

## By Jonathan Lurie

With input from the TechRepublic Community

### Takeaway

What does it take to be a top-notch developer? Here are 11 critical success factors for taking your skills and your career to the next level. Five factors are taken from an article originally published on Builder.com on July 15, 2002. The other six factors were submitted by the TechRepublic Community.

---

### Table of Contents

<b>SUCCESS FACTORS.....</b>	<b>2</b>
JONATHAN'S FACTORS.....	2
<i>Constantly up your IT IQ.....</i>	2
<i>Adopt a reuse mindset.....</i>	2
<i>Understand underlying business principles.....</i>	2
<i>Have a mentor, be a mentor.....</i>	2
<i>Be your own worst critic.....</i>	2
<b>USERS .....</b>	<b>3</b>
[SHELLYDOLL] .....	3
<i>Flexibility.....</i>	3
[DISRUPTED].....	3
<i>Practice makes perfect.....</i>	3
<i>Real-world experience.....</i>	3
<i>Look deeper, grasshopper.....</i>	3
[UBB].....	3
<i>Communication skills.....</i>	3
[EVEREVOLVING].....	3
<i>Empathy and emotional intelligence.....</i>	3
[BBEASLEY] .....	4
<b>ADDITIONAL RESOURCES .....</b>	<b>5</b>
<i>Version history.....</i>	5
<i>Tell us what you think.....</i>	5

## **Success factors**

A friend of mine in his second year at the university asked me what it takes to be a great programmer. He had built a few Web sites and some simple VB6 applications and was interested in pursuing "something in computer science" as a career. My first reaction was to ramble off a quick list of the usual must-haves: C++/Java, networking, database design, and so on.

But a couple days later, I started reconsidering my advice. My first round of suggestions was merely "the basics," and knowing the basics is essential to any developer, top-notch or otherwise. So I came up with what I think are the critical success factors for truly outstanding software engineers.

### **Jonathan's factors**

#### **Constantly up your IT IQ**

Remember that no matter how much natural ability you have, unless your IT IQ is up to par then you will never realize your potential. Choose skills updates carefully. For example, while it's important to have a well-rounded IT education, every developer doesn't need an MCSE 2000. Join a user group or take a class to keep on top of emerging technologies, such as XML or .NET.

#### **Adopt a reuse mindset**

Top developers have both a firm understanding of Object-Oriented Programming (OOP) and a strategic approach to solving development problems. An effective programmer can separate root problems from their symptoms; complex problems should be broken down to individual areas. The best programmers break problems down step-by-step. This approach is conducive to individual code reuse and sharing vital resources with peers. The same problem should never be solved twice.

Develop a strong grasp of OOP principles, such as inheritance. Inheritance, which allows subclasses to utilize already defined classes, is a cornerstone of software reuse. You should be up to speed with all the latest theories on software development.

#### **Understand underlying business principles**

A good programmer recognizes that no solution is effective unless it meets business objectives. Budgets, resources, and schedules play an integral role in the development of the solution. If a solution is not viable when weighed against these constraints, the solution must be revisited until it is viable. Selectively use the mantra "Cheaper, faster, better"—in the real world, you'll end up picking two and moving on.

#### **Have a mentor, be a mentor**

The role of a mentor is integral in a top programmer's development. A mentor motivates, directs, commends, and chastises the programmer throughout his career. Think about a boxer who goes back to his corner between rounds. The boxer is counseled, healed, and motivated by his team. This team acts as a mentor to the boxer. A good developer has a network of mentors. While you should have mentors within your organization, it's often more important to have mentors outside of your current company.

#### **Be your own worst critic**

The idea that you must divest yourself of your ego is absolutely preposterous. Ego is not a problem, egotism is. To many, this may seem like splitting hairs, but the difference is as great as the difference between marginality and excellence. Ego drives us to be excellent; the egoless are content with being marginal.

Have you ever gone to someone for candid criticism, only to repudiate the feedback as it's given? If so, then go back and ask for more candid feedback. You'll find that the criticism is less candid. If you ask for candor, don't argue with it. Contemplate it and thank the person for the help.

It's always easier to accept criticism from others when you are your own worst critic; you'll find that others' criticisms are less severe than your own. If you are able to criticize your own work, you'll find others more receptive to helping you find ways to improve.

## **Users**

When this download was originally published, it inspired several members of the TechRepublic/Builder community to respond with some traits of their own to add to the list.

### **[shellydoll]**

#### **Flexibility**

You don't always get to code things the way you'd like - you have to be flexible in your mindset if you're going to adopt a given architecture and be a part of the team.

### **[Disrupted]**

#### **Practice makes perfect**

Reading and keeping up to date about the technologies you're using is always beneficial, but reading about them and then applying them, seems to be the best way to fully grasp the intricacies.

#### **Real-world experience**

As a developer and analyst within an enterprise, I get inundated with specifications details that I would normally not consider on my own small programming projects at home. Relating this work experience to my own personal projects has helped me to take different approaches to even the smallest development projects.

#### **Look deeper, grasshopper**

When presented with problem areas within projects, the experience gained with solving these problems is very valuable, but looking deeper into the root cause will help a developer/analyst to never forget how or why the solutions to these problems were reached.

### **[UBB]**

#### **Communication skills**

A developer having good communication skills can interact with his team as well as his end user in a more effective way.

### **[EverEvolving]**

#### **Empathy and emotional intelligence**

Put yourself in the shoes of the customer and your team member. This will really help in extracting the potential out of an employee when he/she him/herself doesn't know it or recognize it. Also it helps to pick out enthusiasts among the team who wait for their chances to come. Above all it helps a so called 'trouble some' or 'under productive' team member to realize his stuff and do well. It's always easy for a leader to fire his non performing staff. But if he/she takes the pain in realizing or understanding what's wrong with a person, then I really believe he can work out wonders with that person, its just mutually beneficial. The company as such may be gaining a lot out of the skills that are otherwise hidden and gone waste.

A leader in general, should not be totally business oriented, but also humanitarian. It will help him/her to work miracles out and something more than what was actually expected.

## [bbeasley]

**Editor's note:** While bbeasley did not add additional items to the list of factors, he/she did take time to elaborate on the importance of each factor in one's professional development.

I like these points and some others because they focus on what it means to be a professional.

- 1) Constantly increasing your IT knowledge is always important and it provides many benefits. The IT arena changes dramatically over a short period of time. You will start to anticipate change, or better yet, when not to change.
- 2) Having a reuse mindset goes along with keeping the full development cycle in your mind and looking further down the road than the piece of code you are currently looking at. You must consider deployment and support of applications, how to make it easier on you and make it easier on your users. Part of OOP is making components that are flexible and adapting to change. Requirements change and anticipating this will allow you to develop more flexible components.
- 3) You have got to understand your customer or you are going to try and meet their needs. Time and time again I have seen developers (including myself) code something they thought was perfect only to have the customer not use it. Realize what the customer wants and make sure you understand where they are coming from.
- 4) Nothing makes sure you understand something than trying to teach it to someone else. And the reverse is don't try and figure everything out yourself. Learn from others all the time.
- 5) Learn your weaknesses and anticipate them. Everyone does some things better than others. If you learn from good mentors hopefully you will learn how to critique yourself.
- 6) I like the principal of being flexible. Things change when you don't want them to. Constraints are sometimes out of your control. Adapt to the situation, make the most of it and move on.
- 7) Look at the big picture all the time. Am I meeting the needs of the customer? Am I bringing value to my employer/customer? How can I work smarter? Is there a tool I could create to help me do this easier? Am I having fun?

## Additional resources

- Sign up for our [TechRepublic NetNote](#), delivered on Mondays, Wednesdays, and Thursdays.
- Check out all of [TechRepublic's newsletter offerings](#).
- [Adopt these five critical success factors to be a top-notch developer](#) (Original Article)
- [6 steps to better software documentation](#) (Download)
- [Integrated SCM for Rational Developer Products and Eclipse](#) (White paper)

## Version history

**Version:** 1.0

**Published:** June 17, 2005

## Tell us what you think

TechRepublic downloads are designed to help you get your job done as painlessly and effectively as possible. Because we're continually looking for ways to improve the usefulness of these tools, we need your feedback. Please take a minute to [drop us a line](#) and tell us how well this download worked for you and offer your suggestions for improvement.

Thanks!

—The TechRepublic Downloads Team