Breaking Down Software Development Roles

Sponsored By

IBM

an internet.com Developer eBook

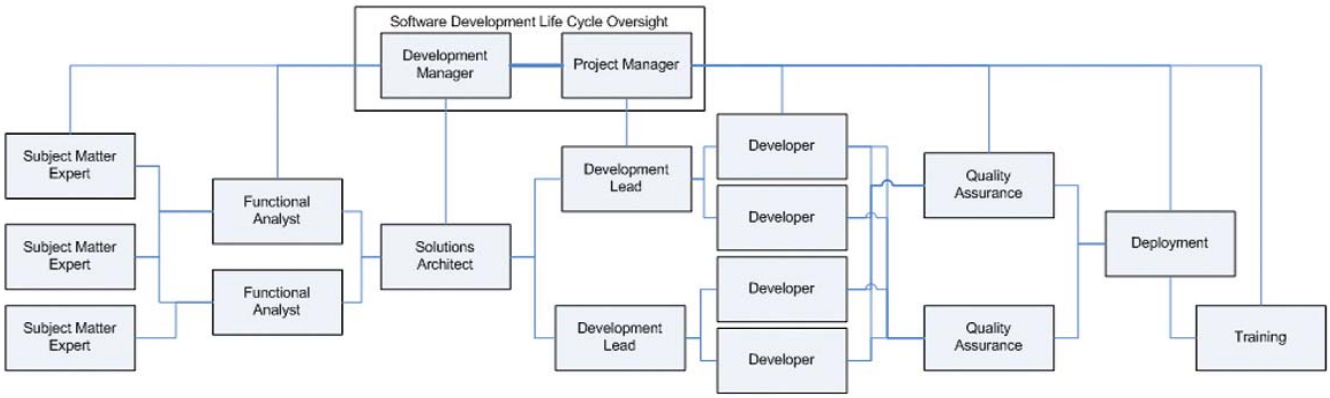
Software development is done differently at every organization, and in every home office throughout the world. The process that one organization or person uses to develop software may work for their specific environment and situation, but may fail miserably in another set of circumstances.

It is, in part, the differences in environment that make it so difficult to quantify the process of software development in a single set of terms that all practitioners can agree on. As newer approaches appear on the scene, the perspective of the world on what the process should look like changes slightly or dramatically.

However, despite these changes there are some things that remain the same. There will always be a need to understand the business problem, convert that problem into an architecture, convert the architecture into a solution, test the solution, and deploy the solution. Although each of these processes may change to some extent based on the programming models and tools being used, fundamentally there are some roles, which every process has in one form or another. One person may be filling all the roles or a handful of the roles, or one very specific role. Despite this there is a need for all of the roles -- each serves a purpose. The organization chart below gives you an idea of how each position fits together within an organization.

Common Roles

There is a series of roles that exist in most software development processes. As mentioned above, one team member may be filling many roles and some roles may be suppressed for a specific type of project, but all of these roles exist in one form or another in every software development project:



Subject Matter Experts (SME)

The Subject Matter Expert is the person or persons from which requirements are captured. These are the people who know what the software needs to do and how the process works. The SME role is somewhat different from the other roles because it is constantly changing as new clients (internal or external) are brought in to help design a solution. SMEs are rarely from IT -- except when the solution is being designed to support IT. SMEs are most frequently the person who will receive the benefit of the system.

Functional Analysts (FA)

Functional Analysts have the unenviable task of eliciting clear, concise, non-conflicting requirements from the Subject Matter Experts who may or may not understand how technology can be used to transform the business processes in a positive way.

Solutions Architect (SA)

The technical architect is responsible for transforming the requirements created by the Functional Analysts into a set

of architecture and design documents that can be used by the rest of the team to actually create the solution. The Solutions Architect is typically responsible for matching technologies to the problem being solved.

Development Lead (DL)

The Development Lead's role is focused on providing more detail to the Solution Architect's architecture. This would include detailed program specifications creation. The Development Lead is also the first line of support for the developers who need help understanding a concept or working through a particularly thorny issue.

Developer (Dev)

The heart and soul of the process, the developer actually writes the code that the Development Leads provided specifications for.

Quality Assurance (QA)

The Quality Assurance role is an often-thankless position that is designed to find bugs before they find their way to the end customers. Using a variety of techniques ranging from keying in data and playing with the system to formalized, automated testing scripts, the Quality Assurance team is responsible for ensuring the quality of the solution and it's fit to the requirements gathered by the Functional Analyst. Sometimes the QA team is known by their less flattering name of testers.

Deployment (Deploy)

The Deployment role is the one that packages up all of the compiled code and configuration files and deploys it through the appropriate environments or on the appropriate systems. The Deployment role is focused on getting the solution used. To that end, the role may include automated software installation procedures or may be as simple as copying the files to the appropriate place and running them.

Training

The Training role is responsible for documentation for the system as well as any instructor or computer-based training solutions that are designed to help the users better understand how the system works and what they can do with it.

Project Manager (PM)

The Project Manager is responsible for ensuring consistent reporting, risk mitigation, timeline, and cost control. The project manager role is a problem-solver role. They try to resolve problems while they are small so that they can be handled more quickly and with less cost.

Development Manager (DM)

The Development Manager is responsible for managing multiple priorities of conflicting projects. The Development Manager role is also an escalation for issues from the team, which it is unable to resolve internally.

Of course, each organization has its own take on these roles; however, these are the roles you'll see most often in an organization doing development.

Critical Skills for Every Role

As we examine each of the roles in detail, we'll include details that are critical to the success of the role. We've also identified a common set of business skills essential to each role. The common skills to all roles are:

Understanding Business

Although some roles are focused very specifically around certain aspects of understanding and converting business

requirements, every role in the process should have awareness and sensitivity to the business processes and needs that require technology in the first place. Without this, technology may be implemented but it may not solve the real needs, and will therefore be considered a failure.

Broad Understanding

Although an understanding of software development is critical there are other areas where an understanding can be invaluable. For instance, understanding how computers work internally including memory, cache, hard drives, etc., can help you learn how to more appropriately conserve those resources. Similarly, understanding networking can help in the development of applications that are compatible or even friendly to the networks that they're working across. SMEs broad understanding of the industry can be invaluable in terms of creating solutions that fit both the organization and the industry. The QA team can benefit the project with a broad understanding by minimizing QA costs while improving testing coverage. In short, a broad understanding can help every role.

Multiple Perspectives

The ability to approach solutions from multiple perspectives is critical to software development. Understanding how each person who is working on a problem views an issue, or how different customers will view the solution, is important to be able to find the best solution based on all of the information. There are always multiple ways of viewing -- and solving -- a problem. The trick is to find the best one from the list of possible options. The larger the list of options (perspectives) the better the solution.

People Skills

Also known as soft skills, the ability to interact with other people and to be a part of a team is essential to nearly every role in a software development project. The lower the overall people skills of the team, the higher the likelihood that the project will end in some explosion.

Lifelong Learning

Although some might argue that the perspective of being a life-long learner is more of an attitude than a skill, it is a critical part of being in a high-change industry, like IT in general and software development specifically. What is learned today will be obsolete tomorrow. The only way to stay ahead of the game is to approach life from the perspective of continuous learning. Each new experience is a new opportunity to learn and each new year brings with it the need for skills renewal.

Subject Matter Expert

Role	Sometimes called "business owner" or "business user"; not normally a role played by an IT person
Starting Point	Typically has very little experience with development process
In the Toolbox	Expertise with business process for which a solution is being developed
Stand Out By	A deep understanding of a process, an industry, and in some cases an organization, as shown in published articles, industry presentations, etc.

SMEs are the people in the process who provide the information on what needs to be built. They serve in the most important role in the development process -- despite not being a part of the permanent development team.

Subject Matter Experts really fall into a few categories. The first category is the business owner who initiates the development process. For internal development, this might be the manager who is sponsoring the project. For external development projects it might be the customer paying the bills

SMEs provide all of the raw material for the development process. This includes the requirements for the system and how it will be used. Their input describes the problem or the opportunity that the software solution will ultimately solve.

SMEs are perhaps the most broadly described part of the development process. Because they can come from all walks of life, all levels of awareness of the software development process, and all levels of interest, trying to describe them is a futile process.

The lack of understanding regarding the process is not a critical limitation, because the SMEs will work with a Functional Analyst who will guide them through the process. Unlike most roles, which bring extra skills to the table, the SME removes some of the inherent skills that other members of the team possess.

As a part of a development team, here are some skills that you should be careful to avoid assuming a SME has:

- Don't assume that the SME will clearly communicate what they know.
- Don't assume that an SME will understand models developed by members of the team.
- Don't assume that an SME understands or can use defect logging and tracking systems.
- Don't assume that a single SME has all the answers.

There will always be a need for Subject Matter Experts -- particularly those who can clearly articulate the needs the organization faces. Although SME is not the primary role that an IT person typically fills, it's one that can be a great asset for an organization. If an SME shows particularly good skills at articulating the business needs, then perhaps there's the opportunity to take a part-time or full-time role as a Functional Analyst (FA).

Being a Subject Matter Expert isn't a career in the same way that being a developer is a career. In most of the roles in the development process, the core learning is around the skills and technologies of developing software. The SME is instead developing a deep understanding of a process, an industry, and in some cases an organization. The SME's value is their unique understanding of the problem that the development process is designed to solve or at least help resolve. In this way, an SME is focused on being the thought leader and expert for a small set of information.

In addition to writing articles and delivering industry presentations, SMEs can stand out in a less public way by learning how to interact with different personalities to develop a network of relationships in the organization or industry that they are working in. It is rare for an SME to clearly understand the challenges faced by the producer for the organization, the sales department, the executive staff, and all of the other various departments.

The Good, The Bad, and the Ugly

The Subject Matter Expert role in the software development lifecycle has its ups and downs, just like every other role within the process. Here are a few examples of what's good about the role and a few items to watch out for:

- Good: The role in the project is generally short lived. Projects tend to come and go
- Good: Subject Matter Experts are a generally well-respected and necessary part of the project.
- Good: Subject Matter Experts have a chance to interact with numerous people at all levels within an organization. This is often great exposure for being noticed within an organization.

- Bad: Since software development isn't the primary process of an SME most feel a bit like a fish out of water.
- Bad: Although generally bright intelligent people the rest of the software development team may have trouble understanding the business that a SME is describing since they are not a part of it. An SME may have to explain things from a couple of points of view for it to be fully understood.

- Ugly: Participating in a software development process may require more time than you're used to.
- Ugly: SMEs may have to interact with geeks and bear through discussions on topics that won't ever help them in their daily jobs.

Functional Analyst

Role	To capture, consolidate, and communicate information from Subject Matter Experts to the rest of the development team.
Starting Point	Developing precise communication that will allow discovery of inner meaning and inconsistency that is essential.
In the Toolbox	Communication and relationship skills, word processing apps, spreadsheets
Stand Out By	Becoming adept at dealing with differing opinions and conflict.

The typical software development project takes several Subject Matter Experts to provide the necessary information to create a solution. Because of this, the Functional Analyst is a critical link between the SMEs that provide the business requirements and the rest of the team, which is trying to construct the solution.

Depending on the organization, the Functional Analyst may be called by other names as well. Another very common label for the FA is Business Analyst, or sometimes simply Analyst. No matter what the name, the need to help capture, consolidate, and communicate the information from the SMEs to the rest of the team is the critical, bridge-the-gap role that this person plays.

The FA will spend a great deal of time asking questions like "What do you mean by that?" or "How does this fit in with what we were talking about earlier?" Questions like these expose potential, often subtle, differences in meaning between the SME and the rest of the development team. More importantly, these questions expose assumptions regarding business logic and processes that may not be clearly stated -- or even stated at all -- by the SMEs.

FAs are also responsible for identifying and resolving conflicting requirements. If SME No. 1 says the sky is blue and SME No. 2 says the sky is red, it will be the FA's responsibility to resolve that discontinuity.

Throughout the software development process, a document or set of documents are being developed. These documents, the requirements documents, will represent the contract between the business that wants a solution and the software development team that wants to create the solution. A requirements document is, at the basic level, a listing of all of the features or aspects that the final solution should have to fully solve the problem that the SME is describing.

These documents need to be understood both by the SME and the development team. The SMEs will need the document to validate that the requirements for the project are correct in every detail. The development team needs the document so they know what is to be built. To accomplish both objectives the documents must be brief but thorough. They must also be expressed in both business and technical language. Done correctly, they are the perfect balance between competing forces.

Working in positions requiring either leadership or detailed documentation best develops the communications skills necessary for the job. Leadership positions in professional or community organizations are an obvious target for training for the FA role, however, those roles require so many more things that they can often be distracting from the core skill that the FA needs to develop. A better role is the often-neglected role of secretary. While the obvious thought here is that the secretary is simply someone who is taking notes, the role can actually be an opportunity to safely develop the core skills for the FA role. Another benefit is that this role is rarely as contested as the role of leader of an organization.

Building a requirements document is usually a key responsibility of the FA. Creating a good requirements document requires an insight into knowing when detail is necessary and when additional detail would only serve to clutter up the understanding. Just as there is no one recipe for creating cookies, there is no single formula for creating an awe-inspiring requirements document. In many ways creating a good requirements document is as much of an art form

as it is about the science of capturing specific, numbered requirements.

The FA's technical toolbox is not extremely specialized (they largely have to use the same tools as other members of the software development team), however, they sometimes have to be more skilled with basic word processing, spreadsheet, and general office tools in order to support their deliverables.

It would also be good to have experience with a drawing program, such as Microsoft Visio, which allows for the definition of use cases, process flows, and other diagrams that would be nearly impossible to express in words alone. The ability to use the program to accurately depict a wide variety of complex ideas is essential to crossing the chasm between the SME's knowledge of the problem and the ability for the software development team to solve the problem. The generally accepted practice for diagramming is becoming UML (Unified Markup Language). UML allows you to describe relationships, states, and other common requirements that are best expressed graphically in a standardized way.

There was once a time when people predicted that everyone would be able to write their own software. They would sit down at a computer and just tell it what they wanted. The computer would write the program from this dialog, and thus developers in their current incarnation would no longer be a necessity. This vision is all but gone from the heads of most practitioners. As more became known about what people wanted to do with computer, it became clear that there would always be increasingly more complex problems to solve.

A part of that realization is the realization that our ability to accurately describe the problem determines the ability for the problem to be solved. Most people are incapable of clearly and precisely articulating -- to the level necessary -- the problems that they're trying to solve. This is a problem that is getting larger and not smaller.

This is the very problem that the Functional Analyst role has been created to solve. The FA's goal is to refine the understanding and communication from the Subject Matter Experts and convert that into the clear, precise vision necessary to create a solution. Because of the growing need to automate in order to be competitive and because of the increasing difficulty for clearly articulating true business needs, the Functional Analyst's role is more important now than it ever has been.

The Good, the Bad, and the Ugly

As with any role there will be good with the bad and then there will be the really ugly. The Functional Analyst has the opportunity to set the software development process on the right path by carefully controlling how the process gets off the ground. These are the key points for the role:

- Good: Key role in the definition of the solution. Being at the start of the process, the FA has the greatest opportunity of any member of the software development team to get the project started in a way that will create the best solution.
- Good: An FA has the greatest opportunity to interact with everyone on a project. This includes people on the development team, as well as people outside the development team. Often this can include higher-level people within an organization. Such exposure can be great for building a positive reputation and a strong career.

- Bad: Not all SMEs are created equal. The quality of SMEs that an FA must work with will vary greatly. Some SMEs will make the FA role easy and others will make the FA want to commit acts of violence.

- Ugly: For most FAs conflict will be a normal consequence of daily work. This can get downright ugly at times.
- Ugly: All fingers often point to the FA. If the FA does their job, then everything should work out. If something is found to be missing from the solution, then the FA is often the role that is blamed first.

Solution Architect

Role	Converting the requirements into an architecture and design that will become the blueprint for the solution being created.
Starting Point	Be the only person on small project; years of success as a Development Lead; invest time in learning patterns
In the Toolbox	A visual documentation language (like UML); database design; development tools and processes; ability to create consensus and understanding around the architecture.
Stand Out By	Researching technologies and approaches critical to the architecture; review patterns

For many developers perhaps the most sought after role is the role of the Solution Architect (SA). The essence of the SA role is the conversion of the requirements into an architecture and design that will become the blueprint for the solution being created. This conversion is based largely upon the previous design patterns that the SA has been involved with in the past, through reading and staying abreast of the latest techniques, or through personal experience.

It is this conversion part of the role -- the role of the SA -- that most often is underestimated in its complexity. Creating effective architectures to create a solution requires the careful balance of dozens of development concepts ranging from "Keep it Simple Stupid" to "Fail to Safe."

In the process of converting requirements to an architecture, there are often parts of the SA's role that seem out of place. For instance, there is often a fair amount of research that happens during this phase. The research may be targeted at testing a technology that will become critical to the architecture. For instance, the SA may test to see if USB or serial port access is available from Java if there's a need to read a device without downloading software. This process can either be done alone, or depending upon the size and velocity of the project, can be delegated to a development lead.

Another component to the role of Solution Architect is the motivation and guidance of the Development Leads. Development leaders need to buy into and accept the architecture to know how the pieces will fit together at a high level. They must also see the art portion of the architecture to get an appreciation of the subtle nuances of their portion of the architecture. It's the art portion of the architecture that makes it elegant. That elegance helps to maintain cohesion between various parts of the design and encourages simplicity. It is necessary for the lower-level design and approach to match the higher-level architecture for the solution to be cohesive. Once the development leader has internalized his or her portion of the architecture, the SA must continuously motivate and reinforce the good work that is being done. They must continue to motivate the Developer Lead(s) to push through tough issues and create the solution.

One of the ways to demonstrate an interest in the SA role, no matter what role you may currently be filling, is to invest time in learning patterns. Because patterns form the basic building blocks of nearly every architecture, learning patterns makes it far easier to identify where they can be helpful. Also, reading books and articles on different architecture perspectives and new development techniques can broaden your point of view and allow you to see opportunities to create your own small sections of the solution.

The distinction between a Development Lead and the SA are often subtle. Where the Development Lead focuses on detailed knowledge of a particular area, the SA is very broad. This allows the SA to view the problem from a different perspective. Instead of getting mired down into the details of implementing one specific thing the SA focuses on integrating various parts of the solution into one cohesive network that solves the larger problem.

The other subtle change is in accountability. While the Development Lead is responsible for their part of the solution, the SA is the proverbial one neck to choke if it doesn't all come together right. The SA has the ultimate responsibility

for making the technologies work together. As a result, the SA role comes with a requisite level of responsibility for the success of the project.

Perhaps the most critical skills for the SA are the ability to create consensus and understanding around the architecture. While a Development Lead may need to involve a few people in their detailed design, the architecture of the application touches every member of the team and there's a need to get them to understand it and agree with it. Once the SA has created the architecture it's time to communicate and sell it.

The Good, the Bad, and the Ugly

- Good: The SA is a key role and one which can provide immense value if done correctly. This generally means a healthy salary.
- Good: An SA is likely to get to interact with many of the key members of the development team, as well as key members of the user community. This makes it a very visible position.

- Bad: Being a SA means keeping up to date on a wide variety of new techniques, patterns, and tools. The effort to keep up can be very draining at times.
- Bad: The role requires balancing so many factors that it's difficult to get right. In other words, it's easy to fail.

- Ugly: Although a good FA can provide great requirements, a moderately skilled one may not. The difficulty is that most people, including seasoned SAs, have trouble spotting bad requirements documents before it's too late. The SA must always have to consider that the requirements may require the SA to do a lot more research and legwork into what the client really needs.
- Ugly: If a project fails, the SA is at the top of the list for people to blame.

Development Lead

Role	Mid-point between developer and solutions architect. Responsible for fleshing out details in the architecture and creating program specifications from which the developers work.
Starting Point	Master the key skills of a developer
In the Toolbox	Word processing, document creation, visual techniques (e.g., flow charts, diagrams), source code control systems, configuration management, troubleshooting
Stand Out By	A willingness to help out and support the SA when they get overwhelmed, and a willingness to take on challenging projects

The Development Lead (DL) is the one role that glues together the overall architecture created by the Solutions Architect (SA) and the developers. DLs are still rooted in the reality of the code and the capabilities of the developers they have working on the project. They were once developers themselves, and instead of spending all day every day coding their own tasks, they now lead and mentor developers as the developers have problems that they don't understand.

Whether the Development Lead has formal responsibility for the developers on an organizational chart or not, they will be the ones who will direct the day-to-day activities of the developer. In many organizations the Development Lead isn't burdened with the formal administrative management of employees, they are instead free to focus their time on helping developers be successful.

The DL role starts with the mastery of key skills of a developer and adds to that the ability to convert concepts into deliverable solutions. The developer must look beyond the tasks that they're doing and see the ability to create more reusability in the work he or she is doing. Developers who are trying to move forward into more efficient and thoughtful ways of doing things will be given more responsibility and will move forward into the development lead role. The transition from developer to Development Lead may happen in as few as three years, but more frequently takes six or more years. This is due in part to the broader range of experiences that the DL requires.

The toolbox of the Development Lead is much like the junk drawer of the average household. It contains a wide variety of implements that can be used in a variety of ways. Most of those implements, however, spend months languishing in darkness only to be revealed when the need arises.

The DL is often the primary person involved in source code walkthroughs. These code walkthroughs are designed to help the DL understand the code that is in the system so they can ensure its quality, but also serve as an opportunity for the DL to help coach, train, and mentor budding developers in ways of thinking about the code more deeply and more thoroughly. Although relegated to a trivial part of the process in many organizations the code walkthrough or code review is an important part in both the quality of the system being built and the long-term development of the developers who are working on the project.

One of the characteristics of a development lead is the ability to create their own tools when no tools exist to solve their problem. Their toolbox may be much like a junk drawer in that you never know exactly what will be in it. You can, however, almost always guarantee at least a few of the tools will be their own unique creations.

The Good, the Bad, and the Ugly

With any position there are good things, some bad, and a few ugly. The development lead position is no different.

- Good: The Solutions Architect role is just a hop, skip, and a jump up the proverbial corporate ladder. The Solution Architect role only requires a bit more experience and a few more skills. It's easily within reach of a development lead.
- Good: If a developer likes to code, then the Developer Lead role allows them to be promoted and spend some of their time coding if they want.

- Bad: The position has the potential to get struck between the grand vision of the solution owner and the reality that is available from the developers on the project. Occasionally, that gap is too far for anyone to bridge.
- Bad: Keeping up with new technology, while enjoyable, can be time consuming.

- Ugly: When there is more "hands-on" coding to be done than there are regular developers, then the Developer Lead is usually the first person pulled onto coding.
- Ugly: The DL can be presented with a great vision, but lacks the technology (due to cost or politics) or products to implement it effectively.

Developer

Role	Workhorse of the system, creating code
Starting Point	Syntactical and algorithm skill, volunteer coding
In the Toolbox	Familiarity with coding environment for language(s) they use, including compiler and debugger; automated testing tools
Stand Out By	Developing understanding, mastery of structures and algorithms, and specializations

From a technical perspective, the Developer is at the most basic level expected to be able to translate algorithms and technical specifications into code that can be executed on a computer system. The language syntax and structures of code must be understood. The language syntax for writing programs is typically the easiest skill for a Developer to learn. There are many books, training programs, and tools to teach you the syntax of the language.

In software development there isn't any one "right" way to do things since the same problem can be solved dozens of ways. However, there are ways that are "more right." Mastering structures and algorithms means that the problem is solved in the most straightforward manner no matter the language. Algorithms are step-by-step patterns that explain the process of performing an action. Structures relate to this by being the containers for the information that is being transformed by the action. When woven together they form the fabric of an application.

A popular question among those interested in a software development career is "What about offshore development?" While it is a concern for programmers, the reality is that the demand for developers is approximately flat according to the ITAA 2004 survey of the IT workforce. Although the outlook for new developers joining the ranks may seem dim, it's important to realize that programmers represent the largest portion of the IT workspace and that positions and opportunities will develop just through attrition and upward movement.

Once we accept that there are going to be enough development positions for everyone it's important to investigate the trends that will be impacting the Developer role going forward and how that can mean.

The Good, the Bad, and the Ugly

No role is perfect and the Developer role is no exception. Here are some of the things about the Developer role that you'll want to consider before deciding it's what you want to be doing with your career

- Good: Since the development area is one of the largest in IT, and because there are so many different specializations and roles within the software development process, there's nearly always the opportunity to move up within the hierarchy. The Developer is the foundation upon which most other positions are built. For instance, you're more likely to grow into a development leader role from development than from any other position.
- Good: Developers are indispensable to the organizations they work for. They are in point of fact the only people who truly understand how the systems do what they do and why they do what they do. As a result, programmers have a relatively stable position even in times of cutback. This is most true of developers who are doing maintenance on critical systems, but it can apply to all developers. This is not to say that developers are recession proof or that they can't be laid off, that happens in every field. However, when compared to other positions in the software development lifecycle, things are relatively stable.
- Good: If you like problem solving, the Developer role may be for you. Developers are in a constant cycle of building and debugging their code. Both sides of that cycle can heavily lean on a problem-solving skill. While coding, the Developer exercises problem solving by figuring out how to get a piece of information that's difficult to get. During the debugging part of this cycle the Developer is focusing on identifying the source of the bug or bugs and determining how to eliminate them.

Human Resources departments are often inundated whenever an IT position is posted. You need to be able to make your credentials or resume stand out of the crowd somehow - and resume tricks aren't going to do it.

Developers can differentiate themselves in a few key ways:

Get a certification
Certification programs exist from a variety of vendors. Microsoft offers MCSD and MCAD certifications for developers. Sun offers a SCJD Java Developer certification. These certifications will demonstrate that you know the languages you're professing to know. Other certifications, such as the IEEE's CSDP Certified Software Development Professional, also exist if you want to show a broader understanding of software development practices.

Get involved with a User Group
User groups, including special interest groups (SIG), are opportunities for professionals to get together and talk. Membership is a great way to show that you're interested in your own development and can be seen as a way of differentiating yourself. If there's not a user group for the development language or development type that you are interested in - start one.

Collaborate in a forum
Become a member of a group of people who speak on forums, public newsgroups, or in other venues. This creates a semi-permanent record of the knowledge you've developed and shows how you're sharing that with others.

Get recognized by a vendor or organization
Microsoft and other vendors have programs that recognize those that make significant positive contributions to the technical community. In Microsoft's case, it is called the Most Valuable Professional (MVP) program. Due to the relatively small number of professionals in these programs, having a vendor recognize your contributions makes your resume stand out.

Attend regional and national conferences and trade shows
In addition to the knowledge that you'll gain, you will also demonstrate a desire to stay up-to-date with the latest technologies and techniques. This will help you be perceived as more in touch with what is going on. Learning new techniques and tools helps broaden your

continued on next page

• Bad: Developers are often relegated to cube land with little interpersonal contact. Because of the need for focused time during development distractions are likely to be minimized. For those with an extroverted personality this might represent a challenge.

• Ugly: One of the ugliest things about being a Developer is that you're at the end of a very long whip. When the software development process works well the Developer feels the crunch of a deadline. When the software development process doesn't work well, and it often doesn't, the Developer can be crushed by conflicting needs to get the product completed and a series of quality or incomplete feature issues. The truth is that many managers in most organizations do a really bad job of managing the software development process, which can create very painful positions for developers.

Quality Assurance

Role	Responsible for guaranteeing a level of quality for the end client, and helping the software development team identify problems early in the process
Starting Point	A basic understanding of the process, and minimal -- if any -- prior experience
In the Toolbox	Patience; automated testing tools and the skills necessary to validate applications, data base values, and workflows when there is no easy way to validate the correct answer.
Stand Out By	Learning additional skills that will help you test; having the ability to read and understand code; problem solving

The QA role works with the FA and the SA to convert the requirements and design documents into a set of testing cases and scripts that can be used to verify that the system meets the client needs. This collection of test cases and scripts are collectively referred to as a test plan. The test plan document itself is often simple, providing an overview of each of the test cases. The testing cases and scripts are also used to validate that there are no unexplained errors in the system.

The test plan is approved by the SMEs and represents the criteria to reach a project closing. If the test cases and scripts in the test plan are the agreed upon acceptance criteria for a project then all that is necessary is for project closure is to demonstrate that all of the testing cases and scripts have been executed successfully with passing results.

horizons and demonstrates a continuing commitment to learning. Additionally, when you attend a conference, take the time to try to meet people.

Speak at a conference
The process of speaking at a local conference is relatively easy. You need only become an expert in one small area - an area that is interesting to others in your geography. Smaller, regional conferences may not have more than 30 people in a room. Many conferences do a "call for papers." You can respond to these calls by sending in an overview of what you would like to present.

Write an article
Being published distinguishes your resume from others because there are relatively few people who have ever had an article published professionally. Although getting your first article published isn't easy, it can be within reach if you're willing to make the investment. Even those who have poor spelling and grammar skills can become proficient at writing articles.

Write a book (or part of a book)
If the field is narrowed from a floodlight to a spotlight by writing an article, it's laser focused by writing a book or a part of a book. Even though the number of those who have written articles is small, those who have written a book is even smaller. Writing a book shows a more comprehensive knowledge of a subject, beyond just what a single article might show. Writing a book also shows commitment.

Network with others
Attending a user group or conference, answering questions in forums, and interacting in other ways is a start, but take the next step as well - meet the people there. Interacting and networking can lead to connections that will distinguish you.

A test case is a general-purpose statement that maps to one or more requirements and design points. It is the overall item being tested. It may be a specific usability feature, or a technical feature that was supposed to be implemented as a part of the project.

Test scripts fit into the test cases by validating that case. Test scripts are step-by-step instructions on what to do, what to look for, and what should happen. While the test cases can be created with nearly no input from the architecture or design, the test scripts are specific to how the problem was solved by the software development team and therefore require an understanding of not only the requirements, but also the architecture, design, and detailed design.

The quality assurance role is split into three parts:
- The role creates test cases and scripts.
- The role executes or supervises the execution of those test cases and scripts.
- The role facilitates or performs random testing of all components to ensure that there's not a random bug haunting the system.

In some organizations, the Quality Assurance role has two specializations. The first is the classic functional testing and quality assurance as described above. The second is a performance quality assurance role where the performance of the completed solution is measured and quantified. The performance QA role is an important part of the large system development quality assurance process.

The Quality Assurance role also has within it a wide range of potential titles and specific responsibilities, from the entry-level QA professional who executes and document tests to the QA lead who works with the FA and SA to create the testing plan, cases, and scripts. The role also extends through QA manager position that may take responsibility for the quality of a solution. At this level, the QA manager and solutions architect work as peers to ensure the final solution has the highest quality.

Split evenly with the developer role the standard QA role is an entry role into the software development process. The QA can be entered with a basic understanding of the process, and minimal -- if any -- prior experience.

The larger and more public the system, the more important performance testing becomes. Because of this, the skills of performance testing and interpreting the results of that testing become more valuable as the system's importance increases. It is common to have QA analysts in a functional testing role; this is the kind of role that most people think of. QA analysts are also found in a performance-testing role, which is focused around the performance-testing aspects of a software solution.

Quality in general has taken a beating over the last several years, as high-profile software failures have made it into the media and because there is a growing awareness that viruses are possible only because of a quality slip in the software that the virus attacks. Despite a growing awareness of software development failures there's little growth in the QA role. This may be due to a variety of factors including: misunderstanding of what the QA role, the lack of understanding of the value, or simple ignorance of the need for Quality Assurance.

There's a general misunderstanding by a great deal of business people and technical managers about what testing is. It is believed that testing is something that the developer does; to some extent this is true, most developers do perform testing. But, whether you believe in the psychological concept of objective introspection -- the process whereby you investigate your own thoughts, actions, and feelings -- or not, it's clear that having the same person who developed a program test it leads to many more errors to be found later on. This is because the developer makes the same mental mistakes in testing as they did during the development. The result is that the defect slips through until much later in the process when it's more expensive and harder to solve. As a result, even though the developer may run the initial unit tests, it is unwise to rely entirely on the developer for testing. The QA role is designed to validate the developer's tests but also to ensure that the work of several developers fit in together.

In some software, the level of appropriate testing is so low as to make it seem completely unnecessary to have a QA role on the project. For instance, the development of a testing harness -- code designed specifically to test other code, is rarely done with any testing whatsoever. There is little need to test the harness itself, as it will be well tested as it's used as a tool to test other things. The thinking of many business and technical managers is that testing has value, but the value it has is low enough that less testing than might be appropriate is done. This misperceived position on quality sometimes prevents managers from allocating enough resources to the QA process.

In some cases when planning for the project the QA need is ignored completely. This is due to complete ignorance that this part of the process exists and that it's a vital part of the software development process necessary to ensure a quality result.

The Good, the Bad, and the Ugly

Never was it truer that each role is met with good things, bad things, and truly ugly things as it is in the QA role. The role in the right environment is a shining example of how one person, or a few people, can make an impact. However, in the wrong situation it can leave someone bitter, frustrated, and burnt out. Here are just a few of the good, bad, and ugly things about the QA role.

- Good: The QA professional can take pride in their ability to instill quality in the result of the software development effort. More than any other individual, the QA professional can make sure that a quality product is being produced.
- Good: Because all aspects of a system have to be assured for quality, the QA role often allows a person to be involved in many phases of a project, including the early ones.

- Bad: One of the challenges is that the QA role is designed to be the bearer of bad news. Because of this it's easy for others on the software development team to develop a negative attitude towards the QA professional. That means a lot of work on the QA professional part to keep everyone in a positive frame of mind.

- Ugly: There's only a certain amount of quality that is appropriate for any given program. Some programs, such as those dealing with life or death situations, require substantial testing. However, many of the systems that we work with today are not life-and-death situations and require only an appropriate level of quality assurance. The ugly part is that no two people will draw the line on how much quality assurance must be done in the same place.
- Ugly: Testing generally occurs after other things are done. This means that testing is often close to the deadline. If others are late in completing their tasks, the QA person may have their time squeezed to near nothing.

Deployment

Role	Create a program that constructs the operating environment that the solution needs, including installing prerequisites, making configuration changes, identifying conflicting software, and manipulating any components of the environment the solution may need.
Starting Point	Experience on the help desk, servers, and networking side of an IT department, or previous work as a developer with a focus on identifying issues that become challenges during deployment.
In the Toolbox	Batch files, scripting languages, packaging tools, package management tools, virtualization software, and disk image software.
Stand Out By	Playing multiple roles, as a developer or QA, for example.

A software solution of any complexity will have dependencies that must be present before the solution can be used. Many of these dependencies go unstated. For instance, a Java program needs a certain level of the Java runtime environment installed to be able to run. .NET-based applications require a specific version of the .NET framework and common language runtime to run. In the case of database applications, specific versions of the software drivers to connect to the software to the database are required.

In addition to these software dependencies, there may also be hardware dependencies. This could include a minimum amount of memory, a required amount of hard disk space, access to multiple machines (such as a database server or an application server), access to the Internet, and more.

The more complex the solution, the more prolific the requirements will be; solving that problem is the role of the Deployment professional. It is the focus of the Deployment role to create a program that constructs the operating environment that the solution needs. This includes installing prerequisites, making necessary configuration changes, identifying conflicting software, and manipulating any other components of the environment that the solution may need.

The first step in the Deployment role is to identify and test by hand all of the steps necessary to make the software work in an environment. These instructions are often written as complete documentation so that even without the installation program they can install the software. These installation instructions sometimes serve as the early procedure for installing the software through alpha pre-release program and even throughout the life of the software if it is installed to a small group of clients.

From the installation instructions, an installation program is constructed to automate the steps identified during the creation of the installation instructions. The program must do more than just install the system, it must also, in today's world, be able to uninstall the software once it has been installed. This must also be done in a way that is respectful of the other software that is already installed in the customer's environment. Figuring out how to be a good software citizen with an installer is one of the most challenging things that any software development team can do.

The Deployment role is challenging for a variety of reasons, but none more important than the need to understand both the software development process as well as an understanding of the operating system and the network infrastructure that may be used to deploy the software. The requirement for understanding the software development process may seem obvious. They're a part of the process so they should understand it; however, the required understanding goes deeper than this. They must also be able to identify inherent dependencies created by the development process that may not readily be obvious.

A command of the operating system (or systems if the software is to be used on multiple platforms) is also necessary since it will be the Deployment professional's role to manipulate the operating system from their installation program. The subtleties of whether to register a .NET assembly in the global application cache or not is an important thing for a Deployment specialist to know. Considering all of the code access permissions and learning how to setup those permissions so that the user will be able to run the application is also important.

The final component that the Deployment professional must understand is how automated deployment tools, such as Group Policies in Active Directory and products like Microsoft's Systems Management Server, deliver software to the clients and how the installations must be constructed when being deployed via these mechanisms. This is a critical consideration when the software will be deployed across the enterprise since it's not feasible to go install the software manually on each machine.

The final consideration for what the deployment professional will do is creating patches. Software teams necessarily create updates to their solutions. These updates need to be deployed to systems just as the original program was. Most consumers expect that the patches that they receive today will integrate into the existing installation rather than being another program that must be added or removed. Deployment professionals are entrusted with developing strategies that deliver patches efficiently.

The Good, the Bad, and the Ugly

The Deployment professional can often experience the good, the bad, and the ugly of their role in one day. They can see the software get deployed across the organization -- only to find out that here was one application it wasn't

tested with, and that one application is one of the mission-critical applications for the organization that the installation just broke. Here are just a few of the good, bad, and ugly things about the deployment role:

- Good: The role is one of high visibility and high impact to the user community. When done right it's easy to save hundreds of hours for the users.
- Good: Often get to use a number of different systems with a number of different configurations.

- Bad: Being near the tail end of the process the installation often gets rushed. The testing time to ensure that the installation doesn't break other applications is often hard to come by.

- Ugly: Building good installations is still very technically challenging today. Over time, good Deployment professionals learn how to avoid big mistakes, however, even veterans are known to make mistakes that have a huge impact.
- Ugly: No matter how many systems you test your installation against, it always seems that a user will have a system that is slightly different. When something goes wrong with the installation, you get blamed.

Trainer

Role	Creates the materials necessary to train users how to use software.
Starting Point	An instructional design background and understanding of the methods adults use to learn.
In the Toolbox	The ability to communicate complex topics with precision and understanding; ability to speak publicly. Word processing, layout/design, presentation/PowerPoint, and virtualization software.
Stand Out By	Mastering different media for training, including online, in-person, teleconference, etc.

There are more training materials than ever in the history of computers, and yet there is still a need for more and improved training. Because of the increasing complexity of software the need for good training is becoming substantially more important. Because of this, the need for future training professionals is well assured. Add to this the proliferation of different mediums for training, including various electronic learning formats and it's easy to see that there's plenty of work to be done.

Training is also a part of the process that will always be personal and therefore often face-to-face. Learning is a sensitive process. We like to learn from instructors that we can trust. In many people, trusting in an instructor requires seeing them. In a globally oriented economy the trend is to source talent from the most inexpensive location. However, because training is so focused on conveying information it's often difficult to relocate. In other words, it's unlikely that the positions will be transferred to areas where cheaper labor is available.

The Training professional is ideally someone who has an instructional design background and therefore understands how to create materials that are effective in helping adults learn. They are also, ideally, someone who can approach the problems that the software solves in a way that makes sense to the users.

Training materials range from slide decks to handouts for training sessions to self-study guides. No matter what the type of material, the focus is on creating a situation where the user will find it easy to use the software -- even if the user interface isn't perfect. The trainer needs to find a way to communicate what the product does and how it does it. Whether the medium is a standard printed manual or an interactive video, the goal is still the same -- efficiently convey how the user can best use the solution. The trainer's role is often the role of the super user. Because they must instruct on the software's use, they need to have a detailed understanding of the software across nearly every feature.

In addition to the creation of training materials, the Training role also often delivers the training itself. The training can be instructor-led training like the traditional classroom, a Webcast, or one-on-one sessions with key users. Performing the actual training sessions often provides invaluable feedback to the development of training materials

themselves. It helps to expose the kinds of questions that are in the minds of the users -- the kinds of questions the trainer must be prepared to answer.

In some cases a special form of a trainer, called an evangelist, may evolve in the software development process. An evangelist is a passionate, charismatic individual who evangelizes the product in the market. Evangelists are almost universally someone capable of filling the Training role, but whose passion and charisma make them stand out as advocates for the product. While they can and often do perform roles similar to the rest of the professionals filling the training role, they often side step or short circuit detailed training and answers in favor of engaging the user in the vision of the solution.

Although an understanding of instructional design is greatly appreciated in the training industry, it is usually not a prerequisite. If the role is primarily focused on delivering content that has already been created, then the ability to speak comfortably in front of a group is often the only requirement. These roles are often entry roles in an organization. More advanced candidates will have the ability to anticipate what the group of students will need and have the ability to "think on their feet." The training role is filled with twists and turns as students ask sometimes blindly simple questions and other times ask the most arcane trivia. Learning how to control the students is yet another skill that develops for most professionals over time.

The Good, the Bad, and the Ugly

The training role, like all of the roles, has its ups and its downs; here are a few of the highlights.

- Good: More than anyone else in the software development process, the Training professional gets to see the real results of the work. The Training professional gets to see the smile (or frown) on the faces of the end users when they are introduced to the solution.
- Good: Get to meet and interact with lots of people.

- Bad: Training is at the end of the lifecycle. It's the last thing that has to happen to get the system fully functional. Because of its location, training is often compressed into less time than it should have both from a content creation perspective and from the training delivery standpoint.
- Bad: Often the trainer has the thankless job of working with users who are frustrated because their process is changing. It may take a great deal of energy to keep the users happy with the solution that is being delivered to them.

- Ugly: Training means working with all kinds of users, including both highly respectable users and crackpots who will have to be endured.
- Ugly: Some trainers are required to do a lot of traveling.

Project Manager

Role	Help ensure that the software development process works as it is intended
Starting Point	Experience as a Functional Analyst; develop templates and processes.
In the Toolbox	Project management software, membership in a project management organization, templates and processes
Stand Out By	Showing an ability to adapt to an ever-changing environment and the ability to right a project that has gotten off track

The Project Manager role is designed to help ensure that the software development process works as it is intended. The Project Manager works closely with the Development Manager role in order to facilitate, encourage, and prioritize the process.

The project management role is perhaps the most clearly defined role within the software development process due to the development of project management as a profession.

While the software industry is nascent, the project management industry is enjoying the advancement of a powerful organization in the Project Management Institute. They have assembled a guide to the body of knowledge for the project management profession that is often referred to as the PMBOK Guide. This organization has developed a widely recognized certification, Project Management Professional (PMP), which has both practical experience requirements as well as traditional testing requirements.

The PMI's PMBOK Guide defines the Project Manager role by defining what project management itself is.

> *"Project management is the application of knowledge, skills, tools and techniques to project activities to meet project requirements. Project management is accomplished through the application and integration of the project management processes of initiating, planning, executing, monitoring and controlling, and closing. The project manager is the person responsible for accomplishing the project objectives."*

You can go to the PMI Web site (www.pmi.org) to download sample content from the PMBOK Guide, buy a copy of the guide, or sign up to become a PMI member.

Within the context of the software development process, the Project Manager role is responsible for driving the work through the process and to completion. Starting with the earliest requirements (discovery sessions) and ending after the training has been completed, the Project Manager is the one role that should be consistent throughout. Project Managers work with both the development team and with business stakeholders to ensure that what is being built will match what the customer expects and that this development occurs within the expected timeframe

The Project Manager is generally responsible for status reports, which are frankly more tools about instilling a sense of urgency and demanding definite answers than they are a process for generating reports. They expose resource and scheduling issues as well as provide a communications vehicle for issues as they begin to occur. The project role is typically spread across several projects, and potentially, development teams. The Project Manager is frequently working on more than one project at one time -- unlike many of the other roles in the project that may be dedicated. The Project Manager need not be an IT specialist, although it is helpful. The nature of the role is to provide broad-based oversight.

In the software development process, the Project Manager is one part prodding and one part smoothing over problems. They're there to make sure that the process goes forward and they use a variety of processes, meetings, and documents (artifacts) to ensure continued forward progress and to identify problems as soon as possible.

The Project Manager role must be capable of executing details with regular consistency. They must also be able to take a step back from those details to look at the big picture to evaluate things like risk and the accuracy of time estimates. All of this takes a measure of restraint and good listening skills. The restraint prevents the project manager from jumping in before it's time and the listening helps to ensure that the real problem is eventually understood -- and can be disseminated to everyone.

Project Managers are the keepers of the process. In some organizations this includes the software development methodology. The Project Manager is the one that makes sure that the process is adhered to so they can get the predictable results they desire.

The Good, the Bad, and the Ugly

The Project Manager role, like other roles, has its high point and its low points. Here are some of the key things that this role has to offer both positive and negative.

• Good: Project Manager is a high-visibility role. Because it has a part in so many different projects, the role is often involved in reporting to executive management in an organization so it has the potential to demonstrate good work to important people.
• Good: The role is a key to the project -- one that can have a great impact on the overall success or failure of the software development process. If you like influence this is a role for you. Additionally, the Project Manager is often considered the person within IT that owns and is responsible for the project.
• Good: Because a project management role is one that works with a large number of people, there's constant progress being made. When a project gets shipped, or even when the first beta comes out, there can be a great sense of accomplishment in that the project was moved forward because of the role that you played.

• Bad: The project management role is at its most crucial when the project has gone awry or when the timelines are too tight. Unfortunately, with the pace of business accelerating, it seems like every project has a tight timeline and doing what is necessary to ensure that timelines are met often creates a high degree of stress.
• Bad: The value that Project Managers can bring to the software development process can't be understated, nor can the amount of work that it takes to create this value. The artifacts that drive consensus also require time to create. However, there's not a solid understanding of the amount of work that is required to create the value and because of this Project Managers often find themselves stretched too thin across too many projects.

• Ugly: Perhaps the most frustrating thing about the project management role is that it is focused on fixing political turf wars, keeping people working with each other, and moving forward. This means that there's a heavy dose of pop-psychology that must go on to get the project done. The drain caused by so many whining people can be overwhelming at times.
• Ugly: The Project Manager is the one that may have to make the decision to have people work extra hours to meet deadlines.
• Ugly: The Project Manager is the one that has to maintain a project's scope -- what will or won't be included. They are often caught in the middle when decisions need to be made on what can be completed within the given time frames.

Development Manager

Role	Keeping the vision on track; "CEO" of the project
Starting Point	Learn how to communicate with the business leadership and how to adapt software development priorities to match the business goals.
In the Toolbox	Methodologies, practices, career development process, scheduling tools
Stand Out By	Learning how to communicate and evangelize to the team, management, and the rest of the organization

The role of the Development Manager is critical to the long-term success of the software development team. The role that the Development Manager plays -- particularly when interacting with the Project Manager -- is essential to a continuously improving process.

The Development Manager keeps the vision on track. This is much like a CEO, who sets the vision for an organization. This, of course, differs from the COO, who (like a Project Manager) ensures the day-to-day operations. While it's the Project Manager's goal to get the project to the finish line, it is the Development Manager's responsibility to look ahead to make sure the finish line is the right finish line to be reaching. While the project management position is a management position, the Development Manager role is a leadership position.

The Development Manager works closely with the Project Management to ensure that the projects are completed. They also spend time working with the business owners on planning and preparing for new projects that will soon be consuming the group. They're constantly making small changes in the current project to get a better product,

develop a better skill set in the group, or just generally preparing the development team for the next hurdle that it must face.

While the project management role is dutifully executing the current process, the Development Manager is examining, evaluating, and assessing the impact of potential changes in the market. The Development Manager is constantly evaluating ways to improve the skill set of the group. That can be through the use of a new tool or technique, or additional training on fundamental skills that the group already knows but doesn't execute consistently. The Project Manager's short-term focus allows the Development Manager the ability to focus on longer-term objectives. In addition to the long-term objectives, the development management role is intimately involved with prioritizing multiple (nearly always conflicting) priorities across software development projects. With the help of the Project Managers, the Development Manager intertwines the activities of multiple projects to improve the efficiency of the team.

In some organizations, the Development Manager is also responsible for evangelizing to business leadership. While the Training role was responsible for evangelizing to the "rank and file," the Development Manager evangelizes the solutions, the process, and the team to business leadership and customers. This portion of the role is more marketing than development, but is an essential part of ensuring that the business is aware of the value of the software development team.

The Development Manager also is often the cheerleader that pumps up the development team, encouraging them to remember the vision, to be a part of greatness, and generally be excited about the work that they're doing. In non-software development or consulting companies, the IT management often plays the Development Manager role. Often times, the IT manager leverages the strengths of a project manager or a solutions architect to fill the needs of the development manager role, retaining tasks such as cross prioritization of projects.

People filling the Solutions Architect role or the Project Manager role have the best chance of moving into a Development Manager role. The Development Manager who comes from the Solutions Architect position generally leads from technical strength. Conversely, the Project Manager who moves into the Development Manager role leads from a process strength.

This content was adapted from EarthWeb's Developer.com Web site and written by Robert Bogue.