DOWNLOAD PDF GUIDE TO SOFTWARE DEVELOPMENT

Chapter 1 : Scrum Guide For Outsourcing Software Development - Outsourcing World

The Agile Alliance and the Scrum Alliance quickly emerged, along with the classic, Agile Software Development with Scrum. Certified Scrum courses became commonplace, and terms like "Scrum Master" and "self-organization" entered the project manager vernacular.

Get the answer Jan 16, , 4: Is this just a development computer? Are you doing compiles? Running an android emulator? For instance, if you are running VMs, or doing compiles, you might want an i CPU since it has 6 cores. You might also want 32GB ram if running VMs. If it is just a development computer, then multiple monitors is probably an important consideration. A machine you are coding on, one of the most important things is the keyboard. You may not want a gaming keyboard, but you may still want a mechanical keyboard. I will look at changing to the i CPU since it has 6 cores. At the moment, I will be primarily using the machine for learning other coding languages. I currently code in. PC gaming is just something that I am interested in doing as well, however, I am currently a PS4 gamer, so it is not essential to my build. All being said, I just want a fast machine that will handle any coding that I want to do for the next 5 years or so. Speed is very important. I have not decided on the monitor yet. Do you know if my video card will support 3 monitors if I decide to do so? Jan 16, , 5: To run three monitors you might want three display ports. If you change to a , then you have to change to an X99 motherboard and want to change your RAM. The is a quad channel memory controller and you want to install memory in groups of 4 for maximum performance. If you are depending on this box to make money, then you need to think about storage. For projects and for backing up your work. NAS storage is even better for your backups since it is physically separate. Do I need to have more wattage for my PSU? The mouse and keyboard are free with this purchase. As for the storage, I only have GB to transfer from my other machine. I can always add later or use my external GB HD. I appreciate all of the feedback and I do think this will be a better machine for software development. Let me know if I am really missing something huge besides a monitor, I will shop around for those.

Chapter 2 : A 7-step Guide to GDPR Compliant Software Development â€“ InData Labs

This book presents a guide to navigating the complicated issues of quality and process improvement in enterprise software implementation, and the effect these have on the software development life cycle (SDLC).

The regulation requires businesses to protect the personal data and privacy of EU residents. And non-compliance could cost companies dearly. GDPR pertains to the full data life cycle, including the gathering, storage, usage, and retention of data. GDPR applies to both automated and manual data processing. For companies in order to become GDPR compliant it is important to understand what is personal data. What is personal data Personal data is any information that relates to an identified or identifiable living individual. When collected together, different pieces of information can lead to the identification of a particular person. These pieces of information constitute personal data. But a person can also be identifiable from other information, including a combination of identification elements such as physical characteristics, pseudonyms, occupation, address etc. Personal data that has been de-identified, encrypted or pseudonymised but can still be used to re-identify a person remains personal data. However, personal data that has been rendered anonymous in such a way that the individual is no longer identifiable is not considered personal data. For data to be truly anonymised, the anonymisation must be irreversible. It is necessary to keep the info that allows to identify the person and the data about this person separately. New rights for data subjects. All the personal data about an individual has to be deleted upon request. It will be necessary to inform the authorities and users about any data breaches within 72 hours after they are found. Privacy by default and privacy by design will be discussed further. Starting to work with any software, the user should have settings with maximum privacy. In case the user does not make any changes in the settings, the protection level should remain unchanged. The application should not require any actions from users to obtain maximum level of personal data protection. Introduce privacy into your software from the very beginning, even before the first piece of personal information gets into the system. Privacy should be at the core of any software and not be installed with some plugin. Such software will be illegal when GDPR becomes effective. Identify personal data and the processes that use it. Set up and maintain a personal data register. It can be a separate document or part of the Information Asset Register. Use this tool to keep records of personal data you collect, indicating places where it is stored, responsible file owner, access level, storage period, data accessibility etc. Determine in advance who in your company maintains this registry. Use of personal data must be reduced to the minimum sufficient level to achieve the goal of processing. Minimize user identification wherever it is possible. Embed the function of deleting unnecessary and used data. This step will not only protect the privacy of users, it will save you from a headache in case of hacker attack on the app. Record the implementation of GDPR rules. Even if the company followed all the regulations, but forgot to document it, all the undocumented measures will be considered unimplemented. The audit will show that nothing has been done in the company to become GDPR compliant. Obtain informed consent for the processing of personal data. In order to process the data, it will be necessary to obtain consent from users in advance. The consent should indicate, how the information will be processed, who and how will transmit it to another country. The text must be unambiguous and understandable. Implement information security measures. The regulation has increased penalty charges for information leakage. From now on companies will not only pay for hacker attacks with their reputation, but will be subject to serious fines. Therefore, it is necessary to take care of the protection system during the initial stage of software development. The authors of GDPR refer to encrypting as one of the measures, but software creators are free to choose any protection measures they find relevant. Under the shadow of GDPR, it is easy to understand why companies are concerned about the regulation with the potential to have such a profound impact on their business. The GDPR represents a huge shift in the way businesses will be expected to handle data. However, compliance with GDPR will soon become a point of differentiation, and the sooner businesses are GDPR compliant, the sooner they will begin to reap the rewards of standing out from the competition. Enhanced customer loyalty and confidence that their personal information is in good hands and maximized selling opportunities, are just one of many other benefits the companies will be getting. Subscribe to our blog

newsletter Data science use cases, tips, and the latest technology insight delivered direct to your inbox. Please leave this field empty.

Chapter 3 : Guide to Running Software Development Projects

This Guide to Software Development addresses the probl Furthermore, the decision whether to make or buy software is not a simple one. This Guide to Software Development addresses the problem of how best to make such decisions, and what effect such decisions have on the software development life cycle (SDLC).

Australia Are you thinking about where to outsource software development? Western Europe Not that long ago we conducted a research on mobile app development cost and chose a few Western European countries to reveal average hourly rate for IT services. However, not all of them appeared to have an office in the above-mentioned countries. Despite their claims to be a local software development company, some of them have ODC in India or Africa. Eastern Europe Although being close geographically, Western and Eastern Europe differ a lot in terms of software development rates. These developing economies have already got a high level of education and noticeable contributions to IT industry, but their salaries remain relatively low. Currently, Ukraine, Belarus, and Russia are the top outsourcing locations of this region, leaving Czech Republic, Poland, Hungary, Romania and Baltic countries far behind based on quality to price ratio. Find out developers rates based on your specific requirements today! Central Asia Mostly when we talk about offshore outsourcing services, such countries as India, Philippines, and Indonesia come to our mind. This region used to be a heart of outsourcing for a long time. But in the situation changed drastically with Donald Trump threatening to the industry in the USA. Having lost their rich clients, local software developers decreased their price even more. At first glance, cheap software development may look tempting, but in the long run such low rate for offshore software development services result in low quality and the things standing behind it â€" underpaid programmers and lack of motivation. East and South Asia And here we have the similar situation to the one faced in Europe â€" a dramatic split of prices. The golden middle between Chinese ambitiousness and Indian cheapness can be found in Indonesia, Malaysia, and Vietnam. Canada stays on the US trail and sometimes even outruns it. However, the market here is flooded with the offshore development companies with actual addresses in Asia, so rates here can vary. Large investments of the neighboring countries like USA and Canada created this pool of cheap software developers â€" demand creates the supply. Though the demand of a rich country creates the supply at the higher price. The proximity of Mexico, Panama, Brazil to the northern part of the continent makes this outsourcing destination attractive despite the rising prices. Hence, the cost of the offshore software development services is expected to go up in the future. Africa Newly discovered IT continent of Africa pleases businessmen with its affordable rates, but the local market still lacks experienced specialists due to its freshness. Offshore development would be cheaper to the north of the continent with the prices similar to Asian. Such fluctuation is caused by severe skill shortage the region is experiencing at the moment. Australia The same happens in Australia. The digital era made the latter criterion more vague as it allows more possibilities for remote work.

Chapter 4 : The essential guide to software containers in application development

My video course to teach beginners all they need to know about software development. JavaScript programming, how to use the tools to code, the best practices, and how to run your own software.

Trying to get a job in the world of software engineering or software development can seem a bit overwhelming nowadays whether you are a seasoned pro who finds themselves unemployed or brand new to the field. What is Software Engineering? Software engineering is developing software programs for computers using engineering design processes. They may create different computer programs people use on a daily basis, or they may engineer foundational computer systems, also known as embedded software programs. Many people think of software engineering as writing code, but that is just one part of this career. Software engineers may also design programs, test them and evaluate their performance throughout the development process, called the software development life cycle SDLC. What is Computer Engineering? Computer engineering is conceptualizing and developing physical pieces of technology, also known as hardware engineers. This can include anything from computer chips, to actual computers like desktops and laptops, but also any device that uses computing technology and these days there are a ton of those. Printers, cameras, video game systems, memory devices and smart TVs are all designed by computer engineers. Computer engineers may also work closely with software engineers to develop products that have synergy between physical design and operation. What is Software Development? Software development is the process of gathering requirements, specifying details, architecture design, documenting, testing, and troubleshooting involved in creating software applications, software frameworks and software components. Application development and software design are two other terms for Software development. It is a more creative field than Software Engineering or Computer Engineering, though these career paths often grow near one another. Software developers help to create programs for computers and other electronic devices that serve a specific function. Advertisement They research, design, code and document their programs. They also test them and fix bugs, typically throughout the life of the product, which include interaction with end users and business clients, 3rdparty vendors and suppliers. Who is a Software Developer? Now there are also several bootcamp type of programs that help enterprising and hard working people become a software developer in just a few months. In order to have a successful career in Silicon Valley, software developers need to have a unique combination of skills and personality traits including working well with a team and the ability to analyze programs and solve complex problems. The difference between software engineering and computer engineering is much more apparent than the subtle differences between other development and engineering positions in the tech world. Advertisement Computer engineers work on physical hardware while software engineers work on the programs and the coding language that make them work. Computer engineers are knowledgeable about electronic engineering and design the physical products, while software engineers help to create the programs, databases and other internal codes that keep them running and functional. Both positions may be knowledgeable about some of the same topics, including software development and integrating hardware and software, but they diverge at the ultimate job function. The difference between software engineering and software development is subtle compared to the differences between computer engineering and software engineering. Both software engineers and developers work on software codes. They develop, test and debug codes and computer software programs. However, there are definitely some differences. Software engineers use scientific engineering concepts - the same ones that other types of engineers, including mechanical or electrical engineers - to develop software. Software developers can often learn on-the-job and develop on-the-fly. While these two terms may be used interchangeably sometimes, there is a slight difference that mainly refers to the mindset of the position. What is Software Design? It is both the process of conceptualizing the full architecture of a program and how databases and system components will work together as well as the continual improvement of the end result. It includes software architecture design or top-level design, which details how the system will be organized upon completion. There is also detailed design, which includes more specific information on how each of the components of the software will function. Advertisement What is Computer Programming? Computer

programming is the act of telling a computer what to do. Computer programmers are rather bossy when it comes to machines! Computer programmers also referred to as embedded programmers or hardware programmers may create anything from a complex operating system to a simple calculator program. The thing that is constant about every program is that they all use a programming language. The aforementioned languages are all compiled languages , meaning a human writes them, but they must be compiled afterward to get into a format that can be read by computers. Interpreted languages like JavaScript and Perl can be read by both humans and computers. What is Computer Coding? Computer Coding is the act of using a software programming language to create an app, system or website. Coding is talked about a lot in terms of the fairly recent popularity of the Information Technology IT industry. Advertisement However, computer coding is just one aspect of computer engineering, programming and development. Coding means using a computer programming language to create an app, system or website. Most programmers and engineers are experts in at least one programming language, but often know multiple languages. Coding, in general, can create websites and mobile apps, but computer coding specifically refers to the development of computer programs and systems. What is Software Testing? Software testing is an important part of the software development lifecycle. After everything has been designed, programmed and developed, it must be tested. Many computer programs are in a constant state of testing and improvement, proving that the software development process is not done when a product or program is delivered to the public. Software testing or software quality assurance process makes sure the program meets the goals and design components planned in the initial software design phase, and that all the software functionalities intended for the program is present and working properly. Advertisement History of Software Development and Engineering The very first piece of software was run at 11 am on June 21, on a computer that was affectionately nicknamed Baby, but officially called the Manchester Small Scale Experimental Machine. The program was written by the early software developer Tom Kilburn and it was basically a giant, slow calculator. It took nearly an hour to perform the first calculation programmed. These early giant-sized computers were programmed with punch cards. The holes in the cards told the computers what to do. Developers created the cards without interacting with a computer. The field continued to develop from this moment on, through the decades of the 20th century, leading us to where we are today. Starting in the s, the popularity of personal computing started to speed of the field of software development. As you can see by the smartphone in your pocket and the laptop in your backpack, we have come a long way from the software that was released on the first PCs back in the 70s. The first spreadsheet, word processing and visual design programs were released in the 80s. The invention of the internet in the 90s innovation spurred more innovation. Modern software developers develop programs that create thousands if not millions of calculations per second - a long way from the 52 minutes that Tom Kilburn and his team had to wait for their one mathematical solution. In fact, the fastest computer in the world can perform more than 33 trillion calculations per second! First Software Developers and Engineers in History The first software developers and engineers were paving the way for all of our modern technology. They were at the cutting edge of science and exploration for their time. These are just a few of the important figures we remember as forging technology forward in the analog age. Ada Lovelace Ada Lovelace is often considered the first software developer even though she lived in the 19th century, well before our digital age. She was a British Countess who was also an accomplished mathematician. Lovelace was the first person to envision how an algorithm could help a computer perform more advanced functions than just calculating numerical sums. She worked closely with Charles Babbage, who conceived of some of the first mechanical machines that were designed to perform calculations via punch cards. Alan Turing Alan Turing is another person who lived before our modern age. A paper he wrote in was one of the key elements that led to the development of the entire field of software development and engineering in the first place. It is estimated that his important working the field of computing helped to shorten World War II by helping to solve encrypted messages. Most of his other innovations were in the field of hardware, including increasing the speed of massive early computers, and the first random-access storage in computers that allowed them to both store programs and information. Software Engineering Industry The software engineering industry is one of the fastest growing fields today, and the need just keeps on growing. IBM relates this boom to the Industrial Revolution that occurred at the turn of the

century with factories and automated processes. Software engineers are well compensated for their expertise. Many tech companies provide their employees with other benefits such as flexible schedules, the ability to work from home, in-office snacks and diversions like video games or a ping pong table. These days the " internet of things " connects all of our devices to the world-wide-web, the internet. There include smart refrigerators, smart TVs and even smart coffee makers, and any other device in your home that can be integrated with your Amazon Alexa or Google Home Assistant! Software engineers are needed to create the programs that make these products work with your smartphone or computer. They work closely with internet experts to make sure everything works without a glitch. While there are many other positions that tech companies need like data analysts, salespeople, operation managers, PMs and technical writers, nothing would get done without the software engineers or developers who make the products work the way they should! First of all, it is often possible for them to work remotely from home because their work only needs a computer and an internet connection. They can work from any location, being the sandy beaches of Guam or the rocky mountains of Nepal. Additionally, it is possible to get short-term contracts as a software engineer or developer. Thus, there are some people who only work part of the year and spend the rest of the time traveling or pursuing another passion or projects. The main benefit of this type of position Software Consultancy is that there is a lot of room for growth and opportunity in this industry. You can even take on multiple projects at the same time, if you really want to work some extra hours and make much more money. Many companies give employees a certain percentage of their working time to work on their own projects. Who knows, you might create the next billion-dollar product in your spare time! What is the Average Software Engineer Salary? As you can see, becoming a software engineer will be great for your bank account. Especially if you are considering becoming self-employed, therefore taking on multiple projects at a time.

Chapter 5 : My first PC build for Software Development - [Solved] - Components

Starting to work with any software, the user should have settings with maximum privacy. In case the user does not make any changes in the settings, the protection level should remain unchanged. The application should not require any actions from users to obtain maximum level of personal data protection.

The traditional programmer career track will only take you so far. There are ways to bust through and increase your salaryâ€”and the enjoyment and satisfaction you get from your work. This book will show you how to become a specialist who can command above-market wages, how building a name for yourself can make opportunities come to you, and how to decide whether consulting or entrepreneurship are paths you should pursue. John went on to become a highly paid consultant in test automation and Agile methodology, and the 55 courses he published with PluralSight makes him one of the most prolific online trainers in the field of software development. John effectively retired at 33 and moved to San Diego with his wife and daughter. Today he runs the hugely popular Simple Programmer blog and YouTube channel, where he helps more than 1. Especially since I am less than a week away from starting a new job after deciding that my current development job of nearly 14 years was no longer enough for me. Reading the section on advancing your career has been very near and dear to my current situation and my view on a lot of things. It has actually helped validate some of my recent decisions and laid some of my doubts to rest The timing for this was perfect! And preferably a first read for anyone seeking to enter the software industry. Or decide if they want to learn it. The insights in this book would have helped me to avoid years of wasted time with an unfocused education and an unfocused career. He was impressed and invited me to his company to learn coding. I am getting taught about different languages and stacks. It is hard as hell but with the learning process I read about in this book I crush the obstacles. I no longer prepare for learning for weeks and months. I developed a learning pace and it is beyond what I thought I was capable of. All of his advice is dead on. The chapter on dressing for success in particular provides unique and valuable advice. His concept of "being a contradiction" makes so much sense. The entire book is filled with great advice but that one chapter is the advice that I need to follow to take my career to the next level. The parts on how to brand yourself and how focus on a niche that suits you best made my career advance faster and made me receive cool offers in my inbox on a near monthly basis. Honest, no BS advice. Easy-to-understand, accessible, and comedic writing style. It gave me a more vivid picture of the software industry, made me rethink some of my choices, but most importantly, it filled in some gaps, instantly making me feel more experienced and knowledgeable. When I saw chapter titles like "Dealing with Coworkers" or "How to Dress" I thought the book would only be for beginners. After reading it I changed my mind. Not only is this book great for beginners, it is also great for experienced developers. Even if you have tons of experience, it will "tell" you what you know in a way that will make you want to act. I think the advice in this book is guaranteed to make you a better developer and more importantly, a better person. Things like "the boy scout rule" and the effective debugging would have climbed me to a higher ground. In this book, John points you even if you are an experienced developer to go to the moon.

Chapter 6 : What is Agile Software Development? [Quick Guide]

This guide will show you whether the offshore software development is a good choice for you, as well as how to make fewer mistakes when you choose an offshore software development company and cooperate with it.

His job is to make sure the project makes business sense. Scrum Master Then there is the Scrum Master. He makes sure the Team follows the work methodology and he helps Team members solve problems that could otherwise block progress and this brings us to the actual Team, a group of specialists in diverse fields, working together to achieve a common goal. These guys are able to organize their own work and take responsibility for the project. The Product Owner makes sure the clients are heard and understood. The Scrum Master, on the other hand, makes sure this understanding translates into tangible effects. How is that achieved? Everything starts with user stories. Think of this as a feature wishlist from the client. Backlog items can be added or deleted on the go, and the less urgent ones may be only loosely defined at first, so the Product Backlog is more than a static list! The Product Owner and the Team use the Product Backlog to select the highest priority requirements with time limitation in mind. The resultant time units are called Sprints, and the tasks selected for the next. The sprint from the Product Backlog makes up the Sprint Backlog. These tasks need to be clearly defined and need to have a measurable goal. A Sprint A Sprint is a joint effort by the whole Team to finish a negotiated set of functionalities in a specific timeframe, usually 2 to 4 weeks. The Scrum Master is there to facilitate the whole process. Now for the actual Scrum Cycles. The Team gets down to business with a goal-oriented mindset. The Team members expect to complete all the Sprint tasks on time. For everyone to stay on the ball, the team holds stand-up meetings, time-boxed to 15 minutes, to talk about what each Team member has done and will do, and about things that may be holding them back. Thanks to the burn down chart, the Team can control the current Sprint progress whenever necessary. Slowly but steadily, the daily iterations contribute to the general Sprint output And this output is something that can be potentially shipped to the end users. Product Backlog, Sprint Backlog, Scrum Cycles and the shippable end product â€” these four elements provide a framework that makes managing IT projects efficient and effective. Provided, of course, that you know how to use certain tools. Artifacts and Tools What is the Burndown Chart? The Burndown Chart allows you to track the current Sprint progress by graphically representing how much work is left to do versus available time. Additional indicators, often presented on the same chart, let you instantly spot how much work is left to do, how much effort went into achieving the current effect and how well your Team estimated the amount of work involved. This versatile tool can be used to visualize the whole project! Agile Wall Agile Wall, in turn, is a simple solution for visualizing the current status of all the items in the Sprint. Just pin tasks written down on post-it notes to the right section of the Agile Wall to let other team members know what the current status of your tasks is. The simplicity of this and other Scrum tools makes getting things done much easier! A completed task is one that has been fully implemented, documented and tested. The end result is functioning and can be presented to the client.

Chapter 7 : SAFECode updates its guide on best secure software development practices - SD Times

A top-level software development executive can make a whole lot more than that. $, is the top salary for a software development executive, and the median for that position is around $,

Related Links The ability to create and respond to change in order to succeed in an uncertain and turbulent environment. Agile methodologies focus on rapid and frequent deliverables of partial solutions that can be evaluated and used to determine next steps. Successful agile teams can produce higher-quality software better meeting user needs quicker and at a lower cost. According to Agile Alliance: Visual Paradigm supports a powerful agile toolset that covers user story mapping, affinity estimation, sprint management, etc. Free Download Manifesto for Agile Software Development The motivated individuals with similar point of view in around , brought out a set of ideas and values that are known as the Agile Manifesto. There are four important aspects that make up the Agile Manifesto as listed below: Through this work we have come to value: Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan Please note items on the left have greater value on them than items on the right; It should be read as left is important "over" right and should not to be replaced with "instead". Principles of the Agile Manifesto: The core Agile Manifesto values are captured in twelve principles: Customer satisfaction through early and continuous software delivery - Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases. Accommodate changing requirements throughout the development process - The ability to avoid delays when a requirement or feature request changes. Frequent delivery of working software - Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software. Collaboration between the business stakeholders and developers throughout the project - Better decisions are made when the business and technical team are aligned. Support, trust, and motivate the people involved - Motivated teams are more likely to deliver their best work than unhappy teams. Enable face-to-face interactions - Communication is more successful when development teams are co-located. Working software is the primary measure of progress - Delivering functional software to the customer is the ultimate factor that measures progress. Agile processes to support a consistent development pace - Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release. Attention to technical detail and design enhances agility - The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change. Simplicity - Develop just enough to get the job done for right now. Self-organizing teams encourage great architectures, requirements, and designs - Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products. Regular reflections on how to become more effective - Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently. The different Agile methodologies provide prescriptive practices to put these values and principles into action and can be visualized by a interesting mindmap. An Agile Toolkit for Software Development Managers" published Agile vs Traditional Development Typically in plan driven model, scope is fixed and the cost and schedule are variables. Many large scale software projects were and continued to be implemented this way. In many instances, when a particular scope is desired within a giving time-frame fixing the schedule , plan driven projects add resources. As a matter of fact, if resources are added late on a software project, it actually has an adverse effect. With Agile software development, the triangle gets inverted - where cost and schedules are fixed and the scope is variable. Traditional Waterfall Scope is fixed, cost and schedules are variables Systems are fully specifiable, predictable, and can be built through extensive planning. Agile Cost and Schedules are fixed, scope is variable Adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change. An Agile Development Framework Scrum is one of the most popular frameworks for implementing agile. So popular, in fact, that many people think scrum and agile are the same thing. Both Agile and Scrum are terms used in

project management. The Agile methodology employs incremental and iterative work beats that are also called sprints. Scrum, on the other hand is the type of agile approach that is used in software development. In other words, many frameworks can be used to implement agile, such as Kanban for example, but scrum has a unique flavor because of the commitment to short iterations of work. These sessions include key members of the project team including the client, project manager, designer, developer, and product owner to ensure a shared understanding across the entire team. The Product Backlog Product backlog is the master list of things that we want to build into the product. During Vision step, the team works together to create a high level Product Backlog, a wish list of all the features that would be useful to the client and their users. The product owner works with the client to prioritize these features, determining the order in which the features are elaborated, developed, tested, and delivered. By allowing the client to determine priority, the team stays focused on delivering the highest value features before moving on to lower value features. These are fixed durations of weeks depending on project size and duration , each delivering a subset of the overall product backlog. Continuing the Cycle Additional Sprints are conducted as needed to deliver additional features and incorporate feedback from previous iterations, reviews, and user beta testing. Each successive Sprint is both Iterative, providing improvements to work completed in previous sprints; and Incremental, adding new features to the system. How does Scrum Framework Work? Scrum is an agile framework most commonly used for product development, especially software development. Scrum is a project management framework that is applicable to many different type of projects with tough deadlines, complex requirements and a degree of uniqueness. In Scrum, projects move forward via a series of iterations called sprints. Each sprint is typically two to four weeks long. A typical scrum process can be summarized as follows: Ideal team size is 5 to 9 people Team has everything it needs to deliver an increment of working product Very clear role and responsibility delineation: A typical Scrum backlog comprises the following different types of items:

Chapter 8 : Guide to Software Development: Designing and Managing the Life Cycle by Arthur M. Langer

As software delivery has moved from multiyear releases to daily updates, software-development practices have evolved to focus on building high-quality software at an increasingly fast pace. DevOps. DevOps is the next frontier in the evolution toward increasingly agile development methodologies.

The essential guide to software containers for application development David Linthicum , Chief Cloud Strategy Officer, Deloitte Consulting Containers are exploding onto the application development scene, especially when it comes to cloud computing. This is largely because portability has been a big chasm in this area, given the proprietary nature of some public clouds, and this technology abstracts applications into virtual containers that can be moved from cloud to cloud. The architecture of containers is the other major benefit. Breaking applications up this way offers the ability to place them on different physical and virtual machines, in the cloud or not. This flexibility offers more advantages around workload management and provides the ability to easily make fault-tolerant systems. Also, with the use of clustering, scheduling, and orchestration technology, developers can ensure that applications that exist inside of containers can scale and are resilient. These tools can manage groups of containers using a well-defined container management layer that provides these capabilities. This provides support directly from existing enterprise tools and technology. Numerous well-funded startups are appearing as well, with innovative solutions to make container development much more interesting and productive. What does all of this mean to software engineers? Testing in the Agile Era: Docker really started the container movement. Companies such as CoreOS have their own container standard called Rocket, and many standards and products are being built around these technologies. This kind of approach is nothing newâ€”containers have been used for years as an approach to componentize whole systems, abstracting them from the physical platform, allowing you to move them around from platform to platform or cloud to cloud. Docker extends a common container format called Linux Containers LXC , with a high-level API that provides a lightweight virtualization solution that runs processes in isolation. The use of this technology is rather exciting, considering it solves an obvious and expansive problem: How to provide true application portability among cloud platforms? While workloads can certainly be placed in virtual machines, the use of containers is a much better approach, and should have a higher chance of success as cloud computing moves from simple to complex architectures. The ability to provide lightweight platform abstraction within the Docker container, without using virtualization, is much more efficient for creating workload bundles that are transportable between clouds. In many cases, virtualization is just too cumbersome for workload migration. Thus, containers provide a real foundation for moving workloads around within hybrid or multi-cloud environments without having to alter much or any of the application. Reduce complexity through container abstractions. Leverage automation to maximize portability. Automation replaced manual scripting. Provide better security and governance, external to the containers. Security and governance services are platform-specific, not application-specific. Placing security and governance services outside of the container significantly reduces complexity. Provide enhanced distributed computing capabilities. This is due to the fact that an application can be divided into many domains, all residing within containers. The portability aspect of containers means they can execute on a number of different cloud platforms. This allows engineers to pick and choose the platforms that they run on, based upon cost and performance efficiencies. Provide automation services that leverage policy-based optimization. There needs to be an automation layer that can locate the best platform to execute on, and auto-migrate to that platform. At the same time, it must automatically deal with needed configuration changes. How to scale container-based applications Most who look to make containers scale take one of two basic approaches. The first approach is to create a custom system to manage the containers. This means a one-off system that you build to automatically launch new container instances as needed to handle an increasing processing load. But remember that if you build it, you own it. As with many DIY approaches, the maintenance will become labor- and cost-intensive. The second approach is to leverage one of the container orchestration, scheduling, and clustering technologies that will provide the basic mechanisms to enable scalability. This is normally the better of the two options. There are a

few choices out there for the second approach: Kubernetes can schedule any number of container replicas across a group of node instances. This container replication and distribution trick is typically enough to make most large container-based applications scale as needed. This is pretty much the same approach to scaling containers that the other tools take. Its YAML-based blueprints let developers describe complex topologies, including the infrastructure, middleware tier, and app layers. Finally, the newest tool, Docker Swarm provides clustering, scheduling, and integration capabilities. Obviously, Swarm is designed to compete with Kubernetes, which has a larger market share. I would suggest a proof of concept with each technology, using real-world workloads. Best practices continue to emerge around scaling containers, including: Devote time to the architecture of your container-based applications. Most scaling issues are traced back to poor designs, not poor technology. Use automated testing tools to simulate the workloads and massive amounts of data for testing. Consider your own requirements. What works for other large companies may not be right for your container-based applications. They have to scale as well. I suspect that scaling containers will be a bit tricky until more is understood about how containers behave at scale. However, installing Docker on a Mac or Windows will require a few more steps. Just follow the appropriate OS instructions. The next step is to attempt to run a Dockerized application. Docker has compiled a public registry of applications available as Docker images, and this community provides many jumping off points for building and running your own container-based applications. Once Docker is installed and online, run a Docker application image by entering: The Docker container is simply an instance of a Docker image, much like applications are instances of executables that exist in memory. So, you can launch multiple isolated instances of the app as containers on a single host. By adding "-rm" to the command, as done above, Docker is instructed to remove the container from memory once it completes its task. This has the effect of removing any changes to the local environment that the application may have made but keeps the cached image. Building a Docker image for an application requires starting with a base image for the core OS, which runs in Docker. Install and configure the necessary tools, and then use the Docker "commit" command to save the container as an image. Finally, push it to the public Docker image registry or keep it private. Another way to create an image is to note the steps required to build the image in a well-formed Dockerfile file. This automates the process of installing and configuring the application , creating a repeatable process. As with any development process, there are more details that you need to understand to master building and running Docker images and containers. In many respects, the success of containers and Docker has been around the ease of development. As the standard and product progresses, things will likely get even easier. A container in every shop The tendency is to think that new ways of building systems will be the way that we build systems for years to come. Containers deliver a standard, useful enabling technology and provide a path to application architecture that offers both managed distribution and service orientation. The viability and versatility of this technology will be something that we continue to explore and exploit over the next several years. Count on the fact that a few mistakes will be made, but the overall impact of containers is a foregone conclusion.