Minnesota Dual-Training Pipeline
Competency Model for Information Technology
Occupation: Software Engineer/Developer

Employer-Specific Requirements

Occupation-Specific Competencies

- Unit and integration testing
- Server automation tools
- Software testing
- Analysis and design
- Bug fixing/de-bugging
- Quality assurance
- Integrated development environment
- Customer consultation

Monitor equipment functioning
Continuous integration
Software systems
Defensive programming
Cross-functional teams
Websites and applications
Data analysis
Translating technical docs into actionable work

Industry-Sector Technical Competencies

<table>
<thead>
<tr>
<th>Bash shell scripting</th>
<th>Software analysis, design and testing</th>
<th>Service orientated architectures</th>
<th>Object orientated programming</th>
<th>Data structures &amp; algorithms</th>
<th>Unified modeling language</th>
<th>Waterfall and Agile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance engineering</td>
<td>Programming</td>
<td>Logic</td>
<td>Databases</td>
<td>Version control</td>
<td>Cloud-based development</td>
<td>Knowledge of encryption</td>
</tr>
</tbody>
</table>

Industry-Wide Technical Competencies

- Principles of information technology
- Databases and applications
- Networks, telecom, wireless & mobility
- Software development and management
- User and customer support
- Digital media and visualization
- Compliance
- Risk mgmt., security and information assurance

Workplace Competencies

- Business fundamentals
- Teamwork
- Innovative thinking
- Planning and organizing
- Problem solving and decision making
- Working with tools and technology

Academic Competencies

- Reading
- Writing
- Mathematics
- Science
- Communication
- Critical and analytic thinking
- Fundamental IT user skills

Personal Effectiveness Competencies

- Interpersonal skills and teamwork
- Integrity
- Professionalism
- Initiative
- Dependability and reliability
- Adaptability and flexibility
- Lifelong learning

Based on: Information Technology Competency Model Employment and Training Administration, United States Department of Labor, September 2012.

*Pipeline recommends the Industry-Sector Technical Competencies as formal training opportunities (provided through related instruction) and the Occupation-Specific Competencies as on-the-job training opportunities.
Competency Model for Software Engineer/Developer

Software Engineer/Developer – Responsible for designing, building, and testing computer systems that help organizations and equipment 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 analysis, design, and testing** – Understanding of modeling and its central role in eliciting, understanding, analyzing, testing, 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.
- **Object oriented programming** – Understand the 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.
- **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.
- **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.
- **Cloud-based development** – Understand the creation and deployment of cloud apps.
- **Performance engineering** – Understanding of the techniques applied during a systems development life cycle to ensure the non-functional requirements for performance will be met.
- **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.

• **Client/server architecture** – Knowledge of the Client/Server Architecture model and how to develop software for such a system.

• **Waterfall and agile** – Understand sequential methodologies of waterfall and agile as processes to complete software development and testing.

**Occupation-Specific Competencies**

• **Unit and integration testing** – Be able to test various computing scenarios for units and integration.

• **Websites and applications** – Understand the use of globally accessible web pages and software applications.

• **Server automation tools** – Know how to use applications which automate computing functions.

• **Software testing** – Understand how to test software’s ability to perform and function before installation.

• **Analysis and design** – Understand activities which help the transformation of requirement specification into implementation.

• **Bug fixing/de-bugging** – Ability to locate, fix or bypass errors (bugs) in code or devices.

• **Quality assurance** – Know how to use appropriate methods to verify overall quality of software design and systemwork.

• **Integrated development environment** – Know how to use the IDE application for software development.

• **Monitor equipment functioning** – Know how to monitor system in order to review information to detect or assess problems.

• **Translating technical docs into actionable work** – Understand how to create working process documents from very technical IT documents.

• **Continuous integration** – Know how to merge developer working copies with shared mainline several times a day.

• **Data analysis** – Understand how to store, retrieve and manipulate data for analysis of system capabilities and requirements. Also know how to understand the process of cleaning, transforming, and modeling data to discover useful information for software development decision-making.

• **Customer consultation** – Know how to work with internal and external customers to gather information regarding software requirements and customization.

• **Software systems** – Be able to design, develop and modify software systems.

• **Defensive programming** – Know how 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 team.
**Software Engineer/Developer Occupational Competency Training Plan**

*Related Instruction* means an organized and systematic form of instruction designed to provide the dual trainee with the knowledge of the theoretical and technical subjects related to the dual trainee’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.

<table>
<thead>
<tr>
<th>Course</th>
<th>Course Description</th>
<th>Credit/Non-Credit</th>
<th>Hours Spent on Competency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bash shell scripting</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance engineering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software analysis, design &amp; testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service orientated architectures</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object oriented programming</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waterfall and Agile</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Databases</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data structures &amp; algorithms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cloud-based development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unified modeling language</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Knowledge of encryption</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client/server Architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**On-The-Job Training** is the work experience and instruction. Training experience need not be in the exact order as listed below.

<table>
<thead>
<tr>
<th>Trainer/Instructor</th>
<th>Name of person responsible for verifying competency mastery</th>
<th>Hours spent on competency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit &amp; integration testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Websites and applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Server automation tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis and design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bug fixing/de-bugging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality assurance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrated development environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitor equipment functioning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Translating technical docs into actionable work</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer consultation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software systems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defensive programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cross-functional teams</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Updated December 2021