

MINNESOTA PIPELINE PROJECT

PRIVATE INVESTMENT, PUBLIC EDUCATION LABOR AND INDUSTRY EXPERIENCE

Software Developer - Software developers design, build and test computer systems that help organizations and equipment to work more effectively. Examples of work include information databases, programs that control robotic systems, and cloud and mobile applications.

Industry-Sector Technical Competencies

- Bash Shell Scripting – Knowledge of scripting a UNIX shell or command language.
- Software Testing – Knowledge of how to evaluate software to make sure it meets specified requirements. Also to identify any gaps, errors or missing requirements.
- Software Analysis and Design – Understanding of modeling and its central role in eliciting, understanding, analyzing and communicating software requirements, architecture and design.
- Programming – Training to create programs by writing "code" in a programming language.
- Service Oriented Architectures – Understand the architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network.
- Logic – Training in the part of the program that encodes the real-world business rules that determine how data can be created, displayed, stored, and changed.
- Object Orientated Programming – Understanding this type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure.
- Databases – Knowledge of implementing data models and database designs to ensure security and data integrity in database software.
- Version Control – Understanding of the system that records changes to a file or set of files over time so that you can recall specific versions later.
- Data Structures & Algorithms – Knowledge of the use of data structures and algorithms in software programming.
- Operating Systems – Understand the function of operating systems and how to properly create software to interact with them.
- Unified Modeling Language – Understanding of the general-purpose modeling language for software engineering, designed to provide a standard way to visualize the design of a system.

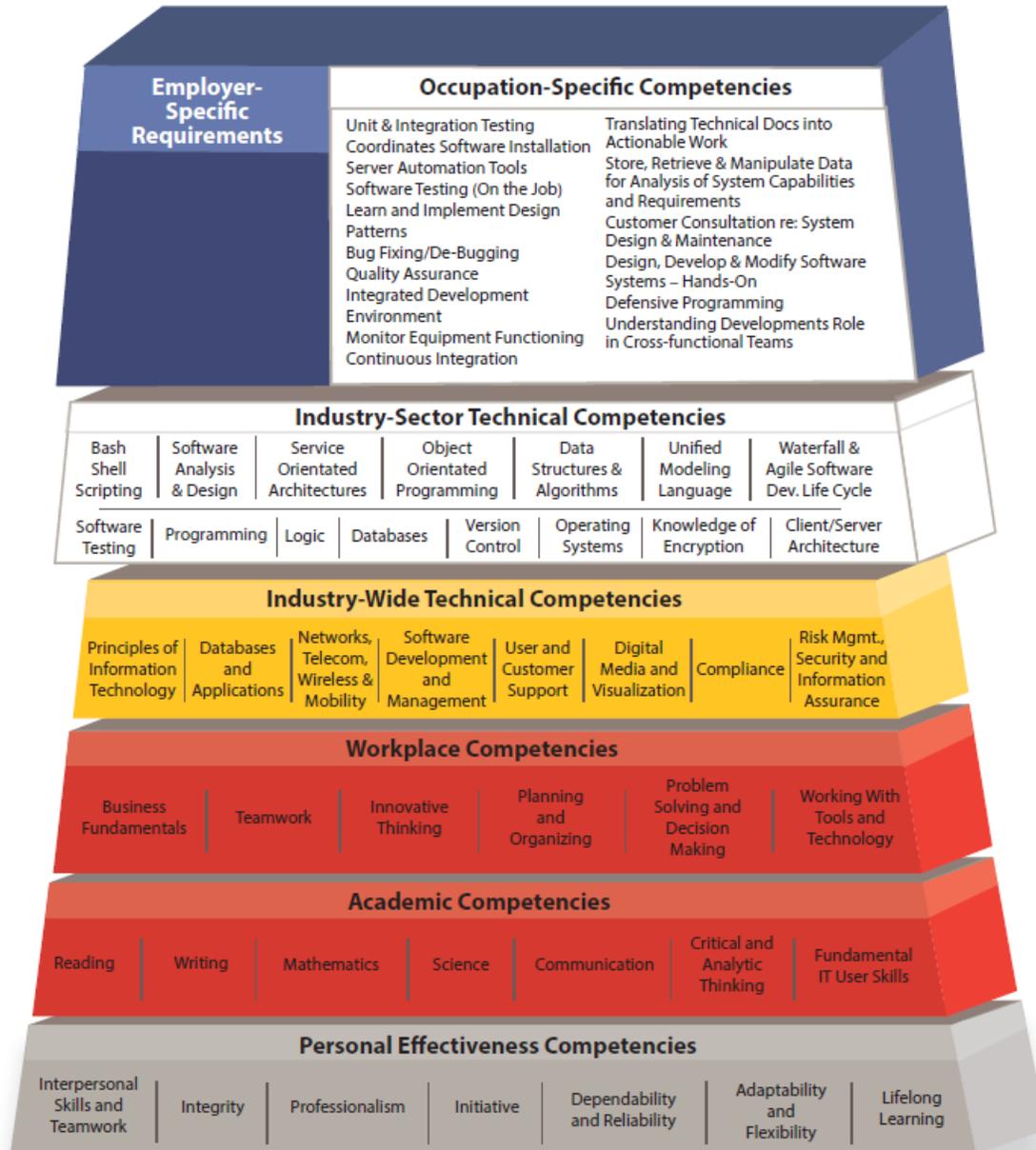
- Knowledge of Encryption – Understanding of how encryption functions and how to work with it within the software development environment.
- Software Development Life Cycle – Knowledge of Waterfall and Agile approaches to software development and when to use the appropriate model.
- Client/Server Architecture – Knowledge of the Client/Server Architecture model and how to develop software for such a system.

Occupation-Specific Competencies

- Unit & Integration Testing – Able to test various computing scenarios for units and integration.
- Coordinate Software Installation – Assist with software installation for the organization and individual user.
- Server Automation Tools – Know how to use applications which automate computing functions.
- Software Testing (On The Job) – Ability to run tests on software and test for compatibility and functionality issues.
- Learn and Implement Design Patterns – Use design patterns for problem solving in programming.
- Bug Fixing/De-Bugging – Ability to locate, fix or bypass errors (bugs) in code or devices.
- Quality Assurance – Use appropriate methods to verify overall quality of software design and system work.
- Integrated Development Environment – Use the IDE application for software development.
- Monitor Equipment Functioning – Monitor system in order to review information to detect or assess problems.
- Continuous Integration – Merge developer working copies with shared mainline several times a day.
- Translating Technical Docs Into Actionable Work – Understand how to create working process documents from very technical IT documents.
- Data Analysis – Store, retrieve and manipulate data for analysis of system capabilities and requirements.
- Customer Consultation - Work with internal and external customers to gather information regarding software requirements and customization.
- Software Systems – Demonstrate ability to design, develop and modify software systems.
- Defensive Programming – Ability to design model intended to ensure the continuing function of a piece of software under unforeseen circumstances.
- Cross-Functional Teams – Understand the software development role while working with cross-functional teams.

PIPELINE Project

Competency Model for Information Technology Occupation: Software Developer



Based on: Information Technology Competency Model Employment and Training Administration, United State Department of Labor, September 2012 and Digital Industries Trailblazer Apprenticeship – Software Developer Occupational Brief. UK <http://www.e-skills.com/apprenticeships/trailblazer-consultation/>

Software Developer Occupational Competency Training Plan

***Related Instruction** means an organized and systematic form of instruction designed to provide the apprentice with the knowledge of the theoretical and technical subjects related to the apprentice's trade of occupation, or industrial courses or, when of equivalent value, by correspondence, electronic media, or other forms or self-study approved by the commissioner.*

	Course	Course Description	Credit/Non-Credit	Hours Spent on Competency
Bash Shell Scripting				
Software Testing				
Software Analysis & Design				
Programming				
Service Orientated Architectures				
Logic				
Object Orientated Programming				
Databases				

Version control				
Data Structures & Algorithms				
Operating Systems				
Unified Modeling Language				
Knowledge of Encryption				
Waterfall & Agile Software Dev. Life Cycle				
Client/Server Architecture				

***On-The-Job Training** is the work experience and instruction. Training experience need not be in the exact order as listed below.*

	Trainer/Instructor	Name of person responsible for verifying competency mastery	Hours Spent on Competency
Unit & Integration Testing			
Coordinates Software Installation			

Server Automation Tools			
Software Testing (On the Job)			
Learn and Implement Design Patterns			
Bug Fixing/De-Bugging			
Quality Assurance			
Integrated Development Environment			
Monitor Equipment Functioning			
Continuous Integration			
Translating Technical Docs into Actionable Work			
Store, Retrieve & Manipulate Data for Analysis of System Capabilities and Requirements			

Customer Consultation re: System Design & Maintenance			
Design, Develop & Modify Software Systems - Hands-On			
Defensive Programming			
Understanding Developments Role in Cross-functional Teams			