GUIDANCE NOTE: FOR END-POINT ASSESSMENT ORGANISATION USE ONLY

AMENDED: 1st November 2021

Software Developer (ST0116)

END-POINT ASSESSMENT ADDITIONAL GUIDANCE ON SOME OF THE OCCUPATIONAL BRIEF COMPETENCY STANDARDS

The following guidance is not intended for training provider use.

The guidance is designed to support End-Point Assessment Organisations (EPAOs) by providing some clarity to those parts of the occupational brief that have caused uncertainty when assessing and moderating apprenticeship work.

The table shows individual competency standards and the minimum expected requirements to pass some of the criteria listed in the occupational brief. It then offers guidance on how this could be interpreted.

Note that there are other criteria (competency standards) in the occupational brief. This table focuses on just those competency standards EPAOs felt needed further guidance.

This is indicative guidance only and represents an attempt to develop a shared understanding of how the competency standards should be interpreted.

The What – what the apprentice has shown they can do			
The Competency Standard	**Definition of the Minimum Requirement**	**Examples and illustrations – NOT the curriculum**	**Comments on the criteria**
Logic: writes good quality code (logic) with sound syntax in at least one language.	Apprentices can write code to achieve the desired functionality and which is easy to read and understand, with good naming, indentation and commenting, and applying the fundamentals of good coding: • development paradigms (where this is object-oriented programming this must include inheritance, abstractions, encapsulation, polymorphism) • software programming languages • software development tools (IDEs) • writing programs and methods • language-specific idioms • logic and flow-of-control	A wide range of software development tools including: • Integrated Development Environments (IDE's) • Version control systems • Configuration management tools For example, in Java the ability to code variables assignment statements data types conditionals statements loops arrays	The apprentice is applying and demonstrating what they know and can do here. This will be apparent when assessing the source code and/or examples provided in their portfolio. Apprentices should be able to describe this competence if asked for further detail in the interview.

	Apprentices can apply: • Elements of programming – variables, assignment statements, data types, conditionals, loops, arrays, and input/output. • Functions - modular programming dividing a program into components that can be independently debugged, maintained, and reused writing at least two reusable functions • Algorithms and data structures - classical algorithms for sorting and searching, and fundamental data structures.	For example, in Java to be able to declare and invoke methods correctly demonstrate parameter passing and returning values overloading and overriding. For example, in Java demonstrate the use of collection classes the difference between standard arrays and collection classes.	For example, they could be asked to describe the principles applied to the language(s) they use to meet the logic criterion in the occupational brief.
Design: can create simple data models and software designs to effectively communicate understanding of the program, following best practices and standards	Can take a high-level design and can interpret and convert the design into simple data models and/or programme modules to communicate it to others. Can apply software design methodologies (e.g., structured or object-oriented). Can use standard design notation such as UML. Can apply data modelling. **Can apply reconcile design against analysis models.** Can design software solutions to meet requirements.	• Software design tools • Data modelling tools	This can be read as '**Can reconcile design against analysis models'**.
Analysis: can understand and create basic analysis artefacts, such as user cases and/or user stories.	Can take a variety of data and business requirements and convert them into basic analysis artefacts to understand and can clarify the intended use of the proposed software Can identify and represent required functionality (e.g. **use** cases) Can identify and represent activity workflow (e.g. activity diagrams).	• Software analysis tools • Case development tools • Activity diagram tools	For 'Can identify and represent required functionality (e.g. use cases), this should read: Can identify and represent required functionality (e.g. ***user*** cases).