L3 Computing - Unit 2 - Principles of Software Engineering

Relevant LINKS

BACK TO COMPUTING UNITS **[1]**

Handbook home page **[2]**

Overview

Computer Science at Platinum Level requires the candidate to completely understand and describe the basic principles of computing such as the use of approved design principles; quality control and maintenance processes and procedures; and quality programming principles including agile and more formal approaches. As a result of reviewing their work, they will be able to identify and use automated methods or alternative ways of working to improve programming and using computers. Unfamiliar aspects will require support and advice from other people.

A work activity will typically be 'straightforward or routine' because:

The task or context will be familiar and involve few variable aspects. The techniques used will be familiar or commonly undertaken.

Example of context – designing, planing, implementing and testing a program for controlling a physical system or solving a complex problem.

Support for the assessment of this award

Example of typical Computing work at this level (Coming Soon)

Assessor's guide to interpreting the criteria

General Information

QCF general description for Level 3 qualifications

- Achievement at Level 3 (EQF Level 4) reflects the ability to identify and use relevant understanding, methods and skills to complete tasks and address problems that, while well defined, have a measure of complexity. It includes taking responsibility for initiating and completing tasks and procedures as well as exercising autonomy and judgment within limited parameters. It also reflects awareness of different perspectives or approaches within an area of study or work.
- Use factual, procedural and theoretical understanding to complete tasks and address problems that, while well defined, may be complex and non-routine.
- Identify, select and use appropriate skills, methods and procedures.
- Use appropriate investigation to inform actions.
- Review how effective methods and actions have been.
- Take responsibility for initiating and completing tasks and procedures, including, where relevant, responsibility for supervising or guiding others.
- Exercise autonomy and judgement within limited parameters information and ideas.

Requirements

- Standards must be confirmed by a trained Level 3 Assessor or higher.
- Assessors must at a minimum record assessment judgements as entries in the online mark book on the INGOTs.org certification site.
- Routine evidence of work used for judging assessment outcomes in the candidates' records of their day to day work will be available from their e-portfolios and online work.
- Assessors should ensure that relevant web pages are available to their Account Manager on request by supply of the URL.
- When the candidate provides evidence of matching all the criteria to the specification, subject to the guidance below, the assessor can request the award using the link on the certification site. The Account Manager will request a random sample of evidence from candidates' work that verifies the assessor's judgement.
- When the Account Manager is satisfied that the evidence is sufficient to safely make an award, the candidate's success will be confirmed and the unit certificate will be printable from the web site.
- Each unit at Level 3 has recommended guided learning hours based on time required to complete by an average learner.

Assessment Method
Assessors can score each of the criteria N, L, S or H. N indicates no evidence and it is the default setting. L indicates some capability but some help still required to meet the standard. S indicates that the candidate can match the criterion to its required specification in keeping with the overall level descriptor. H indicates performance that goes beyond the expected in at least some aspects. Candidates are required to achieve at least S on all the criteria to achieve the full unit award.

Expansion of the assessment criteria

1. The candidate will understand the role of the target audience

1.1 I can describe the rationale for release early, release often

Candidates should be familiar with the work of Eric Raymond in the Cathedral and the Bazaar [3].

Evidence: portfolios.

Additional information and guidance

Candidates should read about the history and background that led to the release early and release often philosophy. They should be able to refer to specific examples of software produced on this and alternative models and debate the advantages and disadvantages of different approaches.

1.2 I can explain principles of user interface design.

Candidates should provide a study of user interface design explaining key things learnt for their own designs.

Evidence: portfolios.

Additional information and guidance

Candidates should provide evidence of significant research including the use of web search but also studies of popular sites analysing them for strengths and weaknesses. Which features would they use and why?

1.3 I can receive user feedback and act positively.

Candidates should actively seek feedback on their work from third parties and act positively on the information provided.

Evidence: portfolios.

Additional information and guidance

Candidates should regularly seek feedback from peers and other third parties and be prepared to give the same. Feedback should be constructive and objective and lead to improvement. Evidence should span all their projects from small exercises to their large scale project.

1.4 I can compare the user role in a range of software development models.

Candidates should carry out a study of software development models from the point of view of user involvement.

Evidence: Portfolios.

Additional information and guidance

In some software development communities the users are often project members and the distinction becomes blurred. In some cases it is easy for users to contribute feedback and even file bug reports and track them. In others it might not be possible to do this at all. Does the maxim “many eyes make bugs shallow” stand up to evidence. This could be a good subject for a team approach to gather evidence, pool resources and discuss final conclusions. This is to be encouraged as long as assessors can be clear that individuals meet the criterion.

1.5 I can describe methods for providing feedback to users from errors in the code.

Candidates should do a critical survey of the error messages provided to users in commonly used software and use the outcomes to inform their own coding projects.

Evidence: Portfolios.

Additional information and guidance

There is still a lot of poor practice in commercial software applications when it comes to error messages. A good starting point for research is here [4]. With the advent of the web, more applications are using the web for help files. This could also be extended to error messages from applications. If there is a knowledge base for a software application in the web, it should be possible to automatically search it when an error occurs and add to it useful information about the circumstances that generated the error. If it is an error due to a bug in the code, it can then be automatically routed to the bug tracking system and linked to a user log file so that programmers could see what caused the bug and eliminate it. If it was a user error it can inform design to make the user error impossible. The overall goal should be to eliminate the need for error messages but where they occur they should be as useful as possible to the user. Linking errors to an online content management system makes it easy to edit the messages, provide language translations etc.

2. The candidate will understand strategies for maintaining quality

2.1 I can identify design techniques to reduce risk

Candidates should research and identify software design methods that will reduce risk to quality in their current or future projects.

Evidence: from portfolios.

Additional information and guidance

Candidates should use their research to help them decide on their approach to quality assurance in their software development. Their evidence should show that they have considered a wide range of

views and opinions.

2.2 I can explain a sound testing strategy.

Testing strategies for software development should be based on accepted industry practice.

Evidence: documentation in portfolios.

Additional information and guidance

Candidates should show that they have researched testing methods and considered a range of views. They should come to conclusions about how they will implement testing and they should be prepared to modify these in the light of practical application.

2.3 I can establish clear communication channels with critical reviewers.

Candidates should develop practical communication channels with reviewers of their work using collaborative technologies to aid the process.

Evidence: assessor observations and portfolios.

Additional information and guidance

Candidates should set up, organise and use appropriate collaborative technologies to make review and feedback straightforward and regular.

2.4 I can explain and demonstrate the importance of courage and persistence in solving problems.

Candidates should understand and demonstrate these attitudes in the pursuit of quality.

Evidence: Assessor observations, documentation in portfolios.

Additional information and guidance

Candidates should keep a diary or Blog to document their use of time in solving problems related to quality in their projects.

2.5 I can demonstrate quality strategies through small scale projects.

Candidates should trial the quality assurance strategies they have researched in preliminary small scale projects in preparation for their main project.

Evidence: from documented small scale projects.

Additional information and guidance

The small scale projects should be used as a practical vehicle for trying out research findings in the more theoretical aspects of the work and as a preparation for their large scale project. Quality will be enhanced in a large scale project by understanding the issues through practical experience in small scale projects.

3. The candidate will adopt suitable methods to match circumstances

3.1 I can compare procedural and object oriented programming

The candidate will be able to make clear judgements about programming methods based on evidence.

Evidence: from assessor judgements and content of portfolios.

Additional information and guidance

Candidates should carry out small scale research into programming methods (formally referencing sources) and gain some experience of procedural and object oriented programming in small scale projects so they have some practical basis for their judgements. Initially they will need considerable guidance but the aim is to get them more experience and therefore more confidence to make judgements themselves.

3.2 I can compare formal and agile methods.

Candidates should have enough knowledge of wider opinions on methods to make objective comparisons analysing strengths and weaknesses.

Evidence: Assessor observations and contents of portfolios.

Additional information and guidance

Candidates should carry out small scale research into programming methods (formally referencing sources) and gain some experience of agile and formal methods in small scale projects so they have some practical basis for their judgements. Initially they will need considerable guidance but the aim is to get them more experience and therefore more confidence to make judgements themselves.

3.3 I can specify a documentation strategy.

The candidate will use research evidence to specify their preferred method of documenting their work and justify it with evidence.

Evidence: Assessor observations and portfolios.

Additional information and guidance

Candidates should carry out small scale research into documentation methods (formally referencing sources) and use documentation of small scale projects so they have some practical basis for their judgements. They should see differences in the documentation philosophies in agile and formal approaches. There are also issues related to openness of the source code, user interface design and how interactive help works. the power of searching web based documentation has to be weighed against inconvenience it the internet is not readily available or expensive to access.

3.4 I can describe an open source community project and its methods.

Candidates should have studied an open source community project in some detail.

Evidence: Assessor observation and contents of portfolios.

Additional information and guidance

Candidates should choose an open source project to study. Sourceforge is a good starting point. As a minimum they should understand the governance of the project e.g. Apache OpenOffice has a project management committee within the Apache Software Foundation and uses Apache software licensing for the release products whereas LibreOffice is managed by The Document Foundation and releases LibreOffice under the LGPLv3+ and MPL. This means code can be taken from Apache OpenOffice and used in LibreOffice but code can not be taken from LibreOffice to be used in OpenOffice. This could and should be an opportunity to link to criteria on intellectual property. If members of a group study different Open Source projects it would be a useful exercise to present each project to the whole group to widen understanding. If candidates use this learning to make a contribution to an Open Source project using their large scale project, so much the better.

3.5 I can explain the different demands of large scale and small scale projects.

Candidates should provide a detailed explanation of the range of demands of small and large scale projects based on analysis of experienced views and their own empirical experience.

Evidence: portfolios.

Additional information and guidance

Candidates should carry out small scale research to determine views on these issues. Evidence might include what goes wrong on large scale government IT projects? Why are they so expensive? At what point does a small scale project become large scale? How does the number of developers involved affect a project's efficiency? Can small scale projects scale to become large scale? - e.g. the Linux Kernel was started by one developer (Linus Torvalds) in 1991 and is now estimated to be used in nearly half of all consumer devices with around 200 active developers and 15 million lines of code. The main outcome from this criterion should be a rational appreciation of considerations for project planning.

Moderation/verification

The assessor should keep a record of assessment judgements made for each candidate and make notes of any significant issues for any candidate. They must be prepared to enter into dialogue with their Account Manager and provide their assessment records to the Account Manager through the online mark book. They should be prepared to provide evidence as a basis for their judgements through reference to candidate e-portfolios and through signed witness statements associated with the criteria matching marks in the on-line mark book. Before authorizing certification, the Account Manager must be satisfied that the assessors judgements are sound.

Links

[1] http://theingots.org/community/Computing_qualification_info_units
[2] https://theingots.org/community/sites/default/files/uploads/common/Handbooks/Computing/L3_OpenSystems_Computing_v2%20.pdf
[3] http://www.catb.org/esr/writings/cathedral-bazaar/
[4] http://http://www.developsense.com/essays/AReviewOfErrorMessages.html