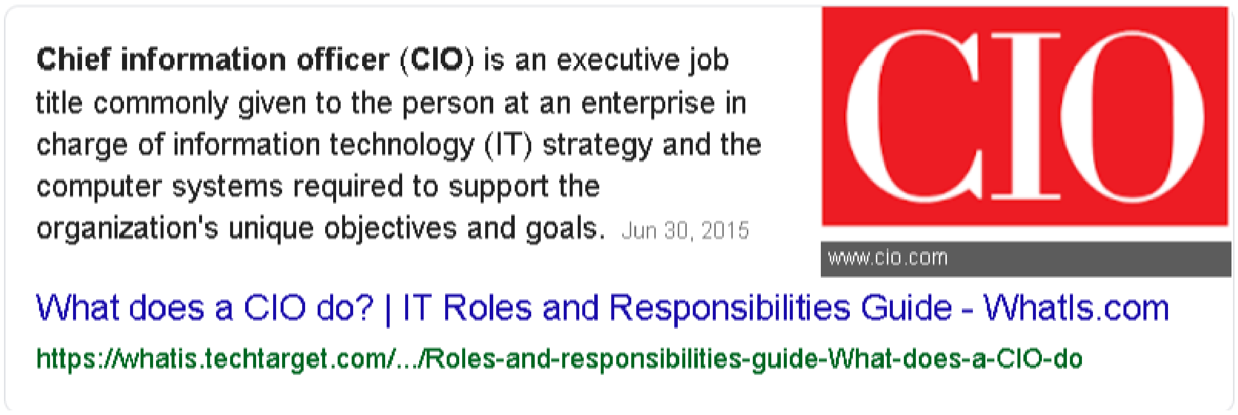
The CIO cheat sheet: 20 testing and QA technical terms you should know



Chief information officer (**CIO**) is an executive job title commonly given to the person at an enterprise in charge of information technology (IT) strategy and the computer systems required to support the organization's unique objectives and goals. Jun 30, 2015

www.cio.com

What does a CIO do? | IT Roles and Responsibilities Guide - WhatIs.com
https://whatis.techtarget.com/.../Roles-and-responsibilities-guide-What-does-a-CIO-do

As a CIO it's important that you're informed of the essential terms in the industry of **software testing** and **QA**. Not only will it make your job easier but it will establish your credibility instantly if you can understand and respond accordingly to your testing team.

Here are 20 technical terms that you should know, including five from the up and coming world of **agile testing**.

So look through this list, update your lingo and begin to implement better software for your business with fewer bugs and thus with a lower price tag.

1. **Quality assurance:** QA is the process of routine checks carried out to ensure that a product or service is meeting its standard requirements.
2. **Software testing:** Software testing is a process of analysing possible risks in a function or application then designing and running tests to prove that those risks have not been introduced by way of defects
3. **Shift left and Compress:** 'Shift left' means carrying out development and test activities earlier in the project life cycle in order to detect defects sooner. 'Compress' relates to reducing the total number of defects introduced by a focused defect prevention process, as Simon Barkley explainsin our video.
4. **Residual product risk:** In software testing, this is the product risk (risk of failure in production) remaining in an application after test execution has completed. It is expressed in terms of likelihood and impact and should be used as an input to help decide if a product should be released to the next stage of development or the live environment.

5. **Cost of poor quality:** If all systems, processes and products were perfect, then these expenses and costs would disappear. In other words they are costs that are caused by defective testing materials.
6. **Defect prevention:** Defect prevention is a process the aim of which is to identify the root cause of defects and to implement changes to help prevent them from recurring.
7. **Test condition: Test Case: Test Script: Test Procedure: Test scenario:** Simply put these are various artefacts that are used by testers to define the tests they will execute
8. **Test Traceability:** Traceability is the capability to link test conditions back to their origins in the **test basis** (the document or source of information needed to write the **test cases, such as requirements**). This can be horizontal through all test documentation for a given test level or vertical through layers of development documentation.
9. **Validation:** This determines that right product is being built, for example, a review of the requirements and the design to ensure the two are aligned
10. **Verification:** Verification determines whether the product has been built correctly and involves carrying out tests on specific functions or portions of a product. This is a more inward looking process than **validation** – it's focused on the software itself rather than what it does for the business.
11. **Regression testing:** When programs are updated, modified or added to, regression testing checks no existing functionality has been damaged.
12. **Continuous integration:** Isolated changes are added to a larger code base and then immediately tested and reported upon – this gives rapid feedback so that any defects can be easily treated and then reintegrated effectively.
13. **Test requirements:** The definition of the functions, both in terms of what it should do and how it should do it e.g. speed of responses. The necessary functions that software can carry out and the capability it should possess. It also details what the software should do for the business.
14. **Requirements-based:** This form of testing focusses on integrating testing throughout the development lifecycle concentrating mainly on the **Business Requirements**.
15. **TMMi:** Test Model Maturity integration is an assessment method that charts the maturity of software testing processes and is used by many organizations to identify process improvement activity An organization, as a whole, will advance its testing processes from ***ad hoc*** to **managed**, then **defined**, up to **measured** and finally **optimized**.

And five from agile testing…

You can read about the agile vs traditional debate in our blog, as well as other trends in the software-testing world.

18. **Velocity:** A metric that predicts how much work an Agile software development team can successfully complete within a sprint or iteration of defined length.

19. **Technical debt:** This accrues when a team decides ‘to fix X later’. Any activity that is postponed is considered debt; it is ‘what you owe the product’, as Joanna Rothman describes it, such as missing unit tests.
20. **Definition of ‘Done’:** This provides the **‘exit criteria’** (the requirements which must be met to complete a specific task) and therefore the conditions under which testers can mark a user story as complete.
21. **Burn up chart:** Burn up allows you to visualise how much work has been finished and the total amount of work required for completion.
22. **Burn down chart:** A burn down chart, conversely, demonstrates what is left to be completed in the project.

https://www.experimentus.com/blog/cio-cheat-sheet-20-testing-qa-technical-terms-know/