

SOFTENG | Domande di Teoria

- **What is the basic principle of the Visual approach to GUI testing? What is different with respect to layout-based scripted GUI testing?**
 - Visual approach: graphic components (buttons, menus etc) are recognized via image recognition
 - Layout approach: graphic components are identified retrieving their ID (used by the graphic library) or by specific unique properties.
 - **Provide an example of a static analysis rule from MISRA-C**
`i=i+1 → i++`
 - **In which cases an Oracle can be automatic?**

An automatic oracle is possible ONLY if a previous and reliable version of the software application is available (regression), or if the function can be expressed mathematically.
 - **Describe briefly 'mutation testing' and its purpose (x2)**

"Mutation test" is a technique used to evaluate how good a test suite is. Errors are inserted in the program under test (mutations). The more mutations the test suite finds, the better the test suite is.
 - **In the context of configuration management, what is the derivation history of a configuration item?**

It consists of the history of versions and changes to them
 - **What is the core content of the ISO 12207 standard?**

ISO 12207 is an ISO standard used to manage the life cycle of a software based on 2 main principles:
Modularity → Defining processes with the lower coupling and the higher cohesion possible.
Responsibility → There is a responsible for every process. There are three type of processes: primary, supporting and organizational.
 - **Describe the 'repository' architectural style, its pros and cons (x4)**

A repository architecture consists of a central data structure and a collection of independent components which operate on the central data structure: there is no direct interaction between applications and the repository file is the only point of interaction (centralized model). Is expensive but with the advantage of an high level of security, and efficient backup management. Ex. Compile linker, Unix cell
-
- This answer is NOT about describing repositories for configuration management (SVN, Git...)
- Alternate: Describe the 'pipeline' architectural style, and when it can be used
- **From the point of view of a user of a software application, what is more relevant, a defect or a failure in the software application? Explain why.**

A fault is a defect in the system: it may cause failure or not. On the other side, a failure is a malfunction perceived by the user (so it's worse)
 - **Describe briefly the problem of interactions and trade offs in non-functional requirements**

Some NF requirements may be in conflict (like performance and security and/or performance)

and accuracy), so it is unfeasible to achieve both, and trade offs must be accepted

- **Given a software project with 4 team members, what are the risks if no configuration management is used?**

Concurrent access and inconsistent modifications of CIs, unavailability of past versions of CIs.

⚠ My answer: without a proper configuration manager, the lack of a repository history complicate the revert to past changes. Concurrent programming is harder without branch handling and all the modifications are uncredited.

- **Describe the scrum process, its pros and cons. (x6)**

The scrum process is an agile methods made of sprints iterations of fixed duration (max 4 weeks) producing a working application in increments. Sprint reviews meetings allow the ranking of requirements by end user / customer (requirements backlog): the ranking may change after an iteration. Every day has a stand up meeting (15') for coordination.

→ PRO focus on customer and his feedback, daily feedback in a team.

→ CONS no document, only code.

Iteration fixed in time: pros to not loose time, cons for uncompleted work

- **Describe the abstract factory design pattern, and when it can be used.**

- **In the context of verification and validation, describe Weinberg's law (x3)**

The creator of a program is unsuitable to test it – for emotional attachment to its code he does not want to find many errors and tends to overlook defects in it.

⚠ Howden Theorem → It's impossible to find an algorithm able to generate a finite ideal test (selected by a valid and reliable criterion)
ParetoZipf Law → 80% of defects come from 20% of modules: it's better to concentrate on the faulty ones

- **In the context of change control, describe the lock-modify-unlock technique, its pros and cons (x3)**

→ Change control gives a discipline in who can modified what. Developer don't work directly in the repository, but in his workspace and the two must be synchronized at important changes.

→ In lock modify unlock a developer try to get a lock on a CI using checkout, works on it and after releases the lock through a check in. Two devs cannot work at the same time on the same CI and starvation if a developer forgets to unlock.

- **In the context of configuration management, explain what is a baseline and when is it used. (x3)**

A baseline is a configuration (== a set of configuration items) that is stable (ex compiles, links, passes all regression tests). Not all configurations are baselines. Used to delivery an application internally / externally, while development continues on next version and to eventually rollback to i

- **Describe the waterfall process, its pros and cons. (x2)**

→ Activities (requirement, design, implementation, unit test, integration test, system test) are done in sequence (activity i+1 starts only after activity i is completed). Document oriented.

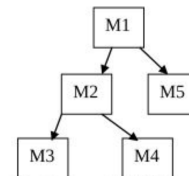
→ Pros: easy structure of activities; agreement on design allows to allocate tasks to many, distributed workers/companies.

→ Cons: delivery to customer and validation of requirements and system happen very late. Changes require to restart the process, slowness and lack of flexibility.

- **Describe the singleton design pattern, and when it can be used. (x2)**

The singleton is a creational pattern which ensures that only one instance of the class is created. For example, in an operating system the load balancer must be unique, and could be implemented by a singleton.

- **Given this dependency graph propose and justify an integration strategy**



BOTTOM UP:

→ Test M3, Test M4

→ Test M2+M3+M4

→ Test M5

→ Test all

TOP DOWN:

→ Test M1 + stubM2 + stubM5

→ Test M1 + M2 + M5 + stubM3 + stubM4

→ Test all

(Bottom up requires less stubs / drivers)



STUB: Unit developed to substitute another unit (fake unit)

- **Describe how versioning is implemented in Subversion**

All configuration items in a configuration inherit version number from the configuration: configuration number increments by 1 (1,2,3,4...)

- **In the context of configuration management, explain what versioning is and why it is useful.**

Versioning = keep copy of each instance of a configuration item (CI). This allows to keep the history of all modifications to a CI, and allows to roll back to any past instance of a CI.

- **Function A calls function B, that calls function C. You want to apply bottom up integration. How do you proceed?**

A(B(C()))

→ Test C

→ Test B+C

→ Test A+B+C

- **Considering GIT, what are the three project sections that it defines, and how are they used? (x2)**

Git directory (which stores metadata and database of versions)

Working directory (composed of a single checkout of one version of the project under modification)

Staging area (keep information on what will go on the next commit)

- **What is the typical lifecycle for a change? (draw states and transitions)**

- Receive change request
- Filter (search for similar CR, discard unfeasible CR)
- Access (Evaluate impact of CR, rank CR)
- Assign CR to a (team of) maintainer
- Implement CR (Design, Code, Unit, Integration...)
- Merge CR with next release of application



Alternatives:

What are the typical states of a Change Request in a maintenance process?

Describe a maintenance software process

- **What measures can be used to evaluate the quality of software? (x2)**

Fault density → Number of defects / size

MTBF (mean time between failures) → Number of defects found over a period of time

User satisfaction (questionnaire)



About fault densities

A good benchmark has less than 1 fault for a KLOC (1000 LOCs)

A bad benchmark has more than 10 fault for KLOC

- **Describe shortly the Adapter Design pattern (x2)**

The adapter design pattern is a structural pattern used to provide an abstract solution to the interoperability problem between different interfaces. For example, speaking of barcodes, I could have different APIs provided by different manufacturers: every barcode has different function and different names so I can use adaptors to avoid changing everything.



I also have various other structural patterns:

Listener → An interface forcing a certain action after an event (like a keyboard press and/or release)

Observer/Observable → Seen in the MVC pattern: we have a single models and many views. The problem is that when there is a change on the observable every view must be updated. The idea is to define a protocol to handle this problem (all the observer are notified).

- **Describe shortly the Test Driven Development technique**

- Write one test case that fails (using requirements)
- Write corresponding code until test case passes
- Repeat until all requirements are satisfied and all test cases pass

- **In project management, what are the units of measure for duration and effort? And what is the difference between these measures? (x2)**

Calendar time (or duration) → It can be relative (from project start like "two days") or absolute ("from 12-7 to 14-7").

Effort → Equal to *person * hours*. As by the definition, duration measures the time needed to

complete a project and the effort the amount of work needed. Given the effort of a project, calendar time depends on how many people work on it (staffing profile).

- **Describe shortly the Facade Design pattern (x2)**

The facade is a wrapper having multiple public methods passing through him: it's a useful solution in order to enforce encapsulation and to prevent the illicit access of public meths by clients from outside.

- **Explain and provide an example of a configuration item (x4)**

It is the basic unit of the configuration management system and correspond to a document or piece of code, eg a class, under the configuration control. Each has a name and a version.

- **Describe the copy modify merge strategy for controlling changes. List advantages and disadvantages (x2)**

In the context of configuration management, what is the "copy modify merge" approach and what are its pros and cons?

Subversion, CVS, and other version control systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client reads the repository and creates a personal working copy of the file or project. Users then work in parallel, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a human being is responsible for making it happen correctly.

- **Describe shortly the Pair Programming technique from eXtremeProgramming (x2)**

In eXtreme programming code is written by a pair of people of same experience looking at the same machine. One writes code and the other thinks about potential improvements, test cases and issues. They swap periodically. There is an overall improvement in quality (due to continuous inspections) but in order to not waste development time, pairs need to function.

- **Describe shortly the MVC Design pattern (x4)**

MVC is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements called model, view and controller. It provides a separation of responsibilities at the cost of more complexity.

⚠ Model → The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View → Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

Controller → Accepts input and converts it to commands for the model or view.

⚠ When I have to show data to user and manage changes to data I also have:

Class Representation → The traditional one, easy at the cost of simplicity

Microservices → I split my application in a series of microservices in need to change information via HTTP calls. It's not pure HTTP but HTTP restful with the advantage of a coupling reduction between two microservices. The downside is an increasing complexity and the worsening of time performance.

- **Someone asks you to develop a simple web site, a work that requires around 3 person months. Which milestones would you define? Which deliverables?**

Milestones:

Month 1 → Only GUI without logic

Month 2 → GUI + Logic interaction

Month 3 → Delivery

Documents:

GUI Mock up and Requirements (M1, Design (M1), Code (M3))

- **Describe what is an oracle in testing, the key problems related to it and how can it be implemented (x2)**

The expected output in testing is given by an oracle, an ideal concept that always gives the right prediction (assumption). It can be automatic (generated from software developed by other parties or previous versions of the program) or generated from formal requirement specifications or by an expert in the field. A human oracle is subject to errors because it is based on program specifications (which may be wrong).

- **List the types of defects that can be found in a requirement document (x3)**

→ Omission and/or incompleteness
 → Incorrect Fact
 → Inconsistency and or contradiction
 → Ambiguity
 → Extraneous Information
 → Over-specification (design)
 → Redundancy

- **What are the main differences between an iterative and a waterfall process?**

The waterfall perform a single, long iteration while the iterative one delivers smaller quantities multiple times

- **140 person hours are equivalent to how many person months? Explain your answer**

The conversion rate depends on national and company rules. If one person month == 10 person hours then result is of course 1. If one person month == 160 person hours then result is 140/160. And so on.

- **Describe the key steps in an inspection process of a requirement document (x3)**

Moderator choose participants and schedule event.

He shows the goal of inspection and techniques to be used, for example perspective-based, scenario-based, defect-based reading

Each participant read the document.

Then they meet up to read the document together and discuss defects found.

Defects have to be solved and then there is a follow up to verify fixes.

⚠ Inspection is a V&V technique where people read the code and search for defects. It's made by a group with the objective of finding defects with no correction. The people who read the code should not be the ones reviewing it as the author is emotionally involved to its program so it's brought to hide eventual defects.

- **What are Function Points used for? How are they defined?**
To estimate the size of code.
- **What kind of problem 'traceability' identifies?**
 - Linking requirements to design components (classes)
 - Linking functions to code components
 - Linking functions to test cases
- **A defect on a SW product signaled by a customer has been allocated to a developer, who has found the corresponding error. What happens next?**
 - Fix the defect
 - Rebuild code
 - Make a test
 - Make a regression test
 - Decide release to attach the patch to
 - Release
- **The size of a project is estimated to be 600 FP. The project is developed in Java, past figures from the company tell that one FP is worth 20 to 30 LOC, and that productivity is 10-15 LOC per person days. How many person days would be required for the project? (show steps followed to compute your answer).**
 - Total LOC → $600 \text{ FP} * 20 \text{ or } 30 = 12000 \text{ to } 18000 \text{ LOC}$
 - Min effort → $12.000 \text{ LOC} / 15 \text{ LOC per person days} = 800 \text{ PD}$
 - Max effort → $18.000 \text{ LOC} / 10 \text{ LOC per person days} = 1800 \text{ PD}$



Function Point (FP) is an element of software development which helps to approximate the cost of development early in the process.

- **A software product is developed custom for a bank. Development takes 18 months, operation and maintenance 10 years. Where do you expect to have the larger part of costs?**
Maintenance, because 10 years is way longer than 18 months, evolutive, corrective and enhancement maintenance will likely require more effort than initial development, especially considering the fact that in maintenance new cycles of requirement, design and implementation happen every 6 months circa.

example: i have 3 people involved in the project

→ Development = $18 * 3$ person months

→ Maintenance = $10 * 12 * (0.5 \text{ [I assume half a person working in maintenance]})$ person months



Similar one found an exam: "an application is developed for an organization in 6 months by 3 people. The it is used by the organisation during 12 years. Argument whether maintenance cost would be higher/lower than any development costs.

- **A project is estimated to require 60 person months for development. What could be the calendar duration for development?**
Depends on number of people working
 - 3 people = 20 calendar months
 - 4 people = 15 calendar months



There is a limit on the number of people! 20 people is unrealistic, so 3 months is unfeasible

- **Describe shortly the Function Point method**
- **Describe shortly the 'incremental' software process**
Like the waterfall but the integration test is split and done incrementally, and each loop produce a part of the system delivered to customers for feedback.
- **A project has the following estimated Gantt. Planned value of the project is 100 units, 50 on activity1, 50 on activity2. One month after start the project has consumed 60 units, activity1 is finished, activity2 is not. What is the Earned Value of the project?**
EV = 50. Activity2 does not count because is NOT finished
- **"Cost of the project should be measured in Euros" what kind of requirement is this?**
Domain requirement (a kind of NF requirement)
- **Describe shortly the Delphi estimation method.**
- **What is the recommend length of an iteration in an agile project?**
4-5 weeks at most
- **Describe shortly the Strategy Design pattern**
- **What are the possible type of defects in a code module?**
 - Syntax, type and semantic errors;
 - Use of undefined or non-initialized variables
 - Not used variable (data-flow)
 - Bad smells (too long classes or list of attributes)
 - Symbolic execution (translation of mathematical formulas)
- **In the context of project management, give the definition of 'deliverable'**
A Deliverable is a product of the process. It may be either final or intermediate, can be internal (if it must be used only in the producer company) or external (if it is done for the customer or must be validated by it). Some deliverables (like the requirements management or the design document) may have contractual values between customer and producer.
- **Describe briefly the 'layered' architectural style**
- **In the context of project management, give the definition of 'milestone' (x2)**
A Milestone is a key event or condition in the project, that can also serve as a synchronization point for its transitions between one phase to another. A milestone has effects on the subsequent activities: whether it is reached or not, the activities to be performed later may have to change.
- **In the context of project management, give the definition of 'slack time'**
Admissible delay to complete an activity without changing the end time of project
- **Provide an example of two conflicting non-functional properties of a software architecture**
 - Very high precision of computation (ex square root precision 10^{-10}) conflicts with Performance.
 - The higher the precision, the slower the response time.
- **A project is estimated to require 40 person months. Give an estimated range of the required calendar time, and explain how you compute it.**

→ A reasonable team for such a project could be 2 to 5 people.
 → Assuming the team has the same number of people from start to end this gives a calendar range of 20 to 8 calendar months (computed as person * months / persons). A team of 40 people is unreasonable (too much coordination and communication overhead) so 1 month duration would be unfeasible. Similarly for 20, 30 people.

- **Give the definition of effort in the context of project management, and its unit of measure (x2)**

The Effort is the time taken by the staff involved in a project to complete a certain task. It is measured in person hours (STD IEEE 1045), and also person day, person month, person year depending on national and corporation parameters.

- **What is a Configuration? (x2)**

A configuration is a set of Configuration Items (CI), each in a specific version, and it can be seen as a snapshot of the software at certain time. Some CIs may appear in different configurations, and also configuration has a version number.

- **Give the definition of calendar time in the context of project management, and its unit of measure**

Calendar time is one of the available tools for managing a process. It's measured in calendar days, weeks, months either using an absolute value (from 13 July 2021 to 23 September 2021) or relative (from project start)

- **Describe the CCB (Configuration Control Board) approach for change management**

- **Describe what are mixed revisions in Subversion?**

→ Mixed revisions are revisions of a working copy whose files have different revision numbers.
 → Mixed revisions are possible only in working copies and not in the central repository.
 Example: Suppose you have a working copy entirely at revision 10. You edit the file foo.html and then perform an svn commit, which creates revision 15 in the repository. Therefore the Subversion marks the file foo.html as being at revision 15. The rest of the working copy remains at revision 10. This is a mixed revision.

- **Describe shortly the perspective based inspection technique (x2)**

Inspection where readers use different points of view (ex end user, designer, tester) to read the requirement document.

- **Function A calls function B, that calls function C. You want to apply top down integration. How do you proceed?**

→ Test A using a stub for B
 → Test A + B using a stub for C
 → Test A+B+C

- **Class A has been developed and you should test it. What is better, do black box testing only, white box only, or both? Why?**

Both. WB and BB have different criteria of writing test cases and, together, higher probability of finding them.

- **The cost of fixing a defect in a requirements document is higher if the project is in the requirement phase, or if the project is in the coding phase? Why?**

Fixing a defect costs less in the requirements phase, essentially because no other artifacts (design document, code, test cases) exist yet at that stage. So fixing a requirements defect in

the requirement phase means only changing one document, instead of many (if fixed in subsequent phases). Think to a house: it costs less to add a window in the project plan (= requirements analysis, you just correct tables or diagrams) rather than adding it when the house is in construction (= writing code, you must correct code) or it is already built and people is living there (=software is in use, you must write patches).

- **What are the most important functions of a configuration management tool?**

- Identify and manage part of software
- Handle repository history
- Handle branches, configurations and versions of software and eventually revert project back to a certain version (rollback)
- Handle accesses and changes to parts (who changed, what changed and when it changed something)

- **A defect can be injected, discovered, removed. Of course not injecting defects at all is the best option. Is this feasible? Argue briefly your answer.**

Answer: Not injecting defects is not feasible: software is not defect-free for definition, since it's the result of a human activity. Probability of inserting a defect writing or changing code is different from zero (Adams's Law)

- **What is an inspection? How does it work?**

Inspection is a V&V technique, used to verify the requirements/design document, test cases or the developed code. Performing an inspection means reading a document (or code) alone, and then repeating the reading with a group of people (typically more than three) that include the author of the document. The goal of an inspection is to find defects.



Another V&V technique worth mentioning is prototyping: a small real software prototype with fewer use cases than the final version. It's more loose on NFRs and not perfect but useful to show the main functionalities to the end user in order to better understand what he/she wants

- **What is a deadline? Give an example of a deadline in a construction project.**

A deadline is a calendar date when a task/deliverable has to be completed/issued

Ex: Roof ready by MAY 31 2009

- **According to the Cocomo model, how are related duration and effort? (x2)**

$TDEV = 3 PM (0.33 + 0.2 * (B Subversion, CVS, and other version control systems use a copy-modify-merge model as an alternative to locking. In this model, each user's client reads the repository and creates a personal working copy of the file or project. Users then work in parallel, modifying their private copies. Finally, the private copies are merged together into a new, final version. The version control system often assists with the merging, but ultimately a human being is responsible for making it happen correctly. - 1.01))$ with $B=1$ in the simplest model. Writing the formula was not necessary, key point was saying that duration (TDEV) depends on effort (PM) with exponential formula, exponent <1 .

- **What is a build? What are the related problems?**

Answer: Process (including compilation, link, possibly testing and other activities) to produce an executable starting from source code modules and libraries. Often automated with a script. Problems: references and dependencies among modules, finding and using the right modules in term of type and version.

- **In the context of verification and validation, describe a static analysis technique Inspection or dataflow analysis or control flow analysis.**

Control Flow Analysis is based on the Control Flow Graph of the program, and checks the control flow constructs. Data Flow Analysis track variables during execution, to find out anomalies eg unused variables.

- **User Mario commits file A on a subversion server, subversion notifies that the file is at revision 300. User Mario and other users execute 3 more commits of other files. Then John commits again file A. What is the revision number of A?**

304 on the repository and on John's working copy. (However, Mario's working copy is at 301, unless Mario updates)

- **What are the key differences between testing and debugging? (x2)**

Testing tries to find failures, debug tries to discover and fix the correspondent fault(s).

- **Explains pros and cons of the Lines of code (LOC) measure. (x2)**

Not well estimable at first as it depends on language and programmer.

- **Software processes. Describe the key points of the evolutionary process**

Answer: The evolutionary model solves the problem of re-doing the whole waterfall that is present in the Incremental model: it is similar to the Incremental model, but requirements can change at each iteration. It is similar to prototyping, since at each iteration an incremental prototype is produced and the last prototype developed is the final product.

- **Describe the difference between bottom up and top down integration testing**

The bottom up incremental integration starts from the dependency graph.

→ First are tested the units that have no dependencies and then one unit at the time is integrated. If some defect is found, it should come from the last integrated unit or from the interaction, but not from the units already tested.

→ The top down incremental integration starts from the highest-level unit and it uses stubs, then integrates the units below one by one (that to be tested used stubs). This testing method fits well if we develop the software in a top down approach and can detect early architectural flaws. This saves the definition of drivers for the bottom classes but require writing stubs for them (the opposite of the bottom-up approach).

- **Define fault and failure, underlining the difference**

A fault is a defect in the system: it may cause failure or not. On the other side, a failure is a malfunction perceived by the user (so it's worse)

- **What is the difference between "correctness" and "reliability"?**

Correctness and reliability are NFRs: correctness consists in the capability to provide an intended functionality in all cases. If we can measure this with precision we can assume defects in reliability, which is measure defects by user per time period or probability of defect per time period.

- **What is the definition of software reliability?**

The best software is the one that has no faults, but a reliable one is a software with no failures, that is different.

- **Highlight the key differences between the incremental and iterative software processes (x2)**

→ In incremental software process, requirement and design document are immutable like in the waterfall but the integration test is split and done incrementally, and each loop produce a part of the system delivered to customers for feedback. The iterative model consists in many

iterations of the waterfall model, which of it produce a partial part of the project.

→ The difference from the Evolutionary is that at the first iteration not all the requirements are written, but also those are made incrementally, while Evolutionary have the requirements written at the first loop and only can modify it to adapt to user wills.


- **List three risks in software engineering projects (from Boehm top 10 list)**

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions
- Developing the wrong user interfaces
- Gold-plating
- Continuing stream of requirement changes
- Shortfalls in externally-performed tasks / furnished components
- Realtime performance shortfalls

- **Given a set of functional requirements, often many designs are possible. How to select one design option versus other ones?**

- **What is a Gantt chart, and when is it used**

A Gantt starts from the work breakdown structure and put activities on a calendar along with how many people are needed for doing it. It defines scheduling and temporal constraints. The Gantt is used in planning but also in replanning until post mortem.

 A Pert diagram contains the same info of the Gantt chart but with only the temporal constraints: an arrow means that the node can start when the previous activity is completed. The Pert is useful to find critical paths (What is the shortest path to complete the project? What are the critical activities to complete the project in the shortest time?)

- **What is the definition of exhaustive testing? Is it possible?**

Exhaustive testing is not possible (except in trivial cases). The goal of testing should be finding defects, not demonstrating that the software is defect free. The key point is HOW to select test cases in favor of reliability and validity to assure a good enough level of confidence.

- **What techniques can be used to validate the functional requirements of an application to be built?**

- Inspection of requirements
- Prototype building
- GUI Prototype building
- Writing acceptance test cases

- **In the context of configuration management, what is the purpose of "check in" and "check out" operations?**

They are used to:

- Enforce sequential changes to the Continuous Integration (CI) (in lock-modify-unlock mode)
- Support parallel changes without inconsistencies (in push-modify-merge mode)

- **There are many software processes. What factors characterize a software process?**

The main factors describing a certain software process are (see slide 154):

- Sequential parallel activities
- Iterations
- Emphasis on documents
- New development/maintenance
- Compliance
- Size



Alternative question: you have to start a software process. What are the factors to consider in selecting a suitable software process?

- **In a maintenance process, what are the possible types of a change?**
 - A defect to be fixed (corrective)
 - A modification to an existing function/characteristic (perfective)
 - A new function / characteristic (evolutionary maintenance, enhancement)

Revisione del documento "domande teoria SE" a cura di Antonio Macaluso