put down as needed. The expert will find it a useful library of ideas, but the beginner finds it lacking specific rules.

The Level 3 listener knows that all the published software development techniques are personal and somewhat arbitrary. Discussions among Level 3 people sound distressingly Zen:

“Do whatever works.”

“When you are really doing it, you are unaware that you are doing it.”

“Use a technique so long as it is doing some good.”

To someone at the fluent level of behavior, this is all true. To someone still detaching, it is confusing. To someone looking for a procedure to follow, it is useless.

My book, *Writing Effective Use Cases* (Cockburn 2001), is a technique book with different information for readers at the three levels.

For practitioners at the first level in use case writing, it details the minutiae of use case writing. It provides them with specific procedures to follow. For practitioners at the second level, it contains rules and tips for varying the basic rules. The book does not try to teach anything specific to the Level 3 reader, who will, in any case, find something new in it to try out one day. Instead, it assures the Level 3 reader that the rules are not binding, that a lot of different ways of working can be effective, and that the people at Levels 1 and 2 are being told this, too.

To the extent that book is successful, it permits the Level 1 reader to get specific advice, the Level 2 reader to learn the boundaries of the rules, and the Level 3 reader to move with freedom.

One member in the *Crystal* family of methodologies is *Crystal Clear*. *Crystal Clear* can be described to a Level 3 listener in the following words:

> “Put 4-6 people in a room with workstations and whiteboards and access to the users. Have them deliver running, tested software to the users every one or two months, and otherwise leave them alone.”

I did, in fact, describe *Crystal Clear* in those words to a savvy project sponsor. He followed those instructions and reported five months later, “We did what you said, and it worked!”

I interviewed the team leader some months later and his report was about as short as my instructions:

“Following your suggestion, the four of us took over this conference room, which has network connections. We kept it for all four months, drawing on the whiteboards over there, delivering software as we went. It worked great.”

If you are an experienced software developer and can apply those instructions, then you have no need for an entire book called *Crystal Clear*. If either you or your sponsor is not at that stage, then you need the

book-length version. This version describes key techniques in detail, exposes the principles involved, considers the zone of applicability for this minimalist methodology, and says how to move out of *Crystal Clear* when the project moves out of the zone of applicability.

One lesson to take away from all this is that if you are reading methodology texts at Level 1, don't become depressed that there are so many techniques and principles to master. Wishing the world were so simple as to only need a single software development technique is a wasted wish. Hire someone who is at Level 2 or 3.

If you read methodology texts at Level 2, note the alternative techniques and look for places to vary them.

If you are reading methodology texts at Level 3, recognize the continued need for methodology definition at Level 1. There will always be people entering the field who will need explicit direction at first, even if you don't.

Kent Beck, author of *Extreme Programming Explained*, described the use of Extreme Programming (XP) using similar levels. Asked about XP and the five levels of the Software Engineering Institute's "Capability Maturity Model," he replied with XP's three levels of maturity:

1. Do everything as written.

2. After having done that, experiment with variations in the rules.

3. Eventually, don't care if you are doing XP or not.

The Three Levels and This Book

As described in the Preface, this book is aimed mostly at Level 2 and 3 readers. It has little to offer a Level 1 software practitioner looking for a simple procedure to follow. In fact, a key point of the book is that all methodologies have limitations, areas where they are more or less applicable. It is not possible to name one best and correct way to develop software. Ideally, the book helps you reach that understanding and leads you to constructive ideas about how to deal with this real-world situation. In that sense, the book is aimed at moving some Level 2 readers to Level 3.

Topics for the Level 2 readers include heuristics for selecting a project's base methodology and the ideas behind agile methodologies.

If you are a Level 3 reader, I hope you will find words to help express what you already know.

A few topics in this book are likely to be new even to experienced developers. Most people are Level 1 readers when it comes to the vocabulary for describing methodologies and just-in-time methodology tuning. These are therefore written in more detail.

Shu-Ha-Ri

The three levels of practice are known in other skill areas. In Aikido, they are called *shu*, *ha*, and *ri* (roughly translating as *learn*, *detach*, and *transcend*). To look up information about shu-ha-ri, you might start with a Web search or at www.aikidofaq.com/essays/tin/shuhari.html. The following extract is from that site's *The Iaido Newsletter*, Volume 7, Number 2, #54, Feb. 1995, "Shu Ha Ri" by Ron Fox, MWKF. (In this extract, the references in square brackets refer to references Ron Fox provides inside his article.) I find it fascinating how his portrayal so accurately predicts our mistaken, early attempt to teach design using CRC cards.

> "Shu, or Mamoru means to keep, protect, keep or maintain [1]. During the Shu phase, the student builds the technical foundation of the art. Shu also implies a loyalty or persistence in a single ryu or, in the modern interpretation, a single instructor [2]. In Shu, the student should be working to copy the techniques as taught without modification and without yet attempting to

make any effort to understand the rationale of the techniques of the school/teacher [3]. In this way, a lasting technical foundation is built on which the deeper understanding of the art can be based.
The point of Shu is that a sound technical foundation can be built most efficiently by following only a single route to that goal. Mixing in other schools, prior to an understanding of what you're really up to is an invitation to go down a wrong path. A path where the techniques developed will not have sound theoretical or practical value. In the traditional interpretation of the Shu stage, it is the instructor that decides when the student moves on from Shu to Ha, not the student. It's up to the student to follow the instructor's teaching as an empty vessel to be filled up [1].
Ha, is the second stage of the process. Ha means to detach and means that the student breaks free from the traditions of the ryu to some extent [2]. In the Ha stage, the student must reflect on the meaning and purpose of everything that s/he has learned and thus come to a deeper understanding of the art than pure repetitive practice can allow. At this stage, since each technique is thoroughly learned and absorbed into the muscle memory, the student is prepared to reason about the background behind these techniques [3]. In academics, the Ha stage can be likened to the stage where enough basic information is available to the student that research papers of a survey nature could be expected.
Ri means to go beyond or transcend. In this stage, the student is no longer a student in the normal sense, but a practitioner. The practitioner must think originally and develop from background knowledge original thoughts about the art and test them against the reality of his or her background knowledge and conclusions as well as the demands of everyday life. In the Ri stage, the art truly becomes the practitioner's own and to some extent his or her own creation. This stage is similar in academia to the Ph.D. or beyond stage.
[1] Kuroda, Ichitaro, "Shu-Ha-Ri" in *Sempo Spring,* pp. 9-10, 1994.
[2] McCarthy, Patrick, "The World within Karate & Kinjo Hiroshi" in *Journal of Asian Martial Arts,* V. 3 No. 2, 1994.
[3] Private conversations with Nakamura, L. Sensei Toronto. Spring, 1994."

(**Note:** Ron Fox's articles might still be online, check http://www.kendo-world.com/wordpress/?p=1822 and http://www.kendo-world.com/wordpress/?p=1824)

From *Agile Software Development 2nd edition* 2006:

The Shu-Ha-Ri distinction dates back, as I learned, not to the origins of Aikido the early 1900s, but to Japanese Noh theater, almost four centuries ago.

Understanding the three levels of listening has turned out to be of greater value than I expected. It was originally placed as a small section in an appendix, but so many people wrote to me during the early drafts of the book, describing how they were using the Shu-Ha-Ri distinction to smooth meetings, that I moved it forward into the Introduction.

You can apply Shu-Ha-Ri to designing courses and writing technique materials. Recognize that some people come to the course in the Shu state, looking for the technique that solves their problems. Other people come already knowing many techniques, afraid that you will try to convince them that your technique is the technique that solves their problems.

Organize your material to reassure the advanced people in your course that it is not being presented as a cure-all, but simply as another technique in the toolbox of the professional. Then teach the technique at the Shu level so that the beginners will have a starter technique to take home with them. But before you finish, put the technique into a larger context so that the beginners can start to see where they will expand to and so that the advanced people can see how this technique bridges to others they might already know or want to learn.

I got caught in a Shu-Ha-Ri mismatch shortly after publishing the first edition and was saved by someone who knew the trio.

> In 2002, Jim Highsmith and I were organizing the first Agile Development Conference (alert readers will notice here that conference organization is a cooperative game of invention and communication, but one in which the incremental / iterative and collocation development strategies are difficult to use, if not impossible). It was my first-ever conference-organizing experience, and I was very much in the Shu stage but without anyone to follow. Accordingly, I was incredibly diligent in listing everything that needed to be taken care of for the conference. I had sheets and sheets of paper containing all manner of minutiae.
>
> We signed an experienced conference organizer, who was baffled by my excruciatingly long lists. A colleague watching our tense conversations pointed out that my long lists were immaterial to her because she operated in Ri mode and could make the necessary plans in her head rather than on slips of paper.
>
> That observation gave me the peace of mind I needed to relax and let her take over. I still kept lists of details to watch for, but only the ones I was specifically concerned about.

You can apply the lessons of Shu-Ha-Ri in your own environment:

- Look for level mismatches at work. Introduce the Shu-Ha-Ri concept and see whether that might help to smooth the discussion.
- Update your training materials to highlight the sections that are for Shu-level learning and where the Ha and Ri levels fit.