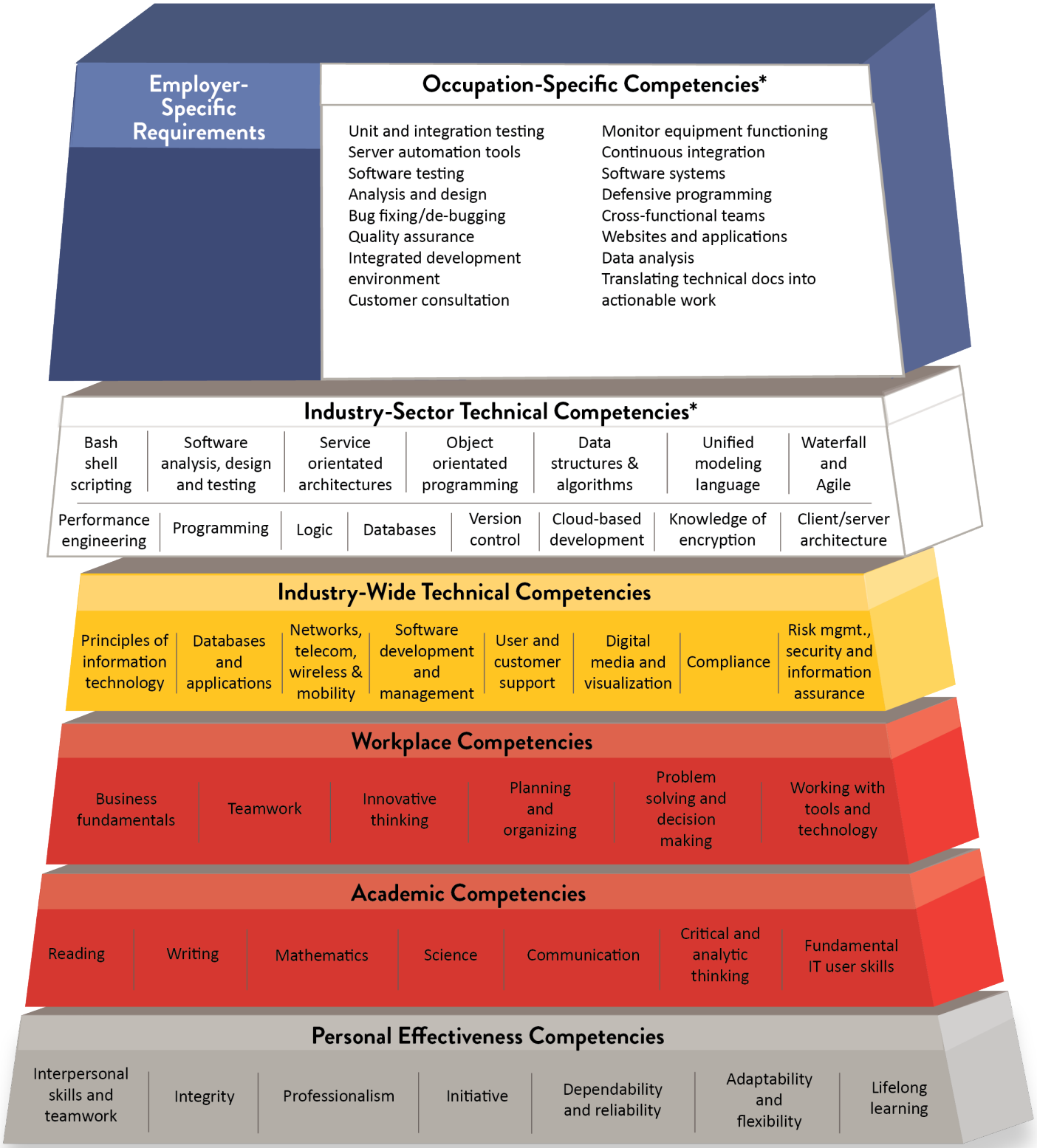
Minnesota Dual-Training Pipeline

Competency Model for Information Technology
Occupation: Software Engineer/Developer



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 – An individual who is 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, cloud and mobile applications.

Industry-Sector Technical Competencies

Related Instruction for dual training means the organized and systematic form of education resulting in the enhancement of skills and competencies related to the dual trainee's current or intended occupation.

- **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 and 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

On-the-Job Training (OJT) is hands-on instruction completed at work to learn the core competencies necessary to succeed in an occupation. Common types of OJT include job shadowing, mentorship, cohort-based training, assignment-based project evaluation and discussion-based training.

- **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 system work.

- **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 documents 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 regardingsoftware 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 a cross-functional team

Updated July 2022