VCE Software Development 2024
Unit 3 School-based Assessment

Video 2
Background to the
Unit 3 Outcome 1 SAC

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY
VICTORIA State Government

Acknowledgement of Country

The VCAA respectfully acknowledges the Traditional Owners of Country throughout Victoria and pays respect to the ongoing living cultures of First Peoples.

VCE Software Development 2024
Unit 3 School-based Assessment

Video 2
Background to the
Unit 3 Outcome 1 SAC

Phil Feain
Digital Technologies Curriculum Manager
VCAA

VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VICTORIA State Government

Purpose of this presentation

- to build the capacity of teachers to develop compliant, rigorous and engaging VCE assessment tasks in line with the VCE assessment principles
- provide an overview of the Unit 3 Outcome 1 School-assessed Coursework (SAC) task.

Unit 3 Outcome 1
School-assessed Coursework (SAC)

Developing a compliant, engaging and rigorous assessment task

Planning

When you are ready to write the assessment task have the following documents in front of you (These are all on the Software Development study page):

- Applied Computing Study Design – U3 O1 Software Development – pages 36–40
  - Area of Study statement, Outcome statement, Key knowledge and Key skills
- Programming requirements document
- Advice for teachers
  - Software Development: Unit 3 Area of Study 1 – Sample approaches to developing an assessment task
  - Software Development: Unit 3 Outcome 1 – Performance descriptors
- School-based Assessment Audit report (2020)
- Resources:
  - 2024 Software Development U3O1 Assessment task development template – Blank
  - 2024 Software Development U3O1 Assessment task development template – Plan
  - 2024 Software Development U3O1 Developing a marking scheme – Sample
  - 2024 Software Development U3O1 SAC Task template

Unit 3 Outcome 1 – The outcome

On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.

Unit 3 Outcome 1 – The assessment task

Contribution to final assessment

School-assessed Coursework for Unit 3 will contribute 10 per cent to the study score.

Outcomes	Marks allocated	Assessment tasks
Unit 3 **Outcome 1** Interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.	100	In response to teacher-provided solution requirements and designs, create working modules.
Total marks	100	

Key knowledge

Data and information

- characteristics of data types
- types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index)

Approaches to problem-solving

- methods for documenting a problem, need or opportunity
- methods for determining solution requirements, constraints and scope
- methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode
- formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats
- a programming language as a method for developing working modules that meet specified needs
- naming conventions for solution elements
- processing features of a programming language, including classes, control structures, functions, instructions and methods
- algorithms for sorting, including selection sort and quick sort
- algorithms for binary and linear searching
- validation techniques, including existence checking, range checking and type checking
- techniques for checking that modules meet design specifications, including trace tables and construction of test data
- purposes and characteristics of internal documentation, including meaningful comments and syntax.

Teaching towards the assessment task

Learning activities have been developed to meet the Unit 3 Outcome 1 Key knowledge dot points.

These learning activities can be found in the Advice for teachers.

Examples of learning activities

- Identify and describe specific uses for a range of data types. For example:
  - Telephone numbers are best stored as strings because often characters such as parentheses or spaces are usually required to be stored and no numerical processing is required.
  - Sorting numeric values or numbers as text achieves a different outcome – 1, 3, 5, 15, 101 (numeric values) vs. 1, 101, 15, 3, 5 (numbers as text) – due to text sorting not taking into account place value.
  - Australian currency should be stored as floating point values to two decimal places, rather than integers, because currency uses dollars and cents.

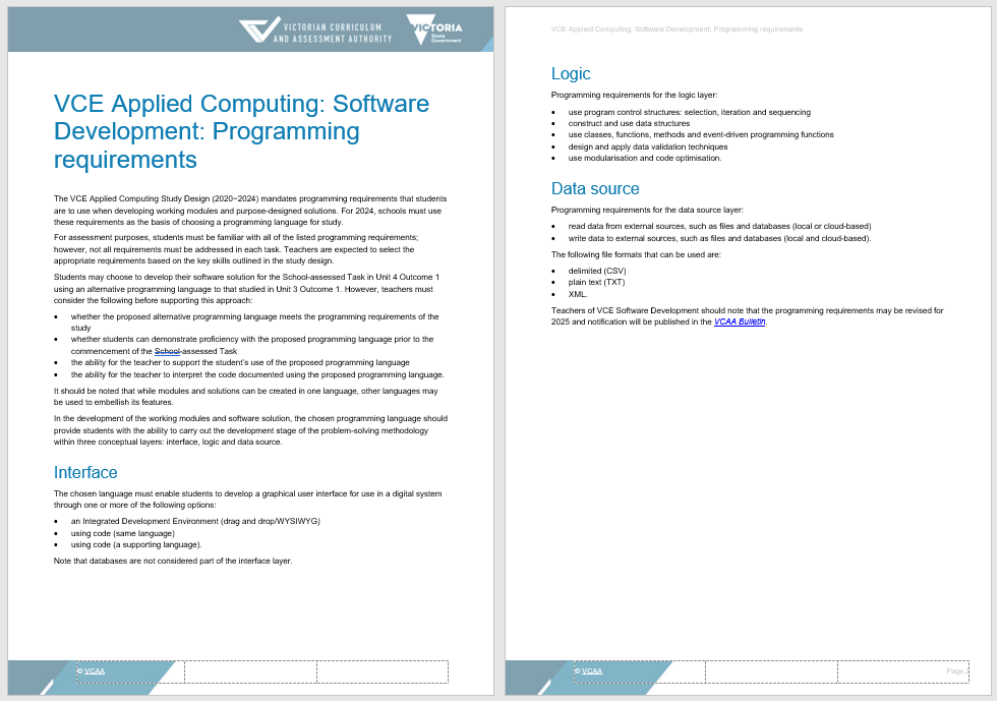
Key skills

- interpret solution requirements and designs to develop working modules
- use a range of data types and data structures
- use and justify appropriate processing features of a programming language to develop working modules
- develop and apply suitable validation, testing and debugging techniques using appropriate test data
- document the functioning of modules and the use of processing features through internal documentation.

VCAA Performance descriptors

SOFTWARE DEVELOPMENT UNIT 3 OUTCOME 1 SCHOOL-ASSESSED COURSEWORK					
Performance Descriptors					
	DESCRIPTOR: typical performance in each range				
	Very low	**Low**	**Medium**	**High**	**Very high**
Unit 3 ***Outcome 1*** ***Interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.***	Limited interpretation of solution requirements and designs to develop working modules.	Some interpretation of solution requirements and designs to develop working modules.	Sound interpretation of solution requirements and designs to develop working modules.	Most solution requirements and designs are interpreted accurately to developing working modules.	All solution requirements and designs are interpreted accurately to developing working modules.
	Limited selection and use of data types and data structures.	Some selection and use of appropriate data types and data structures.	Sound selection and use of data types and data structures to develop working modules.	Detailed selection of relevant data types and data structures to develop working modules.	Comprehensive selection of relevant data types and data structures to develop working modules.
	Limited selection and use of processing features of the programming language to develop some working modules.	Some selection and use of appropriate processing features of the programming language to develop some working modules.	Sound selection and use of appropriate processing features of the programming language to develop some working modules.	Most processing features of the programming language have been selected and used to develop all working modules.	Comprehensive selection and use of relevant processing features of the programming language to develop all working modules.
	Limited explanation of how the selected processing features are used to develop working modules.	Some justification and explanation of how the selected processing features are used to develop working modules.	Sound justification and explanation of how the selection of appropriate processing features are used to develop working modules.	Detailed justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.
	Limited data validation techniques are applied to check the reasonableness of some input data.	Some data validation techniques are effectively applied to check the reasonableness of some input data.	Sound use of data validation techniques are effectively applied to check the reasonableness of input data.	Detailed use of relevant data validation techniques are applied to efficiently and effectively check the reasonableness of all input data.	Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data.
	Limited range of test data is expressed in a testing table, with incomplete or missing results.	Some testing of test data is expressed in a testing table with actual output stated.	Sound range of testing of test data is expressed in a testing table, with both expected and actual output stated and some evidence of debugging.	Detailed use of test data is expressed in a testing table, with both expected and actual output stated with evidence of debugging.	Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.
	Limited internal documentation with few comments regarding the use of the selected processing features.	Some internal documentation with comments regarding the functioning of modules and the use selected processing features.	Sound use of internal documentation with comments regarding the functioning of modules and the use of selected processing features.	Most software modules include detailed internal documentation regarding the functioning of modules and use of selected processing features.	All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.

Programming requirements document



VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VCE Applied Computing: Software Development: Programming requirements

The VCE Applied Computing Study Design (2020–2024) mandates programming requirements that students are to use when developing working modules and purpose-designed solutions. For 2024, schools must use these requirements as the basis of choosing a programming language for study.

For assessment purposes, students must be familiar with all of the listed programming requirements; however, not all requirements must be addressed in each task. Teachers are expected to select the appropriate requirements based on the key skills outlined in the study design.

Students may choose to develop their software solution for the School-assessed Task in Unit 4 Outcome 1 using an alternative programming language to that studied in Unit 3 Outcome 1. However, teachers must consider the following before supporting this approach:

- whether the proposed alternative programming language meets the programming requirements of the study
- whether students can demonstrate proficiency with the proposed programming language prior to the commencement of the School-assessed Task
- the ability for the teacher to support the student's use of the proposed programming language
- the ability for the teacher to interpret the code documented using the proposed programming language.

It should be noted that while modules and solutions can be created in one language, other languages may be used to embellish its features.

In the development of the working modules and software solution, the chosen programming language should provide students with the ability to carry out the development stage of the problem-solving methodology within three conceptual layers: interface, logic and data source.

Interface

The chosen language must enable students to develop a graphical user interface for use in a digital system through one or more of the following options:

- an Integrated Development Environment (drag and drop/WYSIWYG)
- using code (same language)
- using code (a supporting language).

Note that databases are not considered part of the interface layer.

© VCAA

Logic

Programming requirements for the logic layer:

- use program control structures: selection, iteration and sequencing
- construct and use data structures
- use classes, functions, methods and event-driven programming functions
- design and apply data validation techniques
- use modularisation and code optimisation.

Data source

Programming requirements for the data source layer:

- read data from external sources, such as files and databases (local or cloud-based)
- write data to external sources, such as files and databases (local and cloud-based).

The following file formats that can be used are:

- delimited (CSV)
- plain text (TXT)
- XML.

Teachers of VCE Software Development should note that the programming requirements may be revised for 2025 and notification will be published in the VCAA Bulletin.

© VCAA

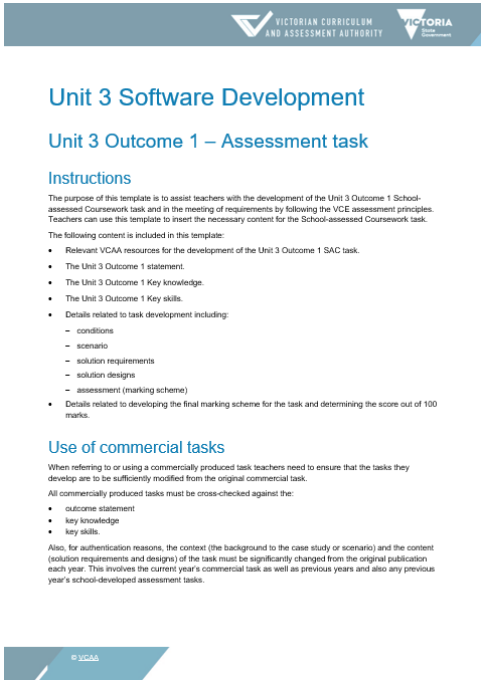
This document is available on the Software Development study page.

It gives guidance for the programming requirements to be followed by students for the Unit 3 Outcome 1 SAC task and the Unit 3 Outcome 2 and Unit 4 Outcome 1 SAT.

Designing the assessment task

To assist with the development of the Unit 3 Outcome 1 assessment tasks we have developed a SAC Task template for teachers to follow and use.

The purpose of the template is to assist teachers in developing an assessment task that meets requirements.



VICTORIAN CURRICULUM AND ASSESSMENT AUTHORITY

VICTORIA State Government

Unit 3 Software Development

Unit 3 Outcome 1 – Assessment task

Instructions

The purpose of this template is to assist teachers with the development of the Unit 3 Outcome 1 School-assessed Coursework task and in the meeting of requirements by following the VCE assessment principles. Teachers can use this template to insert the necessary content for the School-assessed Coursework task.

The following content is included in this template:

- Relevant VCAA resources for the development of the Unit 3 Outcome 1 SAC task.
- The Unit 3 Outcome 1 statement.
- The Unit 3 Outcome 1 Key knowledge.
- The Unit 3 Outcome 1 Key skills.
- Details related to task development including:
  - conditions
  - scenario
  - solution requirements
  - solution designs
  - assessment (marking scheme)
- Details related to developing the final marking scheme for the task and determining the score out of 100 marks.

Use of commercial tasks

When referring to or using a commercially produced task teachers need to ensure that the tasks they develop are to be sufficiently modified from the original commercial task.

All commercially produced tasks must be cross-checked against the:

- outcome statement
- key knowledge
- key skills.

Also, for authentication reasons, the context (the background to the case study or scenario) and the content (solution requirements and designs) of the task must be significantly changed from the original publication each year. This involves the current year's commercial task as well as previous years and also any previous year's school-developed assessment tasks.

Designing the assessment task

Unit 3 Software Development – 2024

Outcome 1 Software development: programming – Template for developing an assessment task – Blank

Outcome 1

On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.

Assessment task development

Key knowledge	Key skills	VCAA Performance descriptors (Very high)
• methods for documenting a problem, need or opportunity • methods for determining solution requirements, constraints and scope • methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode	• interpret solution requirements and designs to develop working modules	• All solution requirements and designs are interpreted accurately to developing working modules.
• characteristics of data types • types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index) • formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats	• use a range of data types and data structures	• Comprehensive selection of relevant data types and data structures to develop working modules.
• a programming language as a method for developing working modules that meet specified needs • naming conventions for solution elements • processing features of a programming language, including classes, control structures, functions, instructions and methods • algorithms for sorting, including selection sort and quick sort • algorithms for binary and linear searching	• use and justify appropriate processing features of a programming language to develop working modules	• Comprehensive selection and use of relevant processing features of the programming language to develop all working modules. • Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.
• validation techniques, including existence checking, range checking and type checking • techniques for checking that modules meet design specifications, including trace tables and construction of test data	• develop and apply suitable validation, testing and debugging techniques using appropriate test data	• Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data. • Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.
• purposes and characteristics of internal documentation, including meaningful comments and syntax	• document the functioning of modules and the use of processing features through internal documentation	• All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.

Developing the assessment task

Unit 3 Software Development – 2024 Outcome 1 Software development: programming – Template for developing an assessment task – Plan			
Outcome 1 On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.			**Assessment task development – Planning for the case study** Create a scenario that is a real-world example that provides students with solution requirements and designs that will enable them to apply a range of functions and techniques using a programming language to develop and test working software modules. The outcome may be completed as three to six modules (tasks). Key content within the tasks should be based on the targeted key knowledge and key skills. The total number of the marks for the outcome should be out of 100.
Key knowledge	**Key skills**	**VCAA Performance descriptors (Very high)**	
• methods for documenting a problem, need or opportunity • methods for determining solution requirements, constraints and scope • methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode	• interpret solution requirements and designs to develop working modules	• All solution requirements and designs are interpreted accurately to developing working modules.	Content to be included in the assessment task should introduce students to a scenario. The scenario should provide solution requirements and designs for between three and six modules. These modules should vary in length and difficulty, providing students with sufficient opportunities to demonstrate their knowledge and to meet the requirements of the outcome. A range of appropriate design tools should be used. Students are not to complete designs themselves. Software modules can be small programs that may or may not form part of a larger software solution.
• characteristics of data types • types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index) • formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats	• use a range of data types and data structures	• Comprehensive selection of relevant data types and data structures to develop working modules.	The scenario with the solution requirements and designs should enable students to determine what data types and data structures they will need to use for the software modules.
• a programming language as a method for developing working modules that meet specified needs • naming conventions for solution elements • processing features of a programming language, including classes, control structures, functions, instructions and methods • algorithms for sorting, including selection sort and quick sort • algorithms for binary and linear searching	• use and justify appropriate processing features of a programming language to develop working modules	• Comprehensive selection and use of relevant processing features of the programming language to develop all working modules. • Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	The scenario with the solution requirements and designs should enable students to determine the appropriate selection and use of processing features, naming conventions and sorting and searching algorithms they will need to develop the software modules. An appropriate programming language should be used by the students (Refer to the Programming requirements document on the study page). Students are to justify and explain their selection of processing features and sorting and searching algorithms used to develop their working modules. This written justification and explanation could be included within the internal documentation or as a separate written report.
• validation techniques, including existence checking, range checking and type checking • techniques for checking that modules meet design specifications, including trace tables and construction of test data	• develop and apply suitable validation, testing and debugging techniques using appropriate test data	• Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data. • Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.	Students are to use and apply relevant data validation techniques to check all input data. A testing table is to be developed that involves the testing of all validation, objects and processing such as calculations, etc. The testing table should include columns for expected and actual output and show evidence of tests that work and don't work.
• purposes and characteristics of internal documentation, including meaningful comments and syntax	• document the functioning of modules and the use of processing features through internal documentation	• All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.	Students are to include internal documentation within their working modules. Internal documentation should state how the modules function and describe the code involving processing and validation.

Developing the marking scheme

Unit 3 Software Development – 2024 Outcome 1 Software development: programming – Developing a marking scheme – Sample			
Outcome 1 On completion of this unit the student should be able to interpret teacher-provided solution requirements and designs, and apply a range of functions and techniques using a programming language to develop and test working software modules.			**Developing a marking scheme – Marks allocated – 100** Refer to the key skills or the VCAA performance descriptors when developing a marking scheme for the assessment task. Determine the weighting of the marks out of 100 for each key skill or performance descriptor. When determining weightings consider the time that students will take to complete each task as well as the level of difficulty of each task. Marks should be allocated to ensure students can demonstrate a range of levels of performance in the task.
Key knowledge	**Key skills**	**VCAA Performance descriptors (Very high)**	
• methods for documenting a problem, need or opportunity • methods for determining solution requirements, constraints and scope • methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode	• interpret solution requirements and designs to develop working modules	• All solution requirements and designs are interpreted accurately to developing working modules.	Students are to interpret the solution requirements and designs for between three and six working modules. Possible number of marks – 10 marks
• characteristics of data types • types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays (single data type, integer index) and records (varying data types, field index) • formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats	• use a range of data types and data structures	• Comprehensive selection of relevant data types and data structures to develop working modules.	Students are to use a range of relevant data types and data structures within their software modules. Possible number of marks – 10 marks
• a programming language as a method for developing working modules that meet specified needs • naming conventions for solution elements • processing features of a programming language, including classes, control structures, functions, instructions and methods • algorithms for sorting, including selection sort and quick sort • algorithms for binary and linear searching	• use and justify appropriate processing features of a programming language to develop working modules	• Comprehensive selection and use of relevant processing features of the programming language to develop all working modules. • Comprehensive justification and explanation of how the selection of appropriate processing features of the programming language are used to develop working modules.	Students are to use appropriate processing features, naming conventions and sorting and searching algorithms to develop their software modules. A higher weighting of marks should be included to meet this key skill or performance descriptor. Possible number of marks – 40 marks Students are to justify and explain their selection of processing features and sorting and searching algorithms used to develop their working modules. Possible number of marks – 10 marks
• validation techniques, including existence checking, range checking and type checking • techniques for checking that modules meet design specifications, including trace tables and construction of test data	• develop and apply suitable validation, testing and debugging techniques using appropriate test data	• Comprehensive use of relevant data validation techniques are applied efficiently and effectively to check the reasonableness of all input data. • Comprehensive use of test data is expressed in a testing table, with both expected and actual output stated, and showing detailed evidence of debugging.	Students are to use and apply relevant data validation techniques to check all input data. Possible number of marks – 10 marks Students test their working modules using appropriate testing techniques. Possible number of marks – 10 marks
• purposes and characteristics of internal documentation, including meaningful comments and syntax	• document the functioning of modules and the use of processing features through internal documentation	• All software modules include comprehensive internal documentation regarding the functioning of modules and use of selected processing features.	Students are to include internal documentation within their working modules. Possible number of marks – 10 marks

Using commercial tasks (SAC)

Recommendations – In order to meet VCE Assessment principles

- If you decide to start off using a commercial task for ideas then you need to check it and modify it.
- Check the commercial task against the current study design. This includes the outcome statement, key knowledge and key skills. Be very watchful that the tasks address the current study design.
- Significantly alter the commercially-produced tasks each year in terms of context and content (even for this current year).
- Check the marking scheme/assessment rubric and ensure it meets the key skills and performance descriptors.
- Do the task yourself to ensure you are satisfied that it meets requirements and is suitable to your cohort.

VASS SAC dates for 2024

- **Unit 4 School-based Assessment – November**
  - Software Development: Unit 4 Outcome 2

Teachers should be aware of the dates for submission of scores into VASS in September and November. These dates are published in the 2024 Important Administrative Dates and Assessment Schedule, published annually on the VCAA website. vcaa.vic.edu.au/pages/schooladmin/admindates/index.aspx.

Contact

- **Phil Feain – Digital Technologies Curriculum Manager (VCAA)**
- **Ph: (03) 9059 5146**
- **Philip.Feain@education.vic.gov.au**

Authorised and published by the
Victorian Curriculum and Assessment Authority