AKAMAI SOFTWARE ENGINEER CAREER MAP

September 2015

Project Team: David Gervais, Harish Kammanahalli, Ken Iwamoto, Mike Szydlo, Larry Underhill, Meena Kamath, Mark Van Horn, Oamr Orqueda, David Duff, Alex Martineau

OVERVIEW

This document presents a matrix of Knowledge, Skills, and Abilities that we expect software engineers to demonstrate at different stages in their career at Akamai. The Knowledge section describes what you need to know to be a good software engineer at Akamai. The Skills section describes the situations and the manner in which you apply your knowledge to solve software engineering problems. The Abilities section describes the general, non-technical skills that distinguish effective software engineers at Akamai. The intended audience of this document consists of the managers and software engineers in the Engineering organizations of Akamai.

There are seven career stages (above Associate Engineer) proposed and each is represented by a column in the matrix. By reading vertically, you can narrow in on the set of expectations that define a single stage in the career of a software engineer. By reading horizontally, you can narrow in on the Knowledge or on a single Skill or Ability across all career stages.

As you read horizontally, note that subsequent stages build on top of previous stages so Knowledge, Skills, and Abilities are cumulative. If a particular aspect of an earlier stage is not explicitly mentioned later, it is still relevant in the later stage, unless obviously superseded by a description of a more complex behavior. In a few cases, you will find that the description of a Skill or Ability is blank for a particular career stage. This means that there is no additional growth expected in that stage beyond what has already been described in previous stages.

It is important to note that the distribution of engineers in each level will depend first and foremost on the Knowledge, Skills, and Abilities exhibited by individuals. For the highest two levels of the matrix, Engineering management will form a committee to evaluate the qualifications of candidates proposed for promotion to those levels.

AKAMAI CAREER MAP OVERVIEW SOFTWARE ENGINEER

KNOWLEDGE	Akamai	Technology/Specialty	Product	Customer															
SKILLS	Scoping	Designing	Programming	Testing	Optimization	Debugging													
KEY SUCCESS FACTORS	Peer Relationships	Written Communication	Problem Solving	Time Management	Self Development	Listening	Functional/Technical Skills	Creativity	Learning on the Fly	Drive for Results	Dealing with Ambiguity (Senior)	Organizational Agility (Senior)	Decision Quality (Senior)	Priority Setting (Senior)	Customer Focus (Senior)	Intellectual Horsepower (Principal)	Presentation Skills (Principal)	Strategic Agility (Principal)	Interpersonal Savvy (Principal)

TYPE	SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Position Overview	A Software Engineer at Akamai is an entry level or early career developer who has had formal training or equivalent experience in software development and the aspects of Computer Science relevant to Akamai. A Software Engineer is expected to demonstrate initiative, to learn quickly from Senior Engineers, and to collaborate fully as a member of the development team	A Software Engineer II at Akamai is a developer who has had experience in developing software and has reasonable understanding of building production quality software. A Software Engineer II should be capable of implementing reliable production software and should be able to follow software development practices.	A Senior Software Engineer at Akamai has had professional or academic experience developing complex software, has acquired a solid understanding of how reliable production software is built, and is well versed in the application of Computer Science to solve practical problems.	A Senior Software Engineer II at Akamai has had significant professional experience developing complex and highly reliable software in a production environment. They demonstrate leadership in the design and implementation of complex software systems within their team's area of expertise. They can be entrusted with organizing and communicating effectively with various project stakeholders and can drive the scoping and development of projects independently and across another functional development teams. Senior Software II Engineers are highly regarded within their team as experts on the products and technologies they develop and support.	A Principal Software Engineer at Akamai is an engineer that has extensive professional experience building production quality software and has demonstrated leadership in the design and implementation of complex projects. In addition, they have distinguished themselves from peers by the quality of their work and the breadth and depth of their knowledge of Akamai. Principal Software Engineers are the key leaders in each area of expertise in Engineering.	A Senior Principal Software Engineer at Akamai is an extraordinary engineer at Akamai and at the very top of the industry. They have accumulated a record of accomplishment across several groups of Akamai Engineering, who consistently demonstrates quality, innovation, and a deep understanding of complex systems and the craft of software development.	A Distinguished Software Engineer at Akamai is an engineer whose accomplishments are recognized beyond Akamai, in the broader industry. A Distinguished Software Engineer is also someone who has the experience, presence, and strategic insight to shape Akamai's business and to represent Akamai externally. Distinguished Engineers are rare, with only a handful achieving this title.

TYPE	SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Knowledge *What you need to know to be a Software Engineer at Akamai.*	Knowledge of programming and programming languages necessary to perform the job. Familiar with a variety of common data structures and algorithms. Understands programming paradigms like Object Oriented Programing. Understands basic database concepts. Understands parallelism and concurrency in software in at least one of these environments: multi-process, multi-task, multi-thread, or concurrent transactions. Understands basics of software verification aspects like: unit testing, system testing. Aware of current software development processes. Understands basic operating systems concepts. Aware of various configurations and deployment environment.	Understands how to compare the relative performance of algorithms, using concepts such as worst case, average running times, or Big O notation. Understands component/module level implementation details. Understands different approaches to testing software to verify correct functionality and to characterize performance. Understands how to use application development frameworks, APIs and a variety of common software implementation techniques. Understands advanced features of programming languages and is comfortable in using relevant tools and techniques to aide software implementation. Adopts software development processes. Uses tools effectively to drive communication and collaboration. Understands how customers use Akamai's product portfolio and broadly, how Engineering systems implement these products.	Knowledge of programming languages and APIs necessary to perform the job. Understands the pros and cons of application development frameworks and how to best use these to build good code quickly. Knowledge of both standard industry techniques for designing and implementing high-quality software and practices unique to Akamai's widely distributed, highly available software environment. Broad knowledge of major subsystems at Akamai, including network messaging protocols, dependencies across subsystems, failure modes, and configuration schemes. Specific and deep knowledge, within own area of responsibility, on how components interact to accomplish a broad task. For example, a software engineer in Load Balancing in Mapping needs to understand the cluster of components that contribute to load balancing decisions. Understands in detail how own software serves Akamai customers; in general, how Akamai solves complex customer business problems; and the importance of high quality, in development and systems operations, to the customer experience. Awareness of how consumers utilize their functionality.	Knowledge of programming languages, APIs, and relevant software engineering principles necessary to perform the job at a high level without requiring peer oversight to ensure correctness. Makes sound judgments when selecting development frameworks and can clearly articulate the design decisions and motivations in their selections along with how to build, develop, and support quality software efficiently. Understands and can quantify the impacts of complex code changes, design optimizations, or system performance trade-off decisions both within their immediate application and its impact to the overall software system. Understands how their components fit within the Akamai landscape and can diagnose complex interactions they have across multiple cross-domain components throughout the Platform. Interacts with users and customers of their software products and can turn those interactions into quantifiable, detailed, and workable solutions to solve customer problems effectively.	Expert programmer who deeply understands programming languages and their implementation. For example, a C++ programmer understands assembly and a Java programmer understands the JVM and bytecode. Understands the performance impact on user code of compiler choices, runtime/library implementations, the file system, the network stack, and the virtual memory system. Understands the source code of critical libraries and application development frameworks in use within their area. Understands enough of the components running across systems on relevant Akamai networks to track an interaction that traverses through major sub-systems. Expert at leveraging or enhancing existing Akamai tools and infrastructure. Understands details of customers' business problems in several market spaces and how Akamai's technology can help.	Expert software designer who can be trusted to take the most complex projects through the Akamai cycle of design, implementation, and operationalization. Understands the evolution of industry software development practices and their impact on Akamai's engineering efforts. Immersed in Akamai's industry and aware of industry trends that affect Akamai's business and its customers. .	Displays a deep understanding of Akamai business and technology and the implementation of Internet-scale systems.

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Skills *Broad skills that distinguish effective Software Engineers at Akamai*	Scoping and Designing	Able to design and implement a well scoped feature independently or with reasonable guidance from a more experienced developer/lead or manager. Actively participates in sizing and estimating development tasks. Able to manage various development environments with guidance.	Designs features within a component using solid principles and following Akamai architecture group design guidelines: http://docs.akamai.com/archgrp. Designs deal gracefully with local phenomena -- such as system limits, thresholds, database options, unexpected inputs, and unknown errors – and restart cleanly in the case of a roll. Designs components, often with the help of senior members in the team, with focus on reliability and balances often competing objectives: fulfilling customer requirements, efficiency and performance, ease of use, consistency in APIs and ease of implementation. Writes effective design documents that are comprehensive and clear. Designs with an understanding of how various Akamai systems work/interact/integrate (like portal, Oracle, Siebel, SFDC networks like freeflow, delivery and DDC) and groups interact towards designing a feature together. Designs for configurability and safe roll out/ roll back of features.	Gathers relevant information to transform an imprecise request into a clear specification that can guide successful development. Designs encompass a substantial code base and follow Akamai architecture group design guidelines. Ensures that designs focus on reliability and balance three often-conflicting objectives: fulfilling customer requirements, efficiency and performance, and ease of implementation. Designs deal gracefully with large-scale systems issues, such as changes in network topology, coherency, consistency, partial failures, oscillations, or resource scheduling in the face of unexpected capacity constraints. Avoids over-engineering by assessing trade- offs across the entire design spectrum, including speed vs. quality, rigor vs. agility, time-to-market vs. reliability, and full functionality at once vs. incremental releases. Incorporates an understanding of Akamai's costs of doing business – including network and operational – into designs. Designs include thorough provisions for software rollout -- including version skew across the network and impact of new features on the installed user base -- and operationalization – such as ongoing maintenance, controlled enabling/disabling of features, NOCC alerts, long term monitoring – all of which are key elements to the success of a system at Akamai. Writes design documents that describe the full context of a solution, including business and technical motivations and risk assessments. Provides valuable feedback to peers during design reviews.	Capable of collecting multiple feature requests with differing and sometimes conflicting requirements and can propose, socialize, and execute on a comprehensive design and technical solution to meet their business needs. Ensures business needs drive scope and design of technical solutions. Considers and evaluates multiple technical solutions. Articulates the pros and cons of different solutions and makes recommendations. Able to lead a proposal through Arch Group review or similar peer group/process. Able to exercise sound judgment and push back on peers or management constructively to offer technical guidance and alternatives on poorly defined or ill-scoped projects and requirements. Understands impact of design decisions and feature requests beyond just their specific component, application, or immediate deliverable; capable of placing design and implementation details in the context of Akamai's overall business needs and can coordinate development across functional teams with little to no management or technical oversight. Able to effectively drive and execute on a team's roadmap; can dissect high-level requirements into quantifiable and deliverable software products. Demonstrates the ability to coordinate large complex tasks across the team and can drive and lead development efforts beyond their individual code contributions. Provides mentorship for other engineers; Acts as a facilitator when communicating across groups and offers insightful design and technical feedback and solutions that leverage both industry-standard technical knowledge and the specifics of Akamai's infrastructure, tools, and frameworks. Capable of working with management and architects to shape business needs and technical solutions to those business needs by gathering support across the company and strengthening the technical merit of their proposals.	Works proactively with requestors to guide the process of clearly defining the business problem, refining a request, and articulating a full technical solution. Anticipates the needs of the market with specifications for products and features that can keep Akamai on the forefront of technology. Designs entire subsystems or groups of components following Akamai architecture group design guidelines. Incorporates knowledge of Akamai's organizational structure into designs. For example, a design that spreads responsibility for a system across many groups is likely to be more brittle than one requiring fewer groups to collaborate. Writes design documents with great clarity and completeness such that they are used regularly by peers and tech writers as references, and by others across Akamai for general insight. Mentors others to ensure that their specs, designs and implementations take into consideration the full technical and business aspects of a request, with functionality that fits into the overall approach for a system or component.	Designs network-scale systems following Akamai architecture group design guidelines. Incorporates knowledge of 3-5 year technology trends to ensure that Akamai systems scale, perform, and meet customer needs over the long term.	Designs Internet scale systems or architectural frameworks to support new businesses at Akamai following Akamai architecture group design guidelines. Writes design documents suitable for audiences outside of Akamai, including the broader industry, standards bodies, and customers.

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Skills (cont.) *Broad skills that distinguish effective Software Engineers at Akamai*	Programming	Strives to produce the simplest, most readable code possible, including good abstractions, well-defined interfaces, meaningful variable names and a clean decomposition into files in the source control system. Follows the Akamai Software Development Lifecycle and knows how to use tools relevant to releases, such as Bart, Bugzilla/Jira, Akamake, the Build System, and Netdeploy. Identifies and implements error checking, exception handling and logging. Creates documents to help maintainability, readability and code handoff. Develops code, fixes with proper comments, test scenarios..	Delivers complex software functions with good quality, requiring few patches. Simple software changes rarely require patches. Avoids re-implementing existing functionality by building on available libraries/APIs, using common design patterns, and leveraging common Akamai tools. Improves code handoff to stakeholders using appropriate tools and techniques to convey complex design or use case scenarios. Implements substantial error checking in own code, including handling the failure modes of libraries or components on which that code depends. Implements components so that debugging, error checking and sustenance can be managed by other groups (for example, Platform Operations or GSS) using common Akamai tools like MapNOCC and ADMS. Understands and demonstrates capabilities to review component level code, effort estimation. Plays active role in SDLC processes by providing adequate and timely status updates. Writes clear and concise documents for the component implementations.	Chooses appropriate programming languages and application development frameworks for each development project. Builds libraries containing common code, growing them over time and sharing them with peers. Refactors software components judiciously, balancing code cleanliness and maintainability with business objectives, such as time-to- market. Develops code that is consistently fault tolerant and secure by coding defensively and by using the wide variety of practices and mechanisms available at Akamai, including message safety checks, KMI, crash rejection, leader election, health checks, and alerts. Makes effective judgments about when to extend existing systems, build new software internally, buy third-party code, or use open-source, including the licensing impacts of using code from outside of Akamai. When relying on code from outside of Akamai, incorporates operational aspects of that code such as patch cycles into the implementation and operation of the system. Able to dive in quickly into unfamiliar code to assist others and to make improvements, including code for third party frameworks or libraries. Updates code quickly when required by time-critical scope changes. Recognizes that peer code reviews are critical to software quality, and takes the initiative to be an active participant in the code reviews of others.	Understands low-level interactions, implementations, and implications of using various program languages, system calls, and relevant application frameworks or libraries for each development project. Capable of recommending and driving code changes or feature enhancements in underlying frameworks or libraries relevant to their components, but owned by other teams or independent third-parties, with the end-result of improving the overall quality of Akamai software and systems. Understands and communicates when a re-write or large scale factoring is prudent to advance items on a roadmap. Mentors other Engineers by providing feedback, design advice, comprehensive code reviews, or domain-specific coaching relevant to their areas of expertise. Able to assess how business decisions will affect a codeline and can balance time, resources, and effort appropriately.	Can be counted on to write production quality code in the first version of a system intended for customer use. Organizes large bodies of code to simplify maintenance and to streamline compilation. Implements code so that incremental performance improvements do not require extensive rework. Recognizes when large-scale refactoring is necessary for the evolution of a subsystem, advocates for the effort, and articulates well how the benefits surpass the cost. Takes a leading role in code reviews, of own code or others' code, to ensure that systems are evaluated not in isolation but in the broader context in which they will operate in production. Invests time in mentoring other engineers in all aspects of software development. Drives for change to development conventions and processes to improve the efficacy of Akamai development while maintaining high quality.	Positively impacts any codeline at any time. Identifies new methods to build and deploy reliable software at Akamai and advocates for processes that can improve Engineering-wide product quality.	

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Skills (cont.) *Broad skills that distinguish effective Software Engineers at Akamai*	Testing	Develops recommended test cases, and implements and performs unit tests consistently. Diligent about not introducing regressions in own code.	Augments code with hooks or instrumentation for diagnostics, monitoring and debugging. Collaborates closely with QA to produce clear test plans that include negative test-cases, validations, identifies corner-cases, develops recommended test cases, and performs unit tests consistently. Rarely introduces regressions in own code.	Tests sufficiently to avoid simple, preventable bugs in new code and approaches zero on regressions in existing code. Develops thorough and largely automated test suites for functional and regression testing, including negative test cases to ensure that error checks work properly. Partners with QA, writes detailed testing plans if appropriate, deployment plans and procedures for Operations, and maintenance plans for fellow Engineers. When working as part of a team, advocates diligence in testing to peers and focuses on finding bugs in own code before these affect others' code. Assesses the severity of bugs found during testing and makes good recommendations to Managers or more experienced Engineers about what needs to get fixed now and what can wait until later.	Demonstrates sound judgment in determining the level of risk associated with rolling-out new, business critical software features; has mitigation plans in place and capable of quickly reacting when bugs, performance problems, or test failures are encountered. Creates tested, quality, and comprehensive test harnesses that provide significant code coverage across their components. Advise and partner with QA team and other groups to update regression and functionalized tests.	Demonstrates sound judgment when determining that large software projects have achieved production quality. Supports that judgment with objective data, such as test failures, open bug severities, test coverage percentages, regression rates, bug open/close rates, or changes past release milestones. Bugs that appear in own code are the result of truly complex software interactions and not the result of sloppiness or improper testing. Leads, advises, or cooperates with QA on creating test infrastructure and automation to replicate the complex environment Akamai software faces in production, including testing of multi-component or network-scale systems.		

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Skills (cont.) *Broad skills that distinguish effective System Software Engineers at Akamai*	Optimization	Design and implement code with efficiency and time in mind. Able to quantify code performance and identify potential bottlenecks.	Proposes and implements local performance improvements: to one or several functions or within the scope of a software module. Uses empirical methods and tools, including profilers, static/dynamic analyzers, database explain plans, and debuggers, to characterize software performance. Avoids basic performance pitfalls in parallel or concurrent code, such as lock contention or thread-safety violations.	Identifies opportunities for optimization across an entire program, beyond a function or module. Uses good judgment to set down performance targets for software, focusing on critical use cases that must be fast, and avoids the trap of premature optimization in cases where software is performing within spec. Establishes procedures to track the long term performance evolution of a software system, so that performance degradations don't result in unexpected surprises.	Collects performance metrics from deployed systems to help inform design decisions and optimizations on the product line in future revisions. Capable of applying mathematical rigor in predicting deployed performance, scaling enhancements, or system availability at a large scale; Able to turn this analysis into quantifiable and measureable improvements in future revisions. Optimizes parallel/concurrent code and develops frameworks such as thread-safe libraries, for others to write safe, efficient programs.	Proposes global optimizations, including across components, based on algorithmic sophistication and large system experience. Finds OS-dependent performance problems using system level profiling tools. Consulted as a resource when others have performance problems in their code.	Can untangle the most complex performance interactions across the entire software/hardware stack: user code, runtime systems, virtual machines, OS, devices and microprocessor arch.	

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Skills (cont.) *Broad skills that distinguish effective Software Engineers at Akamai*	Debugging	Finds deterministic logic errors in own code with little assistance. Uses basic debugging tools and techniques, such as debuggers and leak checkers, to speed up finding the causes of bugs.	Debugs basic concurrent/parallel code effectively, and understands how to manage non-determinism. Develops tools for efficient debugging on various platforms. Able to debug system level behaviors related to their component.	Demonstrates a systematic and tenacious approach to finding bugs, for instance, by developing models of correct system behavior, by developing test cases that eliminate possible root causes, and by reproducing errors whenever possible. Shows ingenuity when debugging problems in production systems (i.e. in the field), where tools and techniques are more limited than in the lab. Reliably diagnoses bugs in complex concurrent/parallel code. Helps others debug their code.	Can diagnose problems across the software and network stack, OS, libraries, and other relevant subsystems; Able to identify reproducible test cases and work with SMEs to resolve problems effectively and efficiently. Demonstrates a calm, focused approach to finding the source of problems in production systems, especially in high- pressure incident situations. Develops debugging tools/frameworks useful for specific use cases or advises QA on what tools to build.	Can trace bugs across the entire software stack, including into the OS, libraries, virtual machine, and compiler if necessary. Demonstrates a calm, focused approach to finding the source of problems in production systems, especially in high-pressure incident situations. Develops debugging tools/frameworks useful to others or advises QA on what tools to build to help overall productivity.	Trusted as a resource for the most difficult debugging problems, across groups and possibly unfamiliar code bases.	

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Key Success Factors ***Competencies needed to be successful in the role***	Drive for Results	Can be counted on to exceed goals successfully; is constantly and consistently one of the top performers; very bottom-line oriented; steadfastly pushes self and others for results.						
	Problem Solving	Uses rigorous logic and methods to solve difficult problems with effective solutions; probes all fruitful sources for answers; can see hidden problems; is excellent at honest analyses; looks beyond the obvious and doesn't stop at the first answers.						
	Functional/Technical Skills	Has the functional and technical knowledge and skills to do the job at a high level of accomplishment.						
	Creativity	Comes up with a lot of new and unique ideas; easily makes connections among previously unrelated notions; tends to be seen as original and value-added in brainstorming sessions.						
	Time Management	Uses his/her time effectively and efficiently; values time; concentrates his/her efforts on the more important priorities; gets more done in less time than others; can attend to a broader range of activities.						
	Self Development	Is personally committed to and actively works to continuously improve him/herself; understands that different situations and levels may call for different skills and approaches; works to deploy strengths; works on compensating for weaknesses and limits.						
	Listening	Practices attentive and active listening; has the patience to hear people out; can accurately restate the opinions of others even when he/she disagrees.						
	Peer Relationships	Can quickly find common ground and solve problems for the good of all; can represent his/her own interests and yet be fair to other groups; can solve problems with peers with a minimum of noise; is seen as a team player and is cooperative; easily gains trust and support of peers; encourages collaboration; can be candid with peers.						
	Written Communication	Is able to write clearly and succinctly in a variety of communication settings and styles; can get messages across that have the desired effect.						
	Learning on the Fly	Learns quickly when facing new problems; a relentless and versatile learner; open to change; analyzes both successes and failures for clues to improvement; experiments and will try anything to find solutions; enjoys the challenge of unfamiliar tasks; quickly grasps ththe underlying structure						

TYPE		SOFTWARE ENGINEER	SOFTWARE ENGINEER II	SENIOR SOFTWARE ENGINEER	SENIOR II SOFTWARE ENGINEER	PRINCIPAL SOFTWARE ENGINEER	SENIOR PRINCIPAL SOFTWARE ENGINEER	DISTINGUISHED SOFTWARE ENGINEER
Key Success Factors ***Competencies needed to be successful in the role***	Dealing with Ambiguity			Can effectively cope with change; can shift gears comfortably; can decide and act without having the total picture; isn't upset when things are up in the air; doesn't have to finish things before moving on; can comfortably handle risk and uncertainty.				
	Organizational Agility			Knowledge about how organizations work; knows how to get things done both through formal and the informal network; understands the origin and reasoning behind key policies, practices, and procedures; understands the cultures of organizations.				
	Decision Quality			Makes good decisions (without considering how much time it takes) based upon a mixture of analysis, wisdom, experience, and judgment; most of his/her solutions and suggestions turn out to be correct and accurate when judged over time; sought out by others for advice and solutions.				
	Priority Setting			Spends his/her time and the time of others on what's important; quickly zeros in on the critical few and puts the trivial many aside; can quickly sense what will help or hinder accomplishing a goal; eliminates roadblocks; creates focus.				
	Customer Focus			Is dedicated to meeting the expectations and requirements of internal and external customers; gets first-hand customer information and uses it for improvements in products and services; acts with customers in mind; establishes and maintains effectives relationships with customers and gains their trust and respect.				
	Intellectual Horsepower					Is bright and intelligent; deals with concepts and complexity comfortably; described as intellectually sharp, capable, and agile.		
	Presentation Skills					Is effective in a variety of formal presentation settings; one-on-one, small and large groups, with peers, direct reports, and bosses; is effective both inside and outside the organization, on both cool data and hot and controversial topics; commands attention and can manage group process during the presentation; can change tactics midstream when something isn't working.		
	Strategic Agility					Sees ahead clearly; can anticipate future consequences and trends accurately; had broad knowledge and perspective; is future oriented; can articulately paint credible pictures and visions of possibilities and likelihoods; can create competitive and breakthrough strategies and plans.		
	Interpersonal Savvy					Relates to all kinds of people, up, down, and sideways, inside and outside the organization; builds appropriate rapport; builds constructive and effective relationships; uses diplomacy and tact; can diffuse even high-tension situations comfortably.		