Software Engineers and Designers

NOC 2173

Introduction

Software engineers and designers research, design, evaluate, integrate and maintain software applications, technical environments, operating systems, embedded software, information warehouses and telecommunications software. They are employed in information technology consulting firms, information technology research and development firms, and information technology units throughout the private and public sectors, or they may be self-employed.

The most important Essential Skills for Software Engineers and Designers are:

- Digital Technology
- Problem Solving
- Oral Communication

Document Sections

- Reading
- Document Use
- Writing
- Numeracy
- Oral Communication
- Thinking Skills
    - Problem Solving
    - Decision Making
    - Critical Thinking
    - Job Task Planning and Organizing
    - Significant Use of Memory
    - Finding Information
- Working with Others
- Digital Technology
- Continuous Learning
- Notes

A. Reading

Reading

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 5	• read emails from co-workers, colleagues, supervisors and testers about requests for fixing issues that may include technical information and possible solutions. (2)
Most Complex	5	• read discussion threads, and messages on ticketing systems that help to plan out tasks. (2) • read forums for information such as questions from the developer or language specific problems. (3) • read information from websites to research new technology or technical information. (3) • read customer reports about issues and bugs in electronic files and source code to understand problems and how they may have developed. (3) • read contract agreements and service level agreements outlining projected costs, timelines and responsibilities. (3) • read test reports from quality assurance that detail results from software testing to ensure products meet customer expectations. (3) • read proposals and design specification documents to understand project specifications. (3) • read official documentation such as guidelines and standards on languages, design and operating systems (OS) as part of information gathering. (4) • read software development kit (SDK) documentation to learn how to use the software to develop applications for specific devices or operating systems. (4) • read software manuals and textbooks written by subject matter experts to locate specific information, such as troubleshooting or how to program a specific function. These manuals require specialized programming knowledge, and they may synthesize information from several sources. (5)

Reading Summary

The symbols >, >> and >>> are explained in the Use of Symbols section.

	Purpose for Reading			
Type of Text	**To scan for specific information/To locate information**	**To skim for overall meaning, to get the 'gist'**	**To read the full text to understand or to learn**	**To read the full text to critique or to evaluate**
Forms	>>>			
Labels	>>>			
Notes, Letters, Memos	>>>	>>>	>>>	
Manuals, Specifications, Regulations	>>>	>>>	>>>	
Reports, Books, Journals	>>>	>>>	>>>	

B. Document Use

Document Use

Tasks	**Complexity Level**	**Examples**
		Software Engineers and Designers:
Typical	1 to 4	• scan lists identifying the various features to be included in software. (1)
Most Complex	4	• locate work items or job tasks to be completed in spreadsheets. Use spreadsheets for sign off requirements. (2) • refer to data sheets on hardware to understand the product. (2) • use graphical forms for timelines and planning information. (2) • may use flowcharts to outline more complicated algorithms. (2) • enter multiple pieces of information into project tracking software. For example, they enter details about job tasks

		that either need to be completed or have been completed. (3) • locate information in computer files, such as debugging software (.gdb files) or source code. Read programming language. Each language has its own function and codes. (3) • consult and synthesize information from technical documents to develop software applications. They refer to specifications that describe the application. (4)

Document Use Summary

- Read signs, labels or lists
- Complete forms by marking check boxes, recording numerical information or entering words, phrases, sentences or text of a paragraph or more. The list of specific tasks varies depending on what was reported.
- Read completed forms containing check boxes, numerical entries, phrases, addresses, sentences or text of a paragraph or more.
- Read tables, schedules and other table-like text.
- Create tables, schedules or other table-like text.
- Enter information on tables, schedules or other table-like text.
- Obtain specific information from graphs or charts.
- Interpret information on graphs or charts.
- Plot information on graphs (e.g., line, pie, bar).
- Obtain specific information from graphs or charts.
- Construct or draw graphs or charts.
- Obtain information from sketches, pictures or icons (e.g., computer toolbars).

C. Writing

Writing

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 4	• write emails to clients to give project updates, respond to feedback or questions, and inform them of changes. These emails may become part of customer reports. (2)
Most Complex	4	• write emails to software developers or colleagues using technical terminology. Also may write emails to ask colleagues for advice. (2) • write lessons learned for self or colleagues as part of a learning process. (3) • write help files, and user and upgrade guides that explain how to use programs and answer questions users may have. (3) • write software source code or technical comments on coding to update, upgrade, or revise the design of the product. For example, writing in language C++ or JavaScript. (4) • write "read me" files following established format and syntax. For example, read me files include configuration, installation, and operating instructions, and troubleshooting information. (4) • write project documentation that details the order of tasks, problems encountered, and causes of problems. This information is used by the project team, new team members, and the team manager as ongoing documentation of the project. The documentation may include reports to managers. Larger projects involve more paperwork and writing to share with stakeholders. (4) • write test reports, quality assurance reports, design documents, and status reports. For example, quality assurance reports summarize the quality of the software and provide details about the testing techniques and procedures used to test the software. Design documents explain how the software will be designed and its functionality. Status reports detail progress made, problems encountered and how they were solved, and next steps. (4)

Writing Summary

The symbols >, >> and >>> are explained in the Use of Symbols section.

	Purpose for Writing						
Length	**To organize/ to remember**	**To keep a record/to document**	**To inform/ to request information**	**To persuade/ to justify a request**	**To present an analysis or comparison**	**To present an evaluation or critique**	**To entertain**
Text requiring less than one paragraph of new text	>>>	>>>	>>>				
Text rarely requiring more than one paragraph	>>>	>>>	>>>				
Longer text	>>>	>>>	>>>	>>	>>	>>	

D. Numeracy

The symbols >, >> and >>> are explained in the Use of Symbols section.

Numeracy

Tasks	**Complexity Level**	**Examples**
>> Scheduling, Budgeting & Accounting >>> Measurement and Calculation >>> Data Analysis	3 5 4	Software Engineers and Designers: • may develop budgets for projects. They calculate costs such as hours, resources and overhead. For example, they calculate the cost of contractors by multiplying the rate times the number of hours. They determine if more resources are needed for a project, if overtime is needed, or if the project needs to be reduced in scope. (Scheduling, Budgeting &Accounting), (3) • schedule tasks on a daily, weekly and monthly basis for team members and others including production teams and testing groups. They adjust schedules to accommodate unforeseen events and to meet deadlines. (Scheduling,

>> Numerical Estimation	2	Budgeting &Accounting), (3) • apply mathematics, computer science and engineering skills to design, develop and test software applications. For example, they write and use algorithms and logarithmic functions to sort data or solve a problem. They use calculus, linear algebra and discrete mathematics. (Measurement and Calculation), (5) • conduct summary calculations and use statistics and probability to analyze and mine data, and to develop forecasts using statistical analysis software. For example, they use statistical software to calculate arithmetic means for high-level analysis of test data to present to management. (Data Analysis), (4) • make estimations when planning for time and scheduling for a project. (Numerical Estimation), (2)

Math Skills Summary

a. Mathematical Foundations Used

The symbols >, >> and >>> are explained in the Use of Symbols section.

Mathematical Foundations Used

Code	Tasks	Examples
		Number Concepts
>>>	Whole Numbers	Read and write, count, round off, add or subtract, multiply or divide whole numbers. For example, when writing algorithms and logarithms.
>>>	Integers	Read and write, add or subtract, multiply or divide integers. For example, when writing algorithms and logarithms.
		Patterns and Relations
>>>	Equations and Formulae	Solve problems by constructing and solving equations with one unknown; use formulae by inserting quantities for variables and solving; write, simplify and solve two variable algebraic problems; write, simplify and solve quadratic equations. For example, they write and use algorithms and logarithmic functions to sort data or solve a problem, such as programming audio controls.
		Statistics and Probability
>>>	Summary Calculations	Calculate averages. For example, Use tables, schedules or other table-like text. Use graphical presentations. For example, they use arithmetic mean to conduct high-level analysis of test data. They use random-number utilities to ensure randomness.

		They measure the average CPU usage using parameters such as processor speed and available memory. They may calculate the number of computer instructions that can be executed in a given time.
>>>	Statistics and Probabilities	Use descriptive statistics (e.g. collecting, classifying, analyzing and interpreting data). Use inferential statistics (e.g. using mathematical theories of probability, making conclusions about a population or about how likely it is that some event will happen). For example, they use statistical analysis software to analyze and mine data, and to develop forecasts. They create and use logic and truth tables to analyze statements to verify whether or not they are logical or true.

b. How Calculations are Performed

- In the worker's head.
- Using a calculator.
- Using a computer.

c. Measurement Instruments Used

- Time using a watch or clock.

E. Oral Communication

Oral Communication

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 3	• attend meetings with developers to report project updates, discuss issues and how they can be resolved, ask questions, and report project status. (2)
Most Complex	3	• attend meetings to discuss assignment of tasks and ideas for completing tasks. (2) • adjust language when speaking to different people. For example, they speak in less technical language when speaking to managers. They ask questions to get a sense of project requirements or obtain information such as login passwords. They also obtain requirements requested by the product manager. They use technical language when speaking with engineers, and must be tactful when answering questions, providing or asking for help. (2)

		• attend meetings with co-workers, colleagues, and supervisors to solve software related problems, to discuss new developments, and to gather and receive information about projects. For example, they brainstorm for ideas to solve problems with bugs or to discuss opinions and share information. At the end of projects, they meet to analyze how the project went and lessons learned. (3) • may host meetings to facilitate the exchange of ideas between subject matter experts. (3) • may train, instruct and advise co-op students or junior developers. For example, they make themselves available to answer questions and demonstrate how to do something, to show them how to plan and document tasks, and to discuss ways of improving their skills. (3) • meet clients face to face or by phone to discuss timelines, or to clarify details of project requirements. (3) • may make presentations to co-workers and managers to share knowledge. For example, they may present current projects or new technology. (3)

Modes of Communication Used

- In person. For example, speaking to colleagues about issues with the project.
- Using a telephone. For example, speaking with clients about project updates.
- Others. For example, video conferencing with clients.

Environmental Factors Affecting Communication

None reported.

Oral Communication Summary

The symbols >, >> and >>> are explained in the Use of Symbols section.

Purpose for Oral Communication (Part I)						
Type	**To greet**	**To take messages**	**To provide/receive information, explanation, direction**	**To seek, obtain information**	**To co-ordinate work with that of others**	**To reassure, comfort**
Listening (little or no interaction)			>>			
Speaking (little or no interaction)			>>			
Interact with co-workers			>>>	>>>	>>>	
Interact with those you supervise or direct			>>	>>	>>	
Interact with supervisor/manager			>>>	>>>	>>>	
Interact with peers and colleagues from other organization			>>	>>	>>	
Interact with customers/clients/ public			>>	>>		
Interact with suppliers, servicers						
Participate in group discussion			>>>	>>>	>>>	
Present information to a small group			>>	>>		
Present information to a large group						

The symbols >, >> and >>> are explained in the Use of Symbols section.

Purpose for Oral Communication (Part II)						
Type	**To discuss (exchange information, opinions)**	**To persuade**	**To facilitate, animate**	**To instruct, instill understanding, knowledge**	**To negotiate, resolve conflict**	**To entertain**
Listening (little or no interaction)			>>			
Speaking (little or no interaction)			>>			
Interact with co-workers	>>>	>>>	>>>	>>>	>>>	
Interact with those you supervise or direct	>>			>>		
Interact with supervisor/manager	>>>					
Interact with peers and colleagues from other organization	>>					
Interact with customers/clients/ public	>>	>>		>>	>>	
Interact with suppliers, servicers						
Participate in group discussion	>>>	>>>		>>>		
Present information to a small group	>>		>>	>>		
Present information to a large group						

F. Thinking Skills

1. Problem Solving

Problem Solving

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 4	• deal with communication and teamwork situations. With large organizations, several people may be working on the same project but the teams are not communicating effectively with each other. (2)
Most Complex	4	• anticipate questions from clients, for example, asking how to perform a specific task in the software. They may need to make changes in features that already exist or create new features. (3) • figure out an appropriate design based on customer requirements and existing software infrastructure. If current software infrastructure is inadequate, they have to figure out workarounds or solutions that might mean rebuilding the infrastructure. (4) • figure out how to clean up or rewrite code that someone else wrote. If code has accumulated over a long period of time, this could take several months. (4) • troubleshoot and fix problems such as bugs or defects. They consult with co-workers, run tests to determine where problems are occurring, use debugging tools and then retest. For example, they may use run time data. They break down the problem into steps to give the machine instructions and make sure that it runs. They may build a prototype to troubleshoot problems. (4)

2. Decision Making

Decision Making

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 4	• may decide which tasks to assign to team members. They consider individual skills and experience. (2) • anticipate what needs to be done in a project during the design stage. For example, they are given a spec and must design something that fits that spec. They decide on tools
Most Complex	4	

		to use. (2) • decide where to make budget adjustments when there is not enough to complete the project. They may reduce the project scope, include overtime or hire more contractors. (3) • look at project plans, research the specs and technology, as well as algorithms to determine if more resources are needed. For example, if they have enough contractors or require more to complete the project. (3) • decide which software design to implement based on the pros and cons of each. They may decide from several design options that meet the customer's requirements. For example, the best technical solution might take more time and require bringing in more consultants meaning delayed delivery and higher costs. A quicker solution might answer current needs but be harder to maintain in the long run. (4)

3. Critical Thinking

Critical Thinking

Tasks	**Complexity Level**	**Examples**
		Software Engineers and Designers:
Typical	2 to 4	• may evaluate quality of work done by co-workers and junior team members. They may evaluate documentation to ensure its adequacy, accuracy and clarity. (2)
Most Complex	4	• test, problem solve, and analyze software that may span multiple software subsystems. For example, software that has safety implications requires multiple checks. (4) • evaluate design options. The goal is to address the needs of the current customer with the best technical solution. For example, they may have a solution for a customer based on previous work. Or a customer may be asking for particular features that, based on analyzing trends, will go out of style in a short period of time. This may mean developing a "quick and dirty" solution that will work for now without investing a lot of resources and allow technical staff to transition to the next generation of products. (4)

4. Job Task Planning and Organizing

Job Task Planning and Organizing

Complexity Level	Description
3	Own job planning and organizing: • Software engineers and designers plan job tasks independently. They review priorities and revise the order of job tasks in response to requests from managers and other members of the team, or modifications to the project. Organizational planning: • Software engineers and designers assign tasks to junior team members. They may coordinate the activities of the development team.

5. Significant Use of Memory

Examples

- remember where to find files on their computers.
- remember passwords and identification numbers.
- remember what issues were dealt with before and where the problem was fixed so there is a reference for how to deal with similar situations.
- remember previous problems and programming bugs and use this information for problem solving.
- remember programming information and sequences.

6. Finding Information

Finding Information

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	3	• ask co-workers and members of other teams for information on how to deal with issues. For example, they inquire about bugging issues during team meetings, exchange email to ask for advice, and ask questions about programming code during professional development presentations. (3)
Most Complex	3	• find information online. They ask for information from Java user groups or system administrator user groups

		about, for example, how to avoid system crashes. They search for information online by using Google and visiting forums to read about computer language specific problems. (3) • look up information on programming code while doing regular programming. For example, they search technical manuals, help desks, and other resources to troubleshoot bugs and other defects. They search gdb files to find solutions to problems with code. (3)

G. Working with Others

Working with Others

	Description
Complexity Level 3	Software engineers and designers work as members of teams. They coordinate and integrate their work with other software engineers and designers, computer programmers, project managers, product managers, sales engineers and others.

Participation in Supervisory or Leadership Activities

- participate in formal discussions about work processes or product improvement.
- have opportunities to make suggestions on improving work processes.
- monitor the work performance of others.
- inform other workers or demonstrate to them how tasks are to be performed.
- orient new employees.
- make hiring recommendations.
- make hiring decisions.
- select contractors and suppliers.
- assign routine tasks to other workers.
- assign new or unusual tasks to other workers.
- identify training that is required by, or would be useful for, other workers.
- deal with other workers' grievances or complaints.

H. Digital Technology

Digital Technology

Tasks	Complexity Level	Examples
		Software Engineers and Designers:
Typical	2 to 5	• use communications software. For example, they exchange email and attach documents with colleagues and clients to discuss details on projects, set up meetings, and send customer reports. (2)
Most Complex	5	• use applications that integrate video, voice and the sharing of data in conferences and meetings. (2) • use visual editors such as VI and Emacs that display the text being edited on the screen. Editors have compilers that run the program and transform the source file into a form that can be run (a binary file). They may also use text editors such as Notepad, which is a text-only editor. (2) • visualize the design of a system using Unified Modeling Language (UML) that gives a graphical representation of the final version of the software. (2) • use the Internet to search for information. For example, they use Internet browsers to access vendor websites, and online trade publications and forums for information on programming code and guidelines on troubleshooting. They use platforms such as SharePoint to integrate intranet content management and document management. (3) • use project management software such as JIRA and Microsoft Projects to assign work and follow team activity, track bugs and issues, and share files at their desktop or mobile. (3) • use presentation software such as PowerPoint to create slide shows for co-workers and customers. For example, they create presentations for professional development with colleagues. (3) • use word processing software such as Word to create, edit and format documents such as project reports, and test reports, memos, reference notes, procedures and schedules. (3) • use statistical analysis software to analyze, interpret and mine data, and to build analytical graphs, maps and charts. (4) • develop and debug computer programs. For example, they use integrated development environments (IDE) such as Eclipse and QT Creator to develop code and debug errors, and IDE such as Visual Studio to develop computer programs as well as web applications. Software engineers and designers use source code control systems to control changes and track the development of source code. They use programming languages, such as JavaScript, C++, and Ruby to develop computer programs such as

		applications, utilities, servers and systems programs. (5)

Computer Use Summary

- use word processing
- use graphics software
- use databases
- use spreadsheets
- use communications software

I. Continuous Learning

Continuous Learning

Complexity Level	Description
4	Software engineers and designers are constantly learning new skills because technology changes so quickly. They need to be able to quickly learn the knowledge required as they move from project to project. They learn on the job, from other team members, and from colleagues with more experience. Senior software engineers and designers mentor junior software engineers and designers. They may receive training through work, online training or university courses. They research and find information on the web.

How Learning Occurs

Learning may be acquired:

- As part of regular work activity.
- From co-workers.
- Through training offered in the workplace.
- Through reading or other forms of self-study:
    - at work.
    - on worker's own time.
    - using materials available through work.
    - Using materials obtained through a professional association or union.
    - using materials obtained on worker's own initiative.
- Through off-site training:
    - during working hours at no cost to the worker.
    - partially subsidized.

J. Other Information

In addition to collecting information for this Essential Skills Profile, our interviews with job incumbents also asked about the following topics.

Physical Aspects

Mostly sitting.

Attitudes

Software engineers and designers need to be fast learners, have good research skills, and a creative approach to problem solving. They should have a positive attitude toward working and communicating as a team and have the ability to work with people at all levels. They need to be able to work under pressure and meet deadlines.

Impact of Digital Technology

All essential skills are affected by the introduction of technology in the workplace. Software engineers and designers' ability to adapt to new technologies is strongly related to their skill levels across the essential skills, including reading, writing, thinking and communication. Technologies are transforming the ways in which workers obtain, process and communicate information, and the types of skills needed to perform in their jobs. In particular, software engineers and designers need enhanced digital skills to work with rapidly changing computer technologies. For instance, they need designing and troubleshooting skills to write and edit code, and use software and platforms to manage and develop files and programs. They also require the necessary skills to solve potential problems to client satisfaction and manage project tasks and deadlines. New developments in network technology require these workers to continually enhance their skills in order to keep current.

Technology in the workplace further affects the complexity of tasks related to the essential skills required for this occupation. Software engineers and designers need the skills to use increasingly complex and specialized software and platforms. For example, they use TFS and IDE to write source code. On the other hand, the use of technology to research information and communicate with others in the field makes it easier for workers to find solutions to issues.

K. Notes

This profile is based on interviews with job incumbents across Canada and validated through consultation with industry experts across the country.

For information on research, definitions, and scaling processes of Essential Skills Profiles, please consult the Readers' Guide to Essential Skills Profiles.