Software Development

A-Level Computer Science · Vocabulary in context

Word	Example
development life cycle /dɪ'veləpmənt laif 'saɪkl/	The **development life cycle** goes from plan to test to use.
requirements /rɪ'kwaɪəmənts/	The **requirements** list what the program must do.
waterfall /'wɔ:təfɔ:l/	In the **waterfall** model, each stage ends before the next one starts.
maintenance /'meɪntənəns/	**Maintenance** fixes and improves the program after it is in use.
iterative model /'ɪtərətɪv 'mɒdl/	The **iterative model** builds the program in small steps.
Rapid Application Development /'ræpɪd ˌæplɪ'keɪʃn dɪ'veləpmənt/	**Rapid Application Development** builds quick versions to show the user.
prototype /'prəʊtəʊtaɪp/	A **prototype** is an early version of the program.
Agile /'ædʒaɪl/	**Agile** teams work in short cycles and talk to the user often.
structure chart /'strʌktʃə tʃɑ:t/	A **structure chart** shows how the modules link.
parameters /pə'ræmɪtəz/	The procedure has two **parameters**: a name and an age.
pseudocode /'su:dəʊkəʊd/	Write the algorithm in **pseudocode** first.
test plan /test plæn/	The **test plan** lists each test and the expected result.

exercises.lol exercises.com.cn

Word	Example
implementation /ˌɪmplɪmən'teɪʃn/	**Implementation** means writing the code.
boundary data /'baʊndəri 'deɪtə/	For ages 0 to 120, the **boundary data** are 0 and 120. 0 120 0 120
acceptance testing /ək'septəns 'testɪŋ/	In **acceptance testing**, the user checks that the program meets the requirements.
hierarchical decomposition /haɪə'rɑ:kɪkl ˌdi:kɒmpə'zɪʃn/	**Hierarchical decomposition** breaks a task into levels of smaller tasks.
decomposition /ˌdi:kɒmpə'zɪʃn/	**Decomposition** breaks a big problem into small parts.
subroutines /'sʌbru:ti:nz/	Split the program into small **subroutines**.
state-transition diagram /steɪt træn'sɪʃn 'daɪəgræm/	A **state-transition diagram** shows how a traffic light changes.
states /steɪts/	A traffic light has three **states**: red, amber and green.
syntax error /'sɪntæks 'erə/	A missing bracket is a **syntax error**.
run-time error /rʌn taɪm 'erə/	Dividing by zero causes a **run-time error**.
logic error /'lɒdʒɪk 'erə/	A **logic error** gives the wrong answer but does not stop the program.
dry run /draɪ rʌn/	Do a **dry run** with a pen and paper.
trace table /treɪs 'teɪbl/	A **trace table** shows the value of each variable at each step.
walkthrough /'wɔ:kθru:/	In a **walkthrough**, the team reads the code together.

Word	Example
white-box testing /waɪt bɒks 'testɪŋ/	**White-box testing** tests every path through the code.
black-box testing /blæk bɒks 'testɪŋ/	**Black-box testing** checks inputs and outputs only.
integration testing /ˌɪntɪ'greɪʃn 'testɪŋ/	**Integration testing** checks that the modules work together.
alpha testing /'ælfə 'testɪŋ/	**Alpha testing** is done by the developers.
beta testing /'biːtə 'testɪŋ/	**Beta testing** is done by real users before the release.
stub /stʌb/	A **stub** stands in for a module that is not written yet.
corrective maintenance /kə'rektɪv 'meɪntənəns/	**Corrective maintenance** fixes bugs.
test strategy /test 'strætədʒi/	The **test strategy** says how the program will be tested.
normal data /'nɔːml 'deɪtə/	For ages 0 to 120, 15 is **normal data**. 0 120 15
abnormal data /əb'nɔːml 'deɪtə/	For ages 0 to 120, “cat” is **abnormal data**. 0 120 ”cat”
extreme data /ek'striːm 'deɪtə/	For ages 0 to 120, 0 and 120 are **extreme data**. 0 120 0 120
perfective maintenance /pə'fektɪv 'meɪntənəns/	**Perfective maintenance** makes the program faster or easier to use.
adaptive maintenance /ə'dæptɪv 'meɪntənəns/	**Adaptive maintenance** changes the program for a new phone or law.

exercises.lol exercises.com.cn

Word	Example
regression testing /rɪ'greʃn 'testɪŋ/	**Regression testing** checks that old features still work.