BTI360 teammate Joel Goldberg recently retired after working in the software industry for over four decades. When he left he shared with our team some of the lessons he learned over his career. With his permission, we reshare his wisdom here.

Looking back on four decades in the software industry, I'm struck by how much has changed. I started my career with punch cards and I am ending in the era of cloud computing. Despite all this change, many principles that have helped me throughout my career haven't changed and continue to be relevant. As I step away from the keyboard, I want to share six ideas I've learned from my career as a software engineer.

1. Beware of the Curse of Knowledge

When you know something it is almost impossible to imagine what it is like *not* to know that thing. This is the curse of knowledge, and it is the root of countless misunderstandings and inefficiencies. Smart people who are comfortable with complexity can be especially prone to it!

If you don't guard against the curse of knowledge it has the potential to obfuscate all forms of communication, including code. The more specialized your work, the greater the risk that you will communicate in ways that are incomprehensible to the uninitiated. Fight the curse of knowledge. Work to understand your audience. Try to imagine what it would be like to learn what you are communicating for the first time.

2. Focus on the Fundamentals

Technology constantly changes, but some fundamental approaches to software development transcend these trends. Here are six fundamentals that will continue to be relevant for a long time.

- **Teamwork** — Great teams build great software. Don't take teamwork for granted.
- **Trust** — Teams move at the speed of trust. Be the kind of dependable person you would want to work with.
- **Communication** — Communicate honestly and proactively. Avoid the curse of knowledge.
- **Seek Consensus** — Take the time to bring your whole team along. Let discussion and disagreement bring you to the best solution.
- **Automated Testing** — Well-tested code allows your team to move fast with confidence.
- **Clean, understandable, and navigable code and design** — Think of the next engineer that will take over your code as your customer. Build code that your successor won't have any trouble reading, maintaining, and updating.

3. Simplicity

Fighting complexity is a never-ending cause. Solutions should be as simple as possible. Assume the next person to maintain your code won't be as smart as you. When you can use fewer technologies, do so.

“A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.”

Antoine de Saint-Exupery

4. Seek First to Understand

One of Stephen Covey’s seven habits is, “Seek First To Understand, Then To Be Understood.” This maxim has helped me more than any other advice to become a good listener and teammate. If you want to influence and work effectively with others, you first need to understand them. Actively listen to understand their feelings, ideas, and point of view before you begin trying to make your own thoughts known.

5. Beware of Lock-In

There will always be the next hot productivity product that will promise to revolutionize how software is built. Computer Assisted Software Engineering (CASE) tools, COTS, Enterprise Resource Planning products like Peoplesoft and SAP and, yes, even Ruby. They claim amazing reductions in cost and time if you buy into their holistic development philosophy. What is not always as obvious is the significant up-front costs or the constraints you may be committing yourself to. Lock-in used to primarily happen with vendors, but now it can happen with frameworks too. Either way, lock-in means significant cost to change. Choose wisely. New is not always better!

6. Be Honest and Acknowledge When You Don't Fit the Role

At some point in your career you may find yourself in a role that isn't a good fit. A bad fit isn't a character flaw, but it's a problem you shouldn't ignore. There may be more than one solution to such a dilemma: you can evolve or the role can evolve. The key is to have the self-knowledge to recognize what is happening and get yourself out of an unhealthy spot. Being unhappy is in no-one's best interests, and BTI360 recognizes this.

When I was at GM, you were a failure if your next move was not *up*—managing more people or taking on bigger, more complex projects. For many, this made for a miserable career path (see the Peter Principle). At EDS, the culture wasn't like this. People moved in and out of management roles. There was no stigma associated with moving from roles with greater scope, like strategic planner, to roles with more narrow scope, like PM or project-level developer. I was one of the people who took advantage of this flexibility, moving from a role at the top of the technical pyramid back to being a project-level developer. I never looked back.

Final Thoughts

Even before I joined BTI360 I knew enough about the culture to know that it was a place that valued the kinds of principles I've described above. I hope each of you will take ownership of maintaining a strong engineering culture that will continue to make BTI360 a great place to build software.