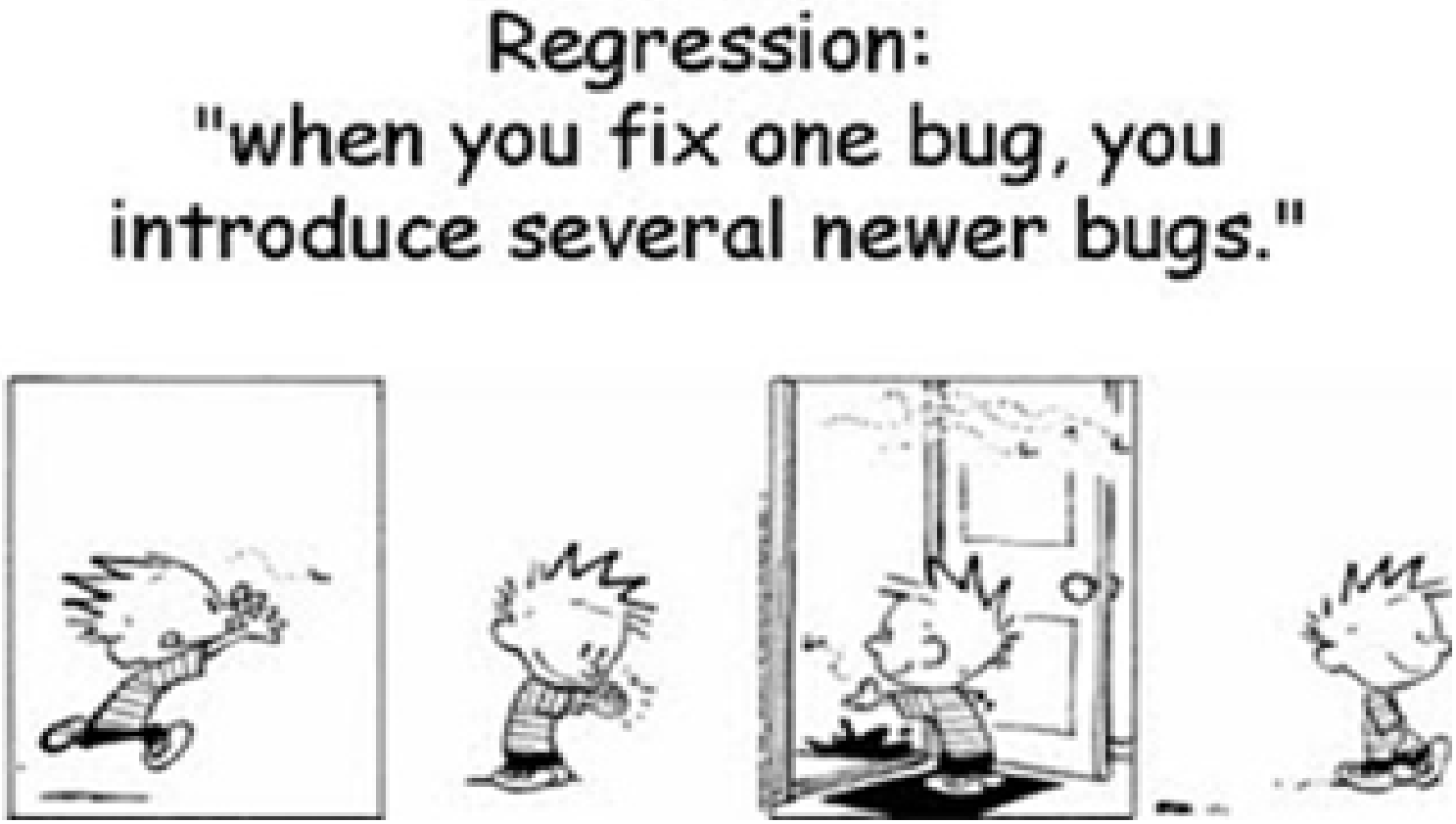
Session 13
Regression Testing

chengbaolei@suda.edu.cn

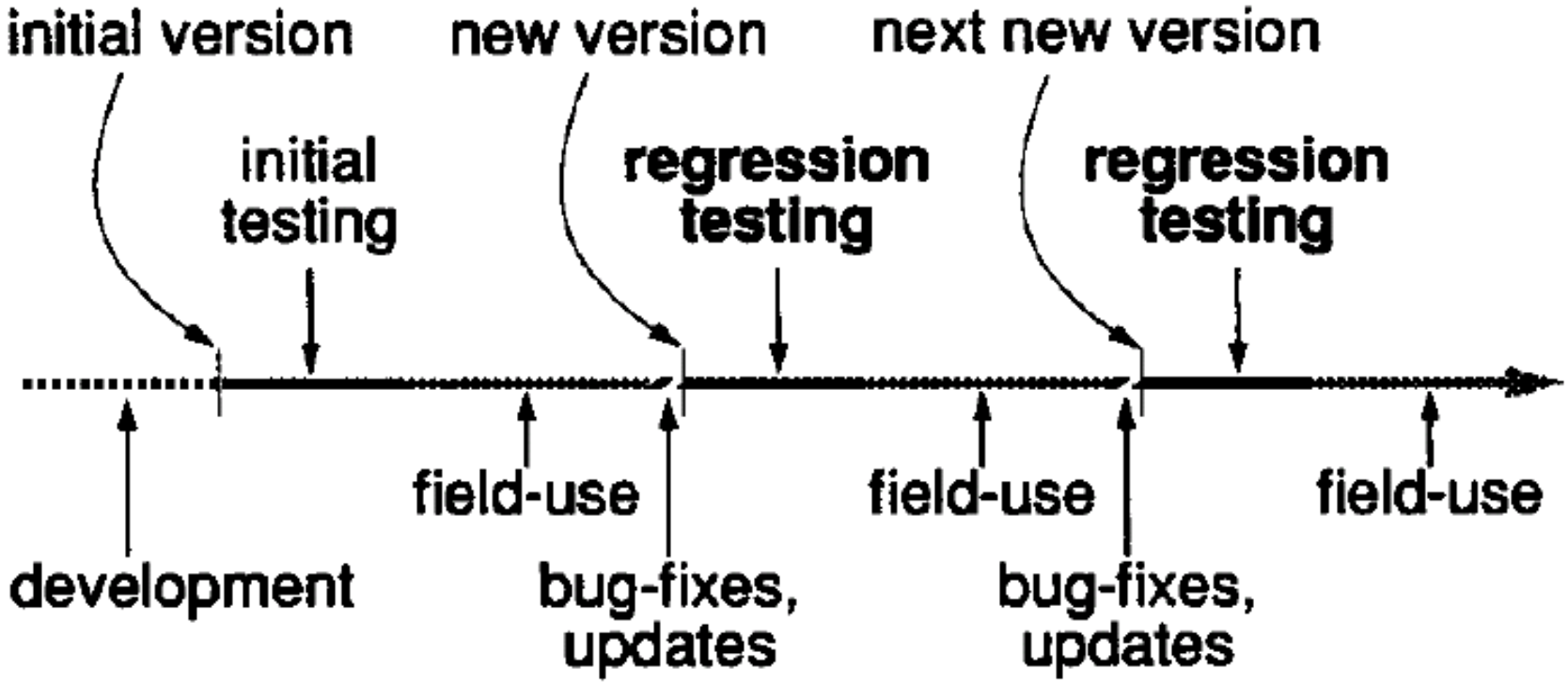
In Session 12:

- Unit testing
- Integration testing
- System testing
- Acceptance testing
- Regression testing

Why Regression Test?



Why Regression Test?



expensive and frequently executed

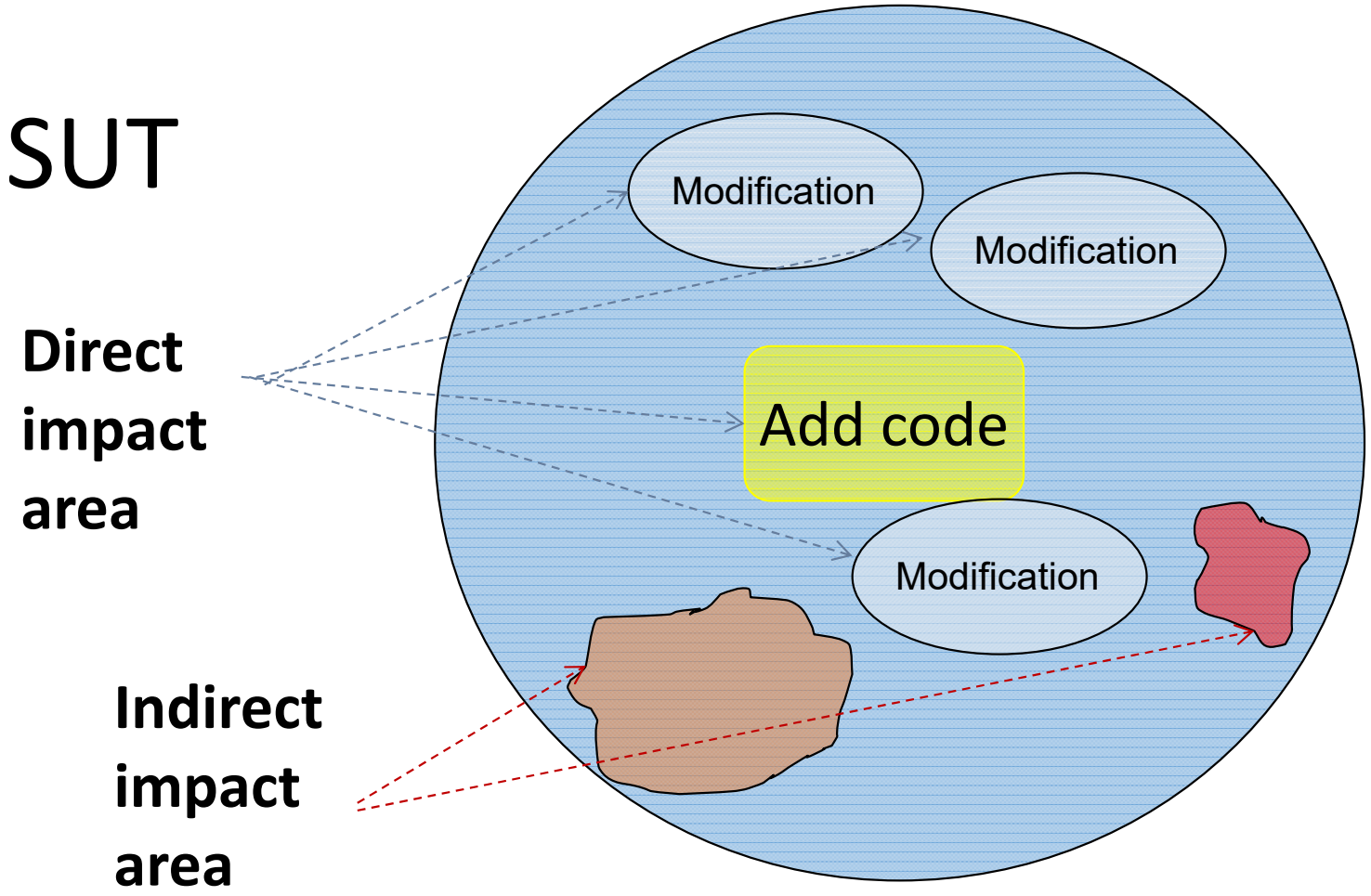
What is Regression Testing?

- Whenever changes happen to software, regression testing is performed to ensure that these changes are correct and they do not adversely affect the existing software.
- It is necessary to perform regression testing when:
  - Defect fixing.
  - Change is in requirements and code is modified according to the requirement.
  - New feature is added to the software.

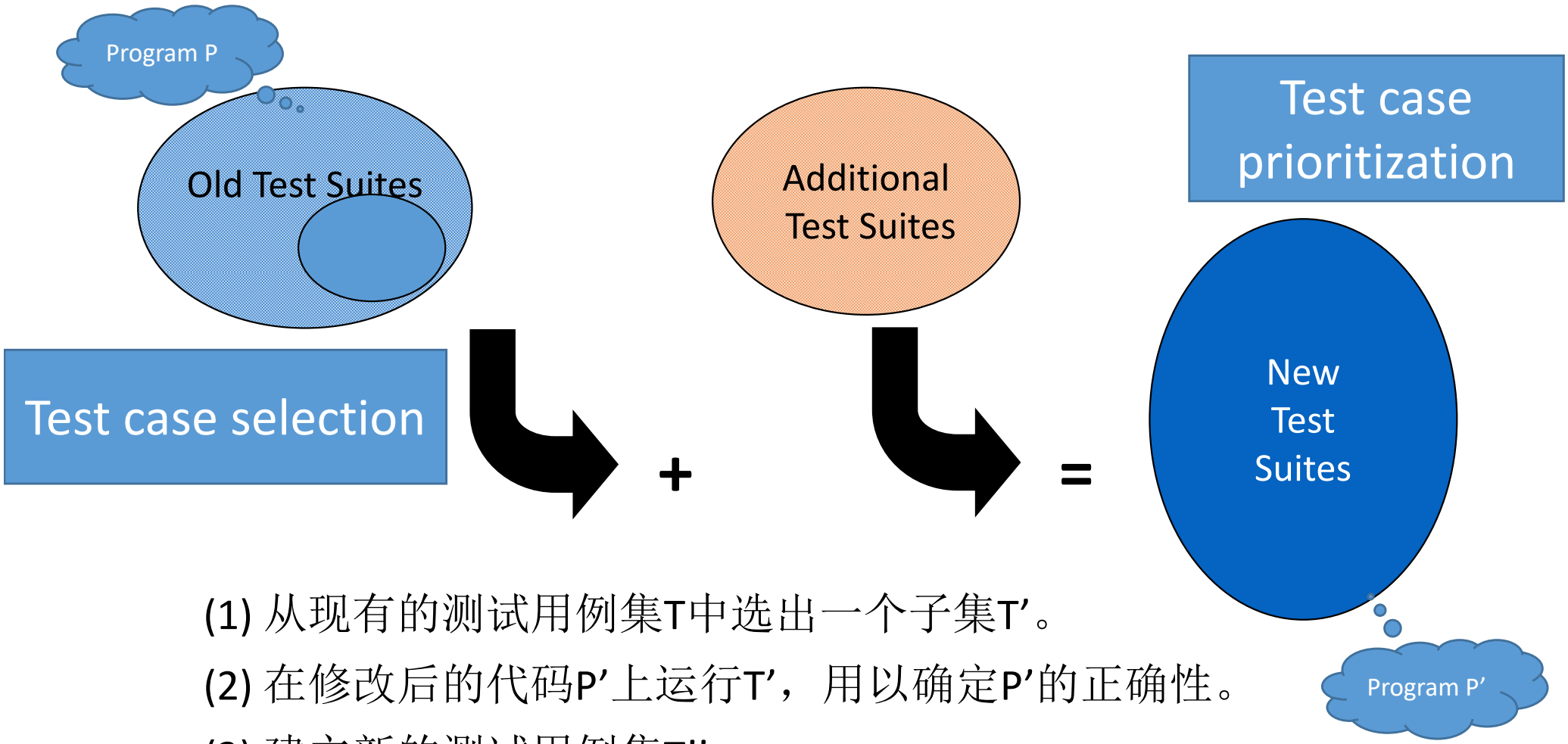
How to do Regression Testing

- How much re-testing is needed?
  - Retest-all?
  - A set of selected test.
  - The test cases that failed due to the defects should be included for future regression testing.
  - An impact analysis is done to find out what areas may get impacted due to those defect fixes. Based on the impact analysis, some more test cases are selected to take care of the impacted areas.

How to do Regression Testing



How to do Regression Testing



(1) 从现有的测试用例集T中选出一个子集T'。

(2) 在修改后的代码P'上运行T'，用以确定P'的正确性。

(3) 建立新的测试用例集T''。

(4) 在修改后的代码P'上运行T''，用以确定P'的正确性。

(5) 由T, T', T''建立P'的测试历史和测试用例集T'''

How to do Regression Testing

Definition 1. ***Test Suite Minimisation Problem***

Given: *A test suite, T, a set of test requirements $\{r_1, \ldots, r_n\}$, that must be satisfied to provide the desired 'adequate' testing of the program, and subsets of T, $T_1, \ldots, T_n$, one associated with each of the r_is such that any one of the test cases t_j belonging to T_i can be used to achieve requirement r_i.*

Problem: *Find a representative set, T', of test cases from T that satisfies all r_is.*

Definition 2. ***Test Case Selection Problem***

Given: *The program, P, the modified version of P, P', and a test suite, T.*

Problem: *Find a subset of T, T', with which to test P'.*

Definition 3. ***Test Case Prioritisation Problem***

Given: *a test suite, T, the set of permutations of T, PT, and a function from PT to real numbers, $f : PT \rightarrow \mathbb{R}$.*

Problem: *to find $T' \in PT$ such that $(\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$.*

Test case minimization

- Aim to identify redundant test cases and to remove them from the test suite in order to reduce the size of the test suite
- Note that the minimal hitting set formulation of the test suite minimization problem depends on the assumption that each r_i can be satisfied by a single test case. In practice, this may not be true.

Test case minimization

Test Case	Testing Requirements					
	r_1	r_2	r_3	r_4	r_5	r_6
t_1	X	X	X			
t_2	X			X		
t_3		X			X	
t_4			X			X
t_5					X	

定义 1(常规 Set Cover[3]) 给定一个由 n 个元组组成的有限集合 $Q=\{Q_1,\cdots,Q_n\}$，元组中的元素取自符号集 $C=\{c_1,\cdots,c_m\}$，目标是寻找集合 Q 的一个子集 Q'，使得 $\bigcup_{Q_i\in Q'}=C$ 且集合 Q' 含有的元组个数最少。

最小化→NP-Hard

An example

Test Case	Testing Requirements					
	r_1	r_2	r_3	r_4	r_5	r_6
t_1	X	X	X			
t_2	X			X		
t_3		X			X	
t_4			X			X
t_5					X	

Greedy: select the test case that satisfies the maximum number of test requirements. *t1-t2-t3-t4*

Additional Greedy: select the test case that satisfies the maximum number of unsatisfied test requirements. *t1-t2-t3/t5-t4*

GE: first select all essential test cases in the test suite; for the remaining test requirements, use the additional greedy algorithm, i.e. select the test case that satisfies the maximum number of unsatisfied test requirements. *t2-t4-t3*

GRE heuristic: first remove all redundant test cases in the test suite, which may make some test cases essential; then perform the GE heuristic on the reduced test suite. *t2-t3-t4*

Exercise

	r1	r2	r3	r4	r5
t1	X				
t2		X	X	X	X
t3		X		X	X
t4			X	X	X
t5			X		
t6				X	X

Test case minimization

- The effectiveness of the minimization:

$$(1 - \frac{\text{number of test cases in the reduced test suite}}{\text{number of test cases in the original test suite}}) * 100\%$$

- The impact of test suite minimization was measured by calculating the reduction in fault detection effectiveness as follows:

$$(1 - \frac{\text{number of faults detected by the reduced test suite}}{\text{number of faults detected by the original test suite}}) * 100\%$$

Test case selection

- Reduce the number of test cases and satisfy the testing requirements.
- Various test criteria
  - Fault detection ability
  - Code coverage
  - Time cost
  - Fault detection history
  - …

VS. test case minimization

Definition 2. ***Test Case Selection Problem***

Given: *The program, P, the modified version of P, P', and a test suite, T.*

Problem: *Find a subset of T, T', with which to test P'.*

Test case selection

- Classify test cases into five classes. The first three classes consist of test cases which already exist in T.
  - Reusable: Reusable test cases execute the parts of the program that remain unchanged and common to program P and P'.
  - Re-testable: Re-testable test cases execute the parts of P that get changed in P'.
  - Obsolete
    - 1) Their input/output relation get changed due to changes in specifications
    - 2) They no longer test what they were designed to test due to modifications in the program
    - 3) They are "structurally" test cases that no longer contribute to structural coverage of the program.

Test case selection

- The remaining two classes consist of test cases that are to be generated for the regression testing of P’.
  - New-structural: New-structural test cases test the modified program constructs that provide structural coverage of the modified parts in P’.
  - New-specification: New-specification test cases test the modified program specifications.

Test case selection

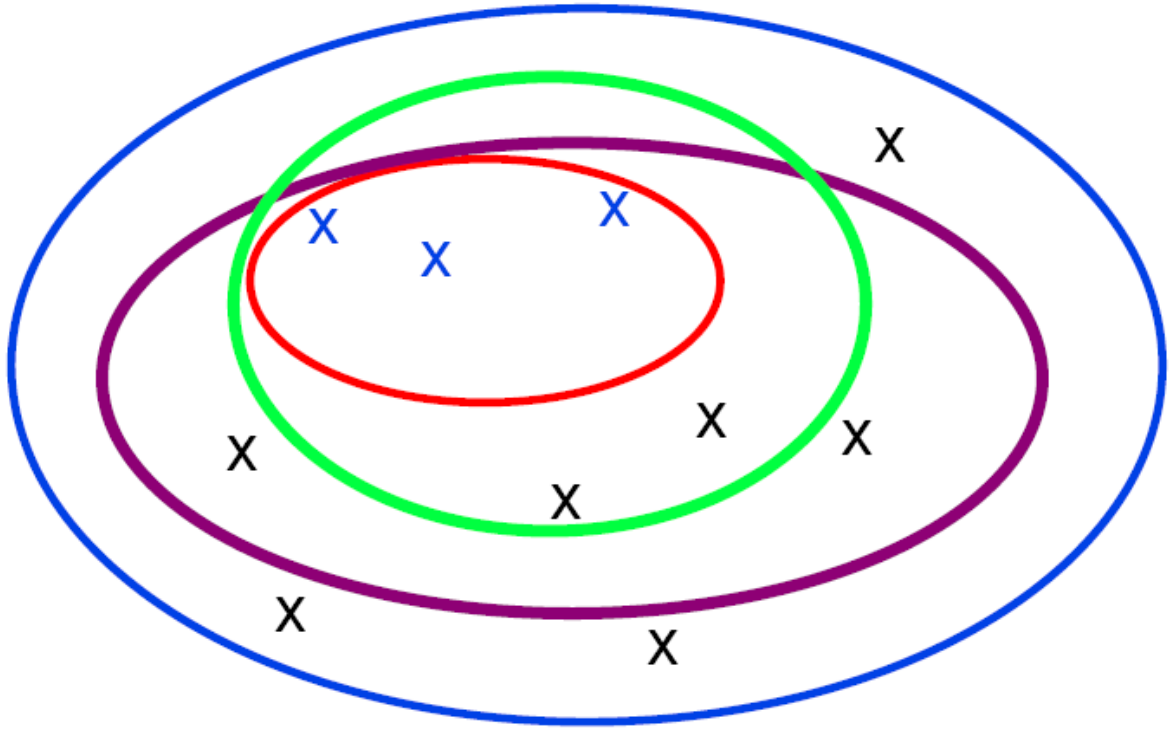
- Two execution traces *ET*(*P*(*t*)) and *ET*(*P*'(*t*)) are *equivalent* if they have the same lengths and if, when their elements are compared from first to last, the text representing the pairs of corresponding elements is lexicographically equivalent.
- Two text strings are *lexicographically equivalent* if their text is identical.
- Test *t* is *modification-traversing for P and P*' if and only if *ET*(*P*(*t*)) and *ET*(*P*' (*t*)) are nonequivalent.

Rothermel G , Harrold M J . A safe, efficient regression test selection technique[J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(2):173-210.

Test case selection

- Fault-revealing Test Cases
  - A test case t ∈ T is said to be fault-revealing for a program P, if and only if by executing this on program P, there is a failure due to incorrect output.
- Modification-revealing Test Cases
  - A test case t ∈ T is considered to be modification-revealing for P and P′ if it produces different outputs for P and P′.
- Modification-traversing Test Cases
  - A test case t ∈T is modification-traversing for P and P′ if and only if the execution traces of t on P and P′ are different.
  - A test case t is said to be modification-traversing if it executes the modified regions of code in P′.

Test case selection



Some Alternatives:

Fault revealing (Conservative and Precise)

-impossible to compute

Modification revealing (Conservative, but not precise)

-hard to compute

Traversal revealing -easier to compute

Retest all (Conservative, but not precise)- trivial to compute

Test case selection

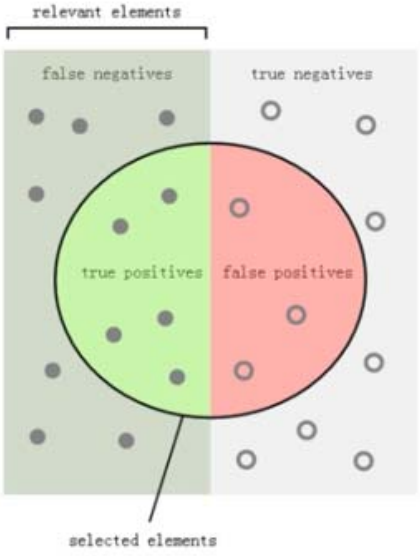
- Inclusive, Precise and Safe Regression Test Cases:
  - Inclusive: Inclusiveness measure the extent to which we select modification revealing test cases from the original test suit T. Expressed as (m/n)*100.
  - Safe: selects all the modification revealing test cases. In other ways, if it is 100% inclusive then we said it a safe technique.
  - Precision: measure the extent to which regression test selection ignores the non modification revealing test cases.

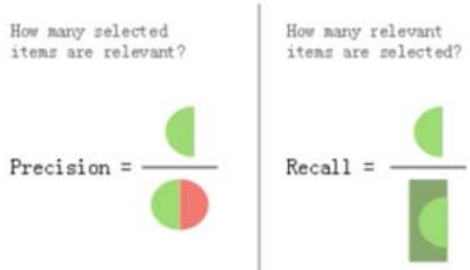
Test case selection

- Evaluation Models
  - the number of test cases in the original test suite is *T*, F_T test cases of them are fault-revealing.
  - A test selection technique selects a subset of *T'* test cases, in which $F_{T'}$ are fault-revealing.

$$Recall = F_{T'} / F_T * 100\%.$$

$$Precision = F_{T'} / T' * 100\%.$$





Precision:被分类器挑选(selected)出来的正样本究竟有多少是真正的正样本!

Recall:在全部真正的正样本里面分类器挑选了多少个!

Test case selection

- Effective Regression Tests can be done by selecting following test cases -
  - Test cases which have frequent defects
  - Functionalities which are more visible to the users
  - Test cases which verify core features of the product
  - Test cases of Functionalities which has undergone more and recent changes
  - All Integration Test Cases
  - All Complex Test Cases
  - Boundary value test cases
  - Sample of Successful test cases
  - Sample of Failure test cases

Test case Prioritization

Definition 3. ***Test Case Prioritisation Problem***

Given: *a test suite, T, the set of permutations of T, PT, and a function from PT to real numbers, $f : PT \rightarrow \mathbb{R}$.*

Problem: *to find $T' \in PT$ such that $(\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$.*

Test case prioritization seeks to find the ideal ordering of test cases for testing, so that the tester obtains maximum benefit, even if the testing is prematurely halted at some arbitrary point.

Test case Prioritization

- Average Percentage of Fault Detection (APFD) metric.
  - Higher APFD values denote faster fault detection rates.
  - APFD can be calculated as the area below the plotted line.
  - let *T* be the test suite containing *n* test cases and let *F* be the set of m faults revealed by *T*. For ordering T_0, let TF_i be the order of the first test case that reveals the *i*th fault.

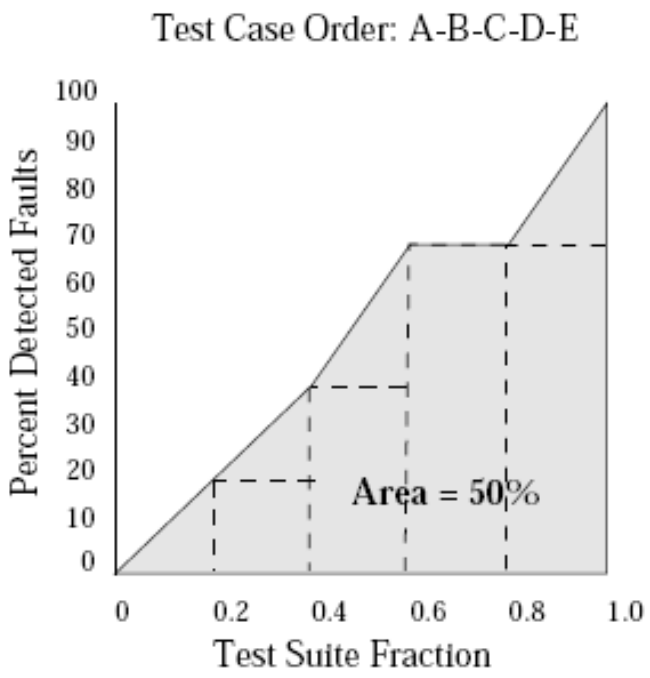
$$APFD = 1 - \frac{TF_1 + \ldots + TF_m}{nm} + \frac{1}{2n}$$

Test case Prioritization

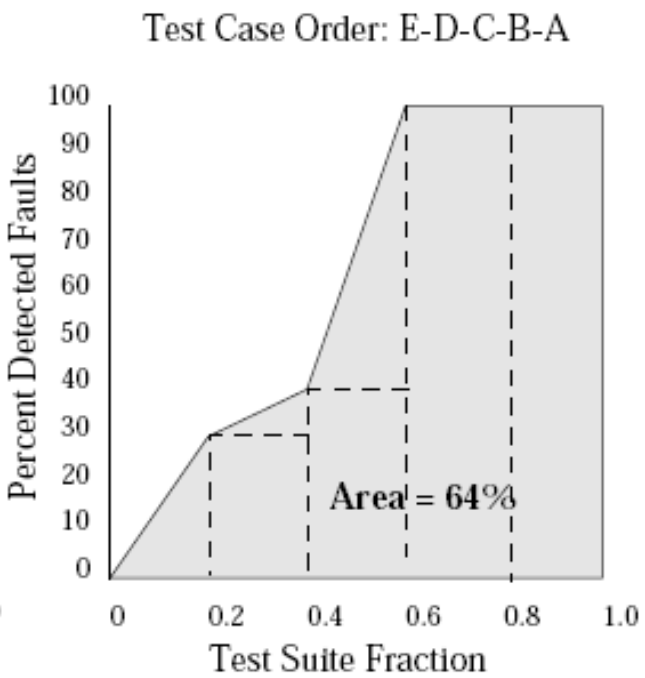
test	fault									
	1	2	3	4	5	6	7	8	9	10
A	x				x					
B	x				x	x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

$$APFD = 1 - \frac{TF_1 + \ldots + TF_m}{nm} + \frac{1}{2n}$$

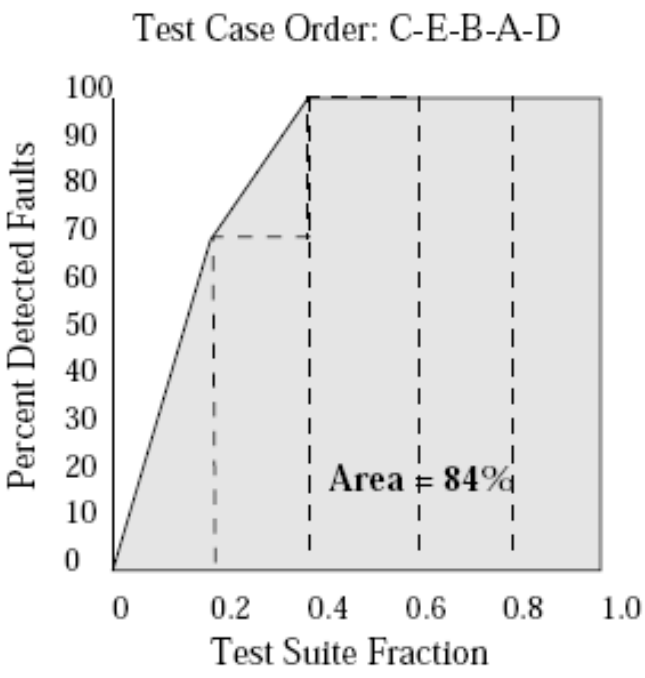
A. Test suite and faults exposed



B. APFD for prioritized suite T1

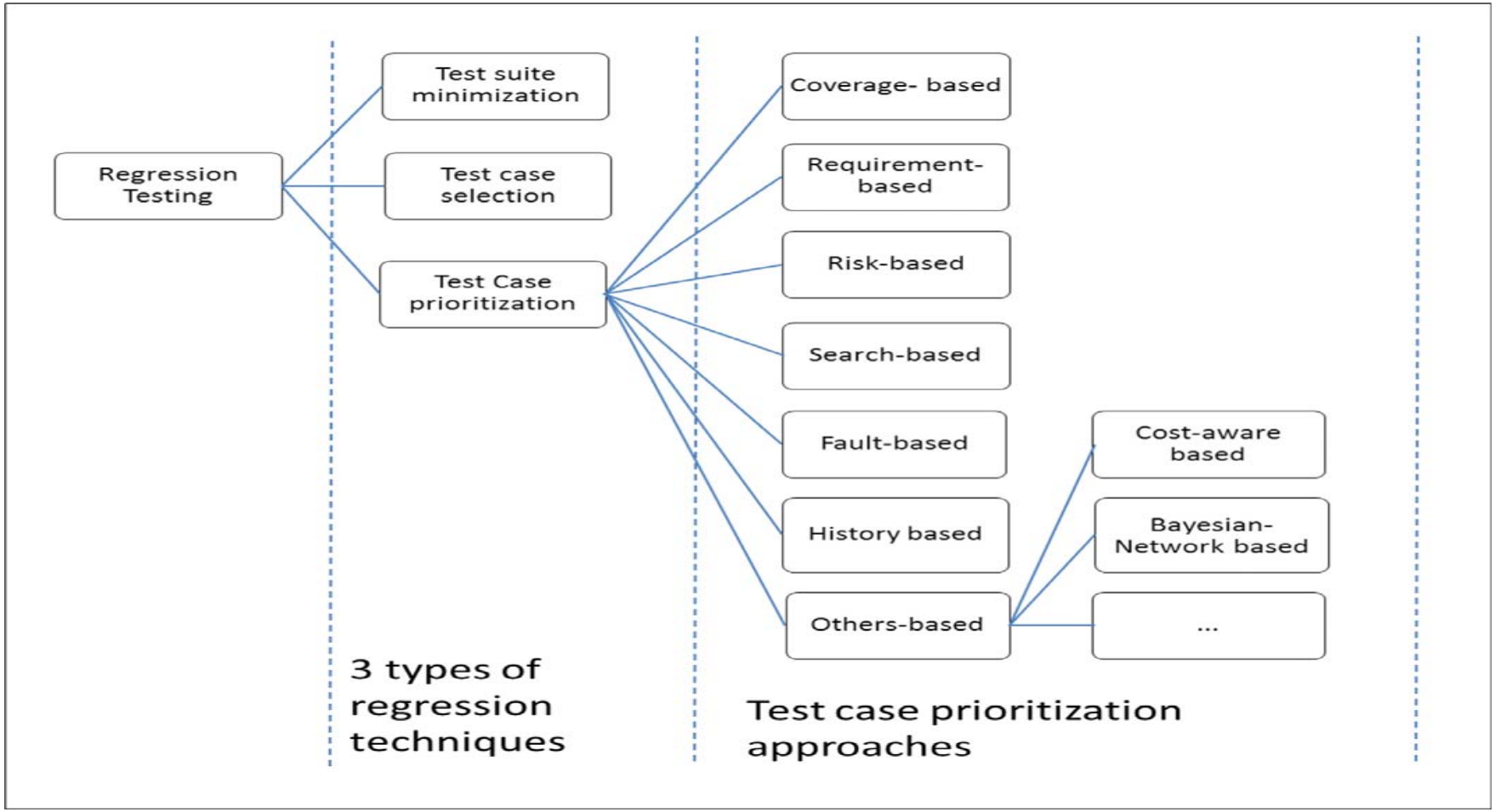


C. APFD for prioritized suite T2



D. APFD for prioritized suite T3

Test case Prioritization



Khatibsyarbini, M.; Isa, M.A.; Jawawi, D.N.A.; Tumeng, R. Test case prioritization approaches in regression testing: a systematic literature. *Information and Software Technology*, v 93, p 74-93, Jan. 2018

When to do Regression Testing

- Regular Execution (daily, weekly, monthly)
- Perform in accordance with the regulations (after each change, after an important change, before the final version is released)
- The regression test ends with:
  - All errors have been detected
  - Run all designed test cases
  - Exhaustion of resources
  - Project Expiration
  - …

Regression Testing Tools

- Important tools used for both functional and regression testing:
  - **Selenium:** This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.
  - **Quick Test Professional (QTP):** HP Quick Test Professional is automated software designed to automate functional and regression test cases. It is a Data driven, Keyword based tool.
  - **Rational Functional Tester (RFT):** IBM's rational functional tester is a Java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

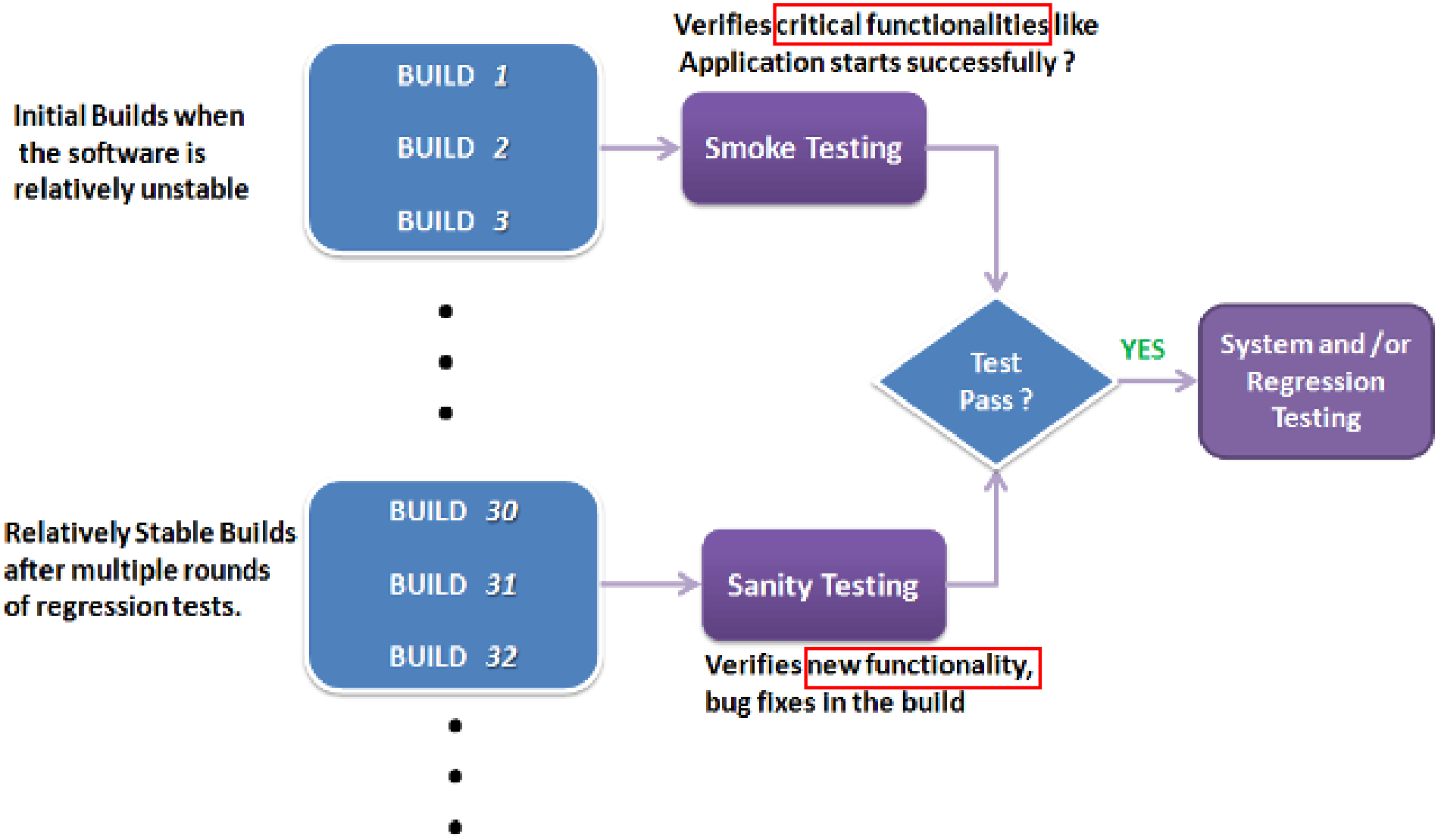
Comparison of concepts

- Re-Testing and Regression Testing:
  - Retesting means testing the functionality or bug again to ensure the code is fixed. If it is not fixed, Defect needs to be re-opened. If fixed, Defect is closed.
  - Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

Comparison of concepts

- 冒烟测试（smoke test）
- 可用性测试（sanity test）
- BVT （Build Verification Test）

- Both sanity tests and smoke tests are ways to avoid wasting time and effort by quickly determining whether an application is too flawed to merit any rigorous testing.


Verifies critical functionalities like Application starts successfully ?
Initial Builds when the software is relatively unstable
BUILD 1
BUILD 2
BUILD 3
Smoke Testing
Test Pass ?
YES
System and /or Regression Testing
Relatively Stable Builds after multiple rounds of regression tests.
BUILD 30
BUILD 31
BUILD 32
Sanity Testing
Verifies new functionality, bug fixes in the build

What is Smoke Testing?

- To ascertain that the critical functionalities of the program is working fine.
- It is executed "before" any detailed functional or regression tests are executed on the software build.
- The purpose is to reject a badly broken application, so that the QA team does not waste time installing and testing the software application.

What is Smoke Testing?

- A typical smoke test would be
  - Verify that the application launches successfully
  - Check that the GUI is responsive
  - …

  - “Hello, world ”

What is Sanity Testing?

- After receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.
- If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

What is Sanity Testing?

- The objective is "not" to verify thoroughly the new functionality, but to determine that the developer has applied some rationality (sanity) while producing the software.
- For instance, if your scientific calculator gives the result of 2 + 2 =5. Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

Smoke Testing Vs Sanity Testing - Key Differences

Smoke Testing	Sanity Testing
Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine	Sanity Testing is done to check the new functionality / bugs have been fixed
The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing	The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing
This testing is performed by the developers or testers	Sanity testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Tester acceptance testing

shallow and wide

narrow and deep

Points to note

- One of the best industry practice is to conduct a Daily build and smoke test in software projects.
- First execute Smoke tests and then go ahead with Sanity Testing.
- In industry, test cases for Sanity Testing are commonly combined with that for smoke tests, to speed up test execution. Hence, it's a common that the terms are often confused and used interchangeably.

Summary

- 各用一句话分别描述UISA的基本定义
- 结合5W+1H对比UISA
- 理解回归测试，分别论述TC-Minimization, TC-Selection, TC-prioritization的特点和做法
- 对比Smoke Testing 和 Sanity Testing