ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τεχνολογία Λογισμικού, 7ο εξάμηνο 2021-2022

Τεχνολογία Λογισμικού

Β.Βεσκούκης

“Testing is not responsible for the bugs inserted into software any more than the sun is responsible for creating dust in the air.”— Dorothy Graham

“**No amount of testing can prove a software right, a single test can prove a software wrong**.”— Amir Ghahrai

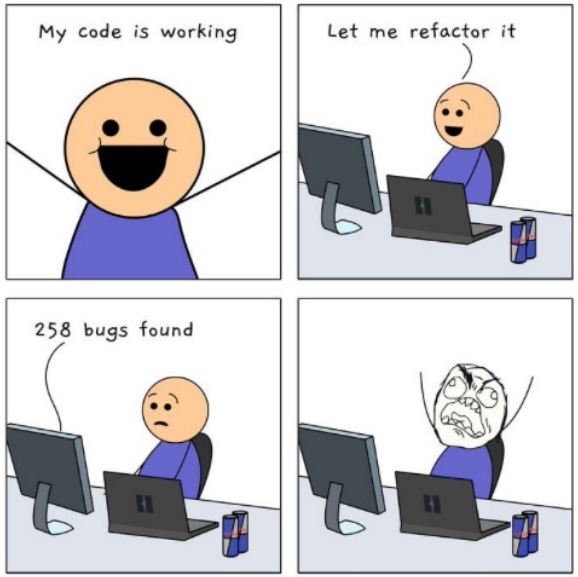
“I am pretty sure there is a difference between “this has not been proven” and “this is false.”— Ron Jeffries

“To those who say that “if you need testing at the end, you’re doing it wrong”, would you prefer a Boeing, or are you going Air Icarus?”— Michael Bolton

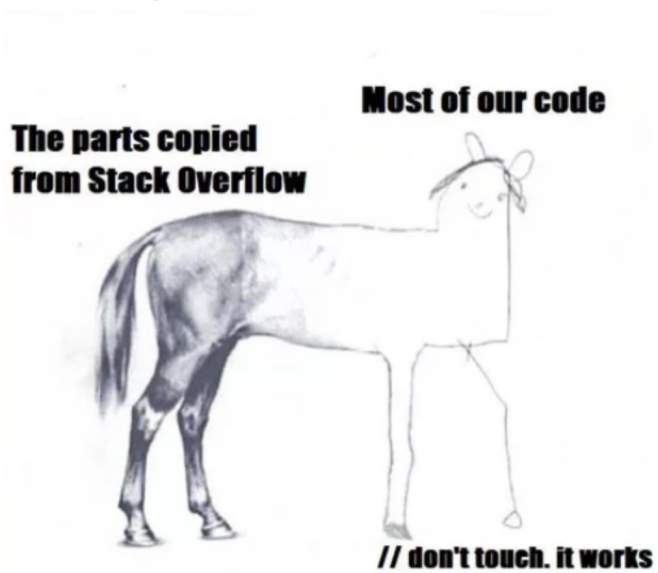
Software testing

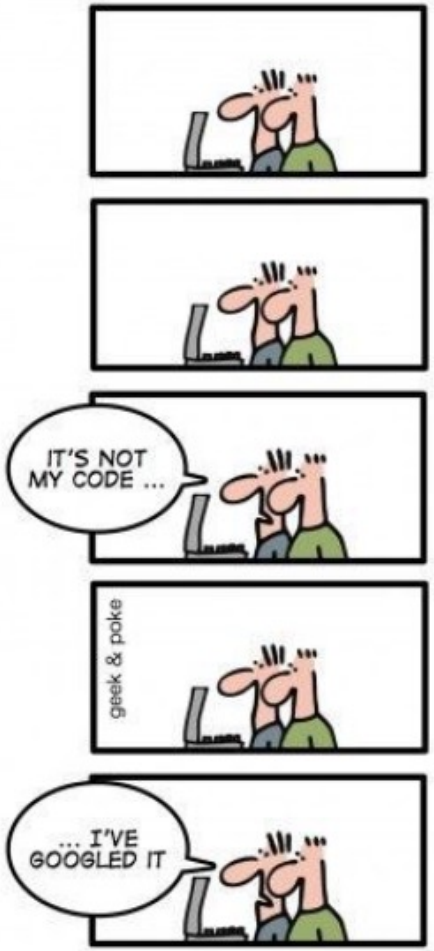
> There are two ways to write error-free programs; only the third one works.
>
> Alan J. Perlis

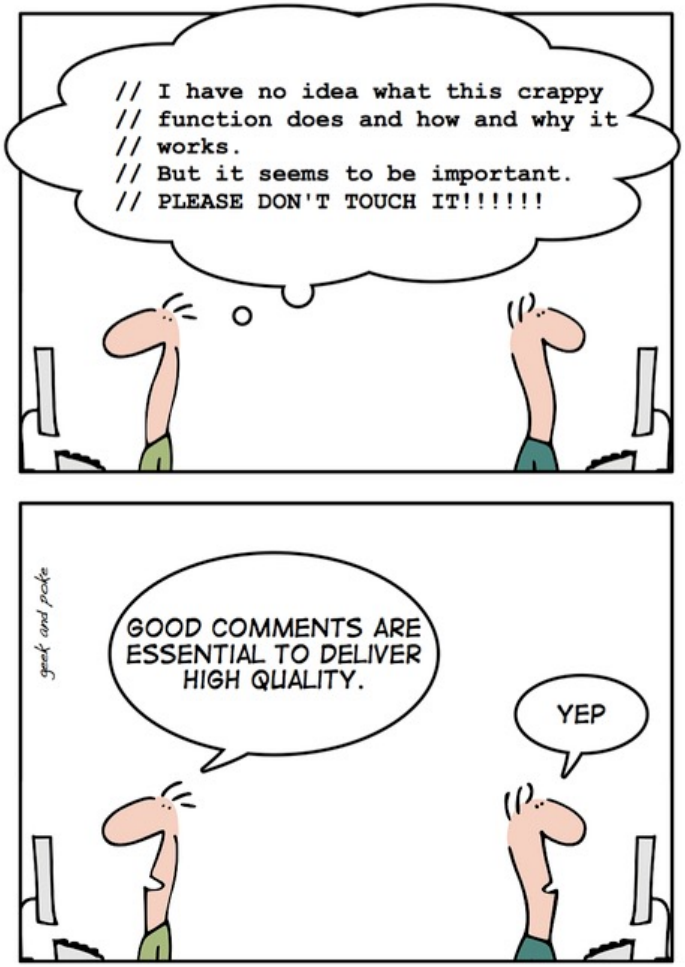
“Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the anonymous.”— James Bach



~~"If it works, don't touch it!"~~







Ελεγχος λογισμικού

Η τελευταία από τις εργασίες (κάθε κύκλου) ανάπτυξης λογισμικού

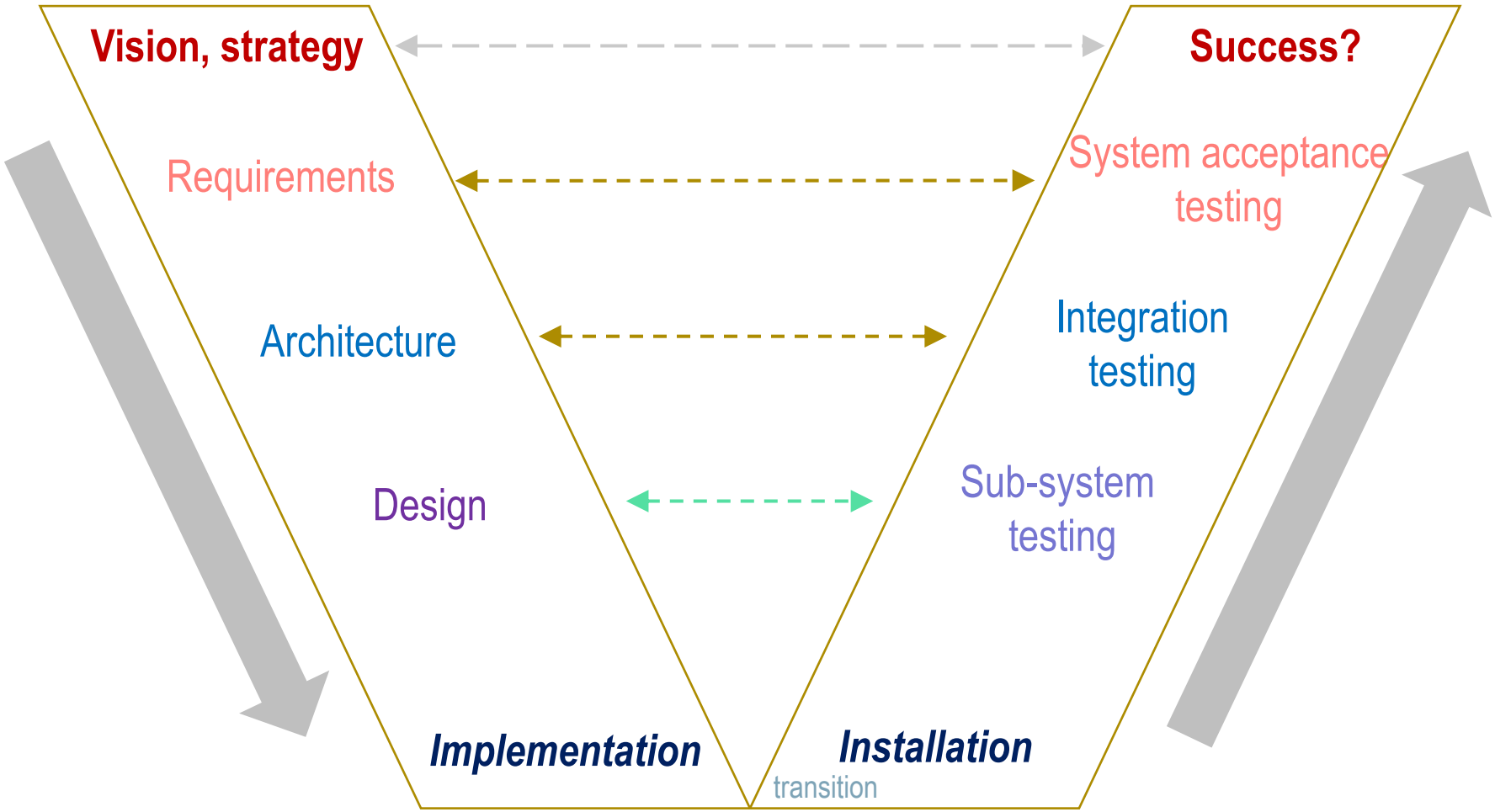
Εφαρμόζεται είτε στο σύνολο του λογισμικού, είτε σε τμήμα αυτού, ανάλογα με το Μοντέλο Κύκλου Ζωής που ακολουθείται.

Σκοπός του ελέγχου:

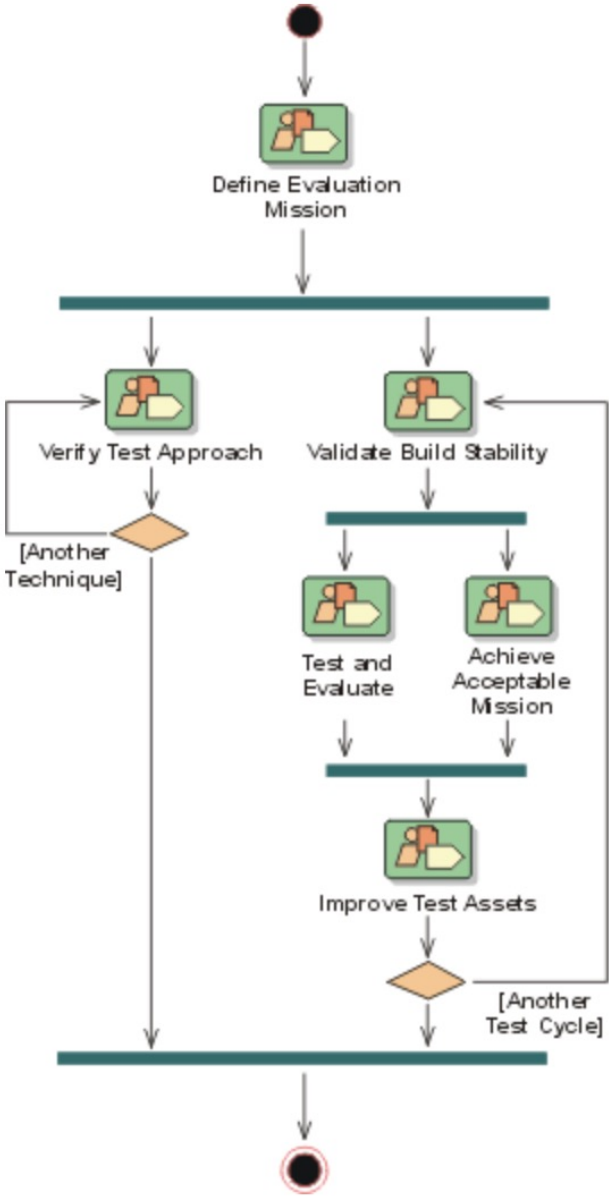
- Να επαληθεύσει ότι το λογισμικό ικανοποιεί τις απαιτήσεις που έχουν τεθεί κατά την φάση προδιαγραφών των απαιτήσεων
  ΚΑΝΕΙ ΑΥΤΟ ΠΟΥ ΠΡΕΠΕΙ (validation)*
- Να εντοπίσει προγραμματιστικά σφάλματα
  ΤΟ ΚΑΝΕΙ ΣΩΣΤΑ (verification)

Επίπεδα ελέγχου
Προϊόντος / Ικανοποίησης απαιτήσεων / Αρχιτεκτονικής / Σχεδίασης / Υλοποίησης / Μονάδων / Συνένωσης / Συστήματος / Αποδοχής

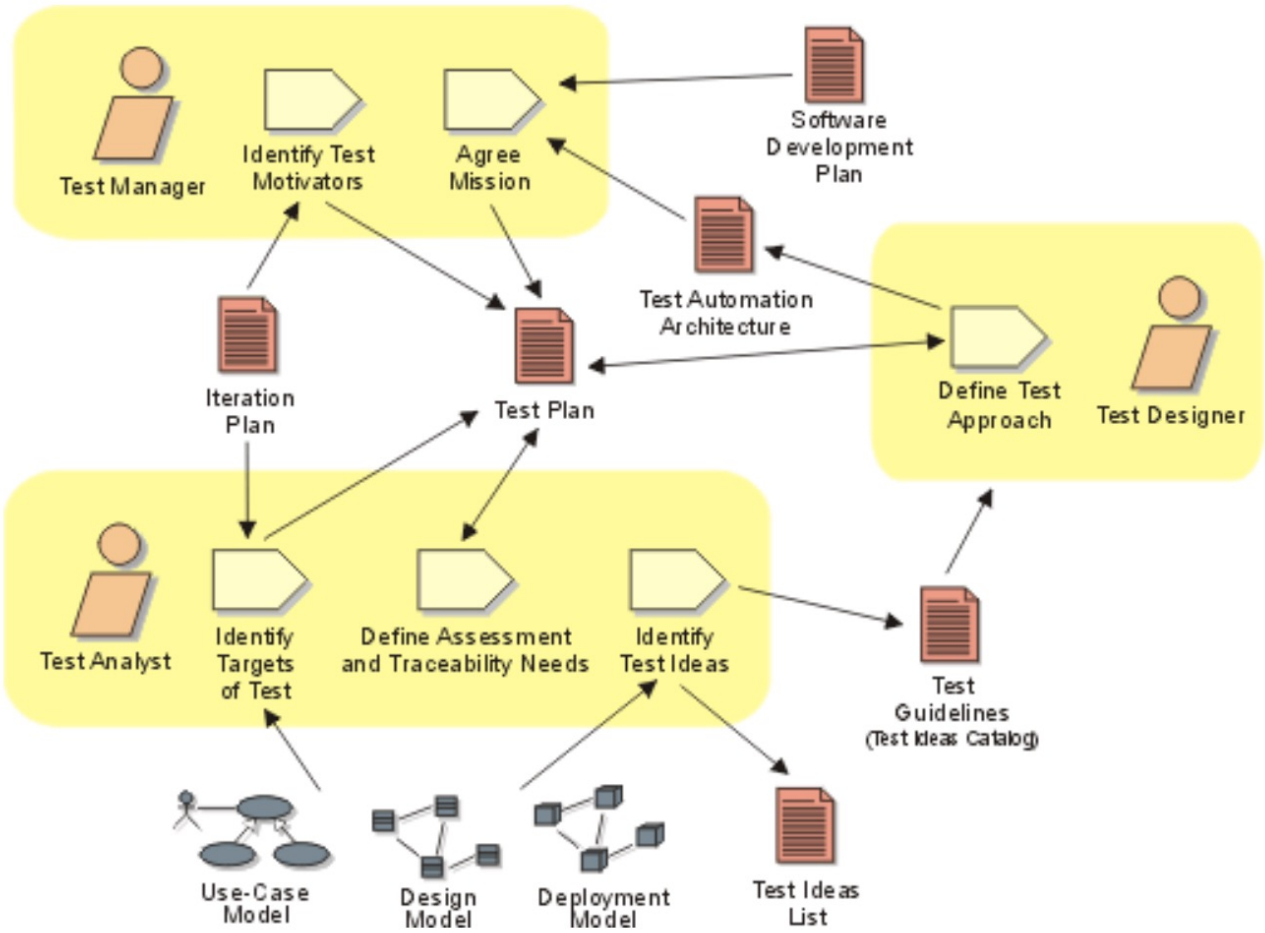
Testing, testing, testing...



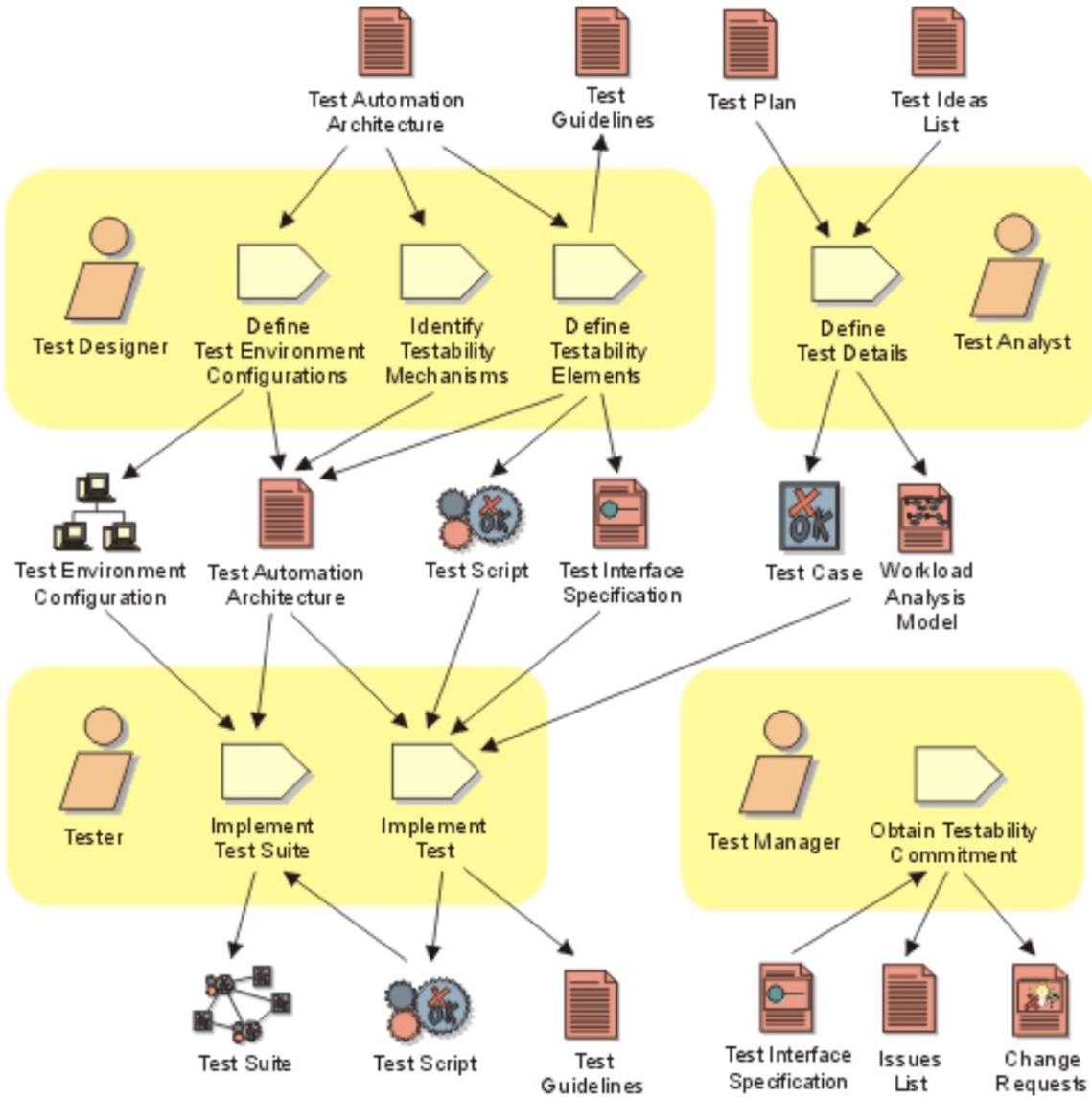
RUP's approach on testing



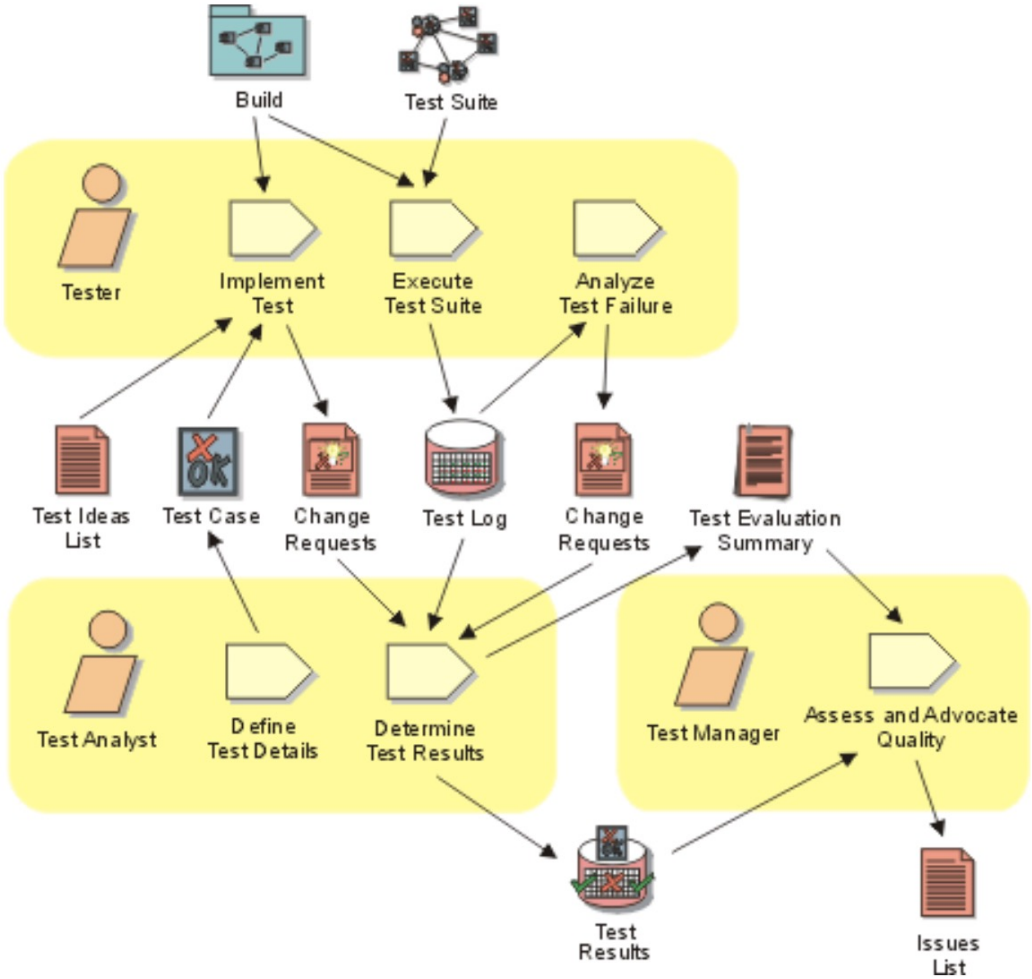
Workflow Detail: Define Evaluation Mission



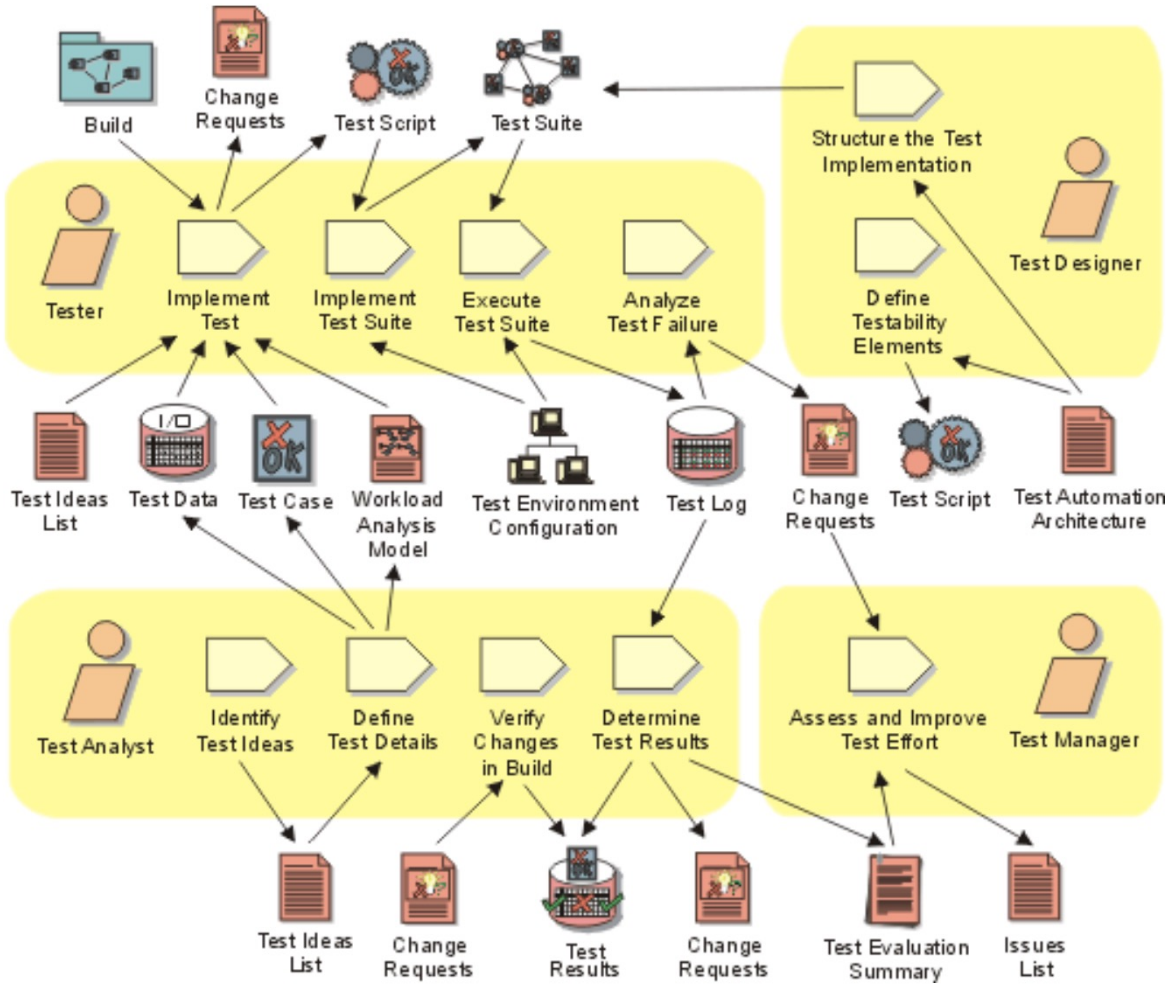
Workflow Detail: Verify Test Approach



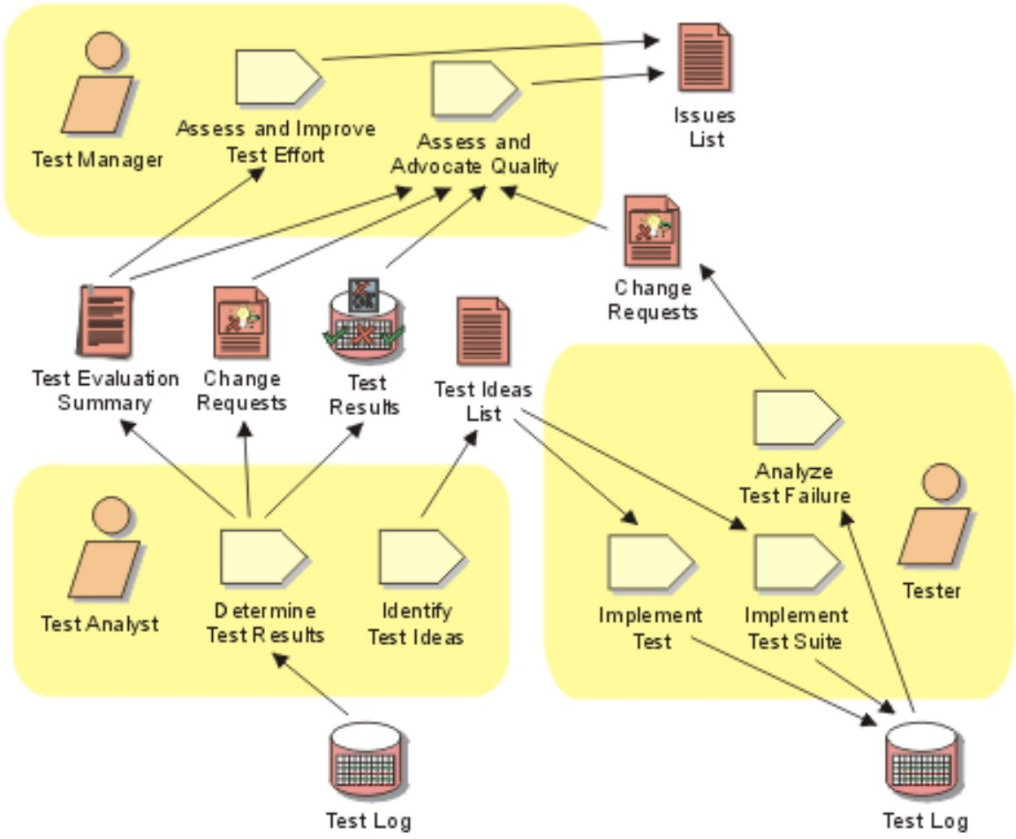
Workflow Detail: Validate Build Stability



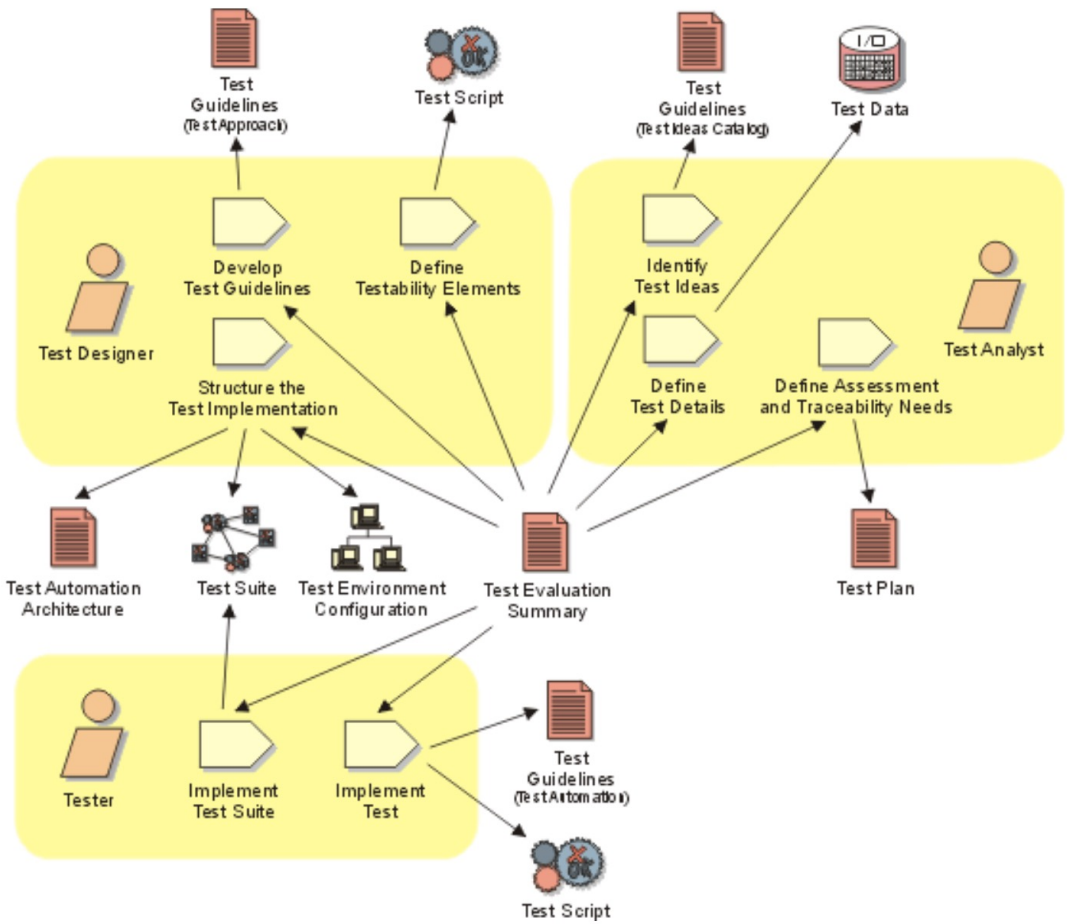
Workflow Detail: Test and Evaluate



Workflow Detail: Achieve Acceptable Mission



Workflow Detail: Improve Test Assets



Ελεγχος λογισμικού

Ικανοποίηση λειτουργικών απαιτήσεων

- Περιπτώσεις χρήσης, λειτουργίες, ορθότητα αποτελεσμάτων

Ικανοποίηση μη-λειτουργικών απαιτήσεων

- Επιδόσεις
- Ασφάλεια
- Χρηστικότητα
- Συμπεριφορά σε "οριακές" καταστάσεις (define "οριακές")
- Ανάνηψη από σφάλματα

Μέτρηση δεικτών ποιότητας (ποιων; πώς;)

Ελεγχος λογισμικού

Στατικός έλεγχος

- Επιθεωρήσεις (reviews)
- **Στατική ανάλυση**: δεν εκτελείται το ίδιο το λογισμικό, αλλά ο πηγαίος κώδικας είναι η είσοδος σε κάποιο εργαλείο που κάνει τον έλεγχο ως "στατική ανάλυση"
- Ενδεικτικά εργαλεία (JVM):
  Checkstyle, FindBugs, PMD, Jdepend, Jacoco, ErrorProne, SonarQube
- Πολλά σύγχρονα IDEs ενσωματώνουν στοιχεία στατικού ελέγχου (πχ **lint**)

Ελεγχος λογισμικού

Δυναμικός έλεγχος: Δοκιμές ελέγχου (test cases)

- Εκτελείται το λογισμικό, καταγράφεται η συμπεριφορά του και συγκρίνεται με κάποια "αναφορά" (επιθυμητή συμπεριφορά, αποτελέσματα κ.ά.)
- Ορθότητα αποτελεσμάτων για δεδομένα εισόδου με γνωστά αποτελέσματα
- Επιδόσεις (performance), καθυστερήσεις (latency)
- Συμπεριφορά "στα όρια" (συνδεσιμότητα, δικτυακές κλήσεις, κατάτμηση, κ.ά.)
- End-to-end testing: ελέγχουμε αν για γνωστή είσοδο, έχουμε την αναμενόμενη έξοδο
- Ενδεικτικά εργαλεία (JVM):
  JUnit (unit testing), TestNG (unit testing), Mockito (mockup testing), Selenium (web browser automation), Jenkins (Continuous integration) Travis (Continuous integration), Spock (specifications)

Εργαλεία ελέγχου

Γεννήτριες δοκιμαστικών δεδομένων

Εργαλεία σύγκρισης αρχείων

Εργαλεία δυναμικής ανάλυσης

Αυτόματοι εκτελεστές ελέγχου

- Πρόκληση εξοντωτικού αριθμού περιπτώσεων ελέγχου
- Παραγωγή μεγάλου όγκου δεδομένων εισόδου
- Ελεγχος καταπόνησης
- Προσομοίωση πραγματικών συνθηκών λειτουργίας

Ενσωματωμένα σε ολοκληρωμένα περιβάλλοντα ανάπτυξης

Προγραμματισμός του ελέγχου

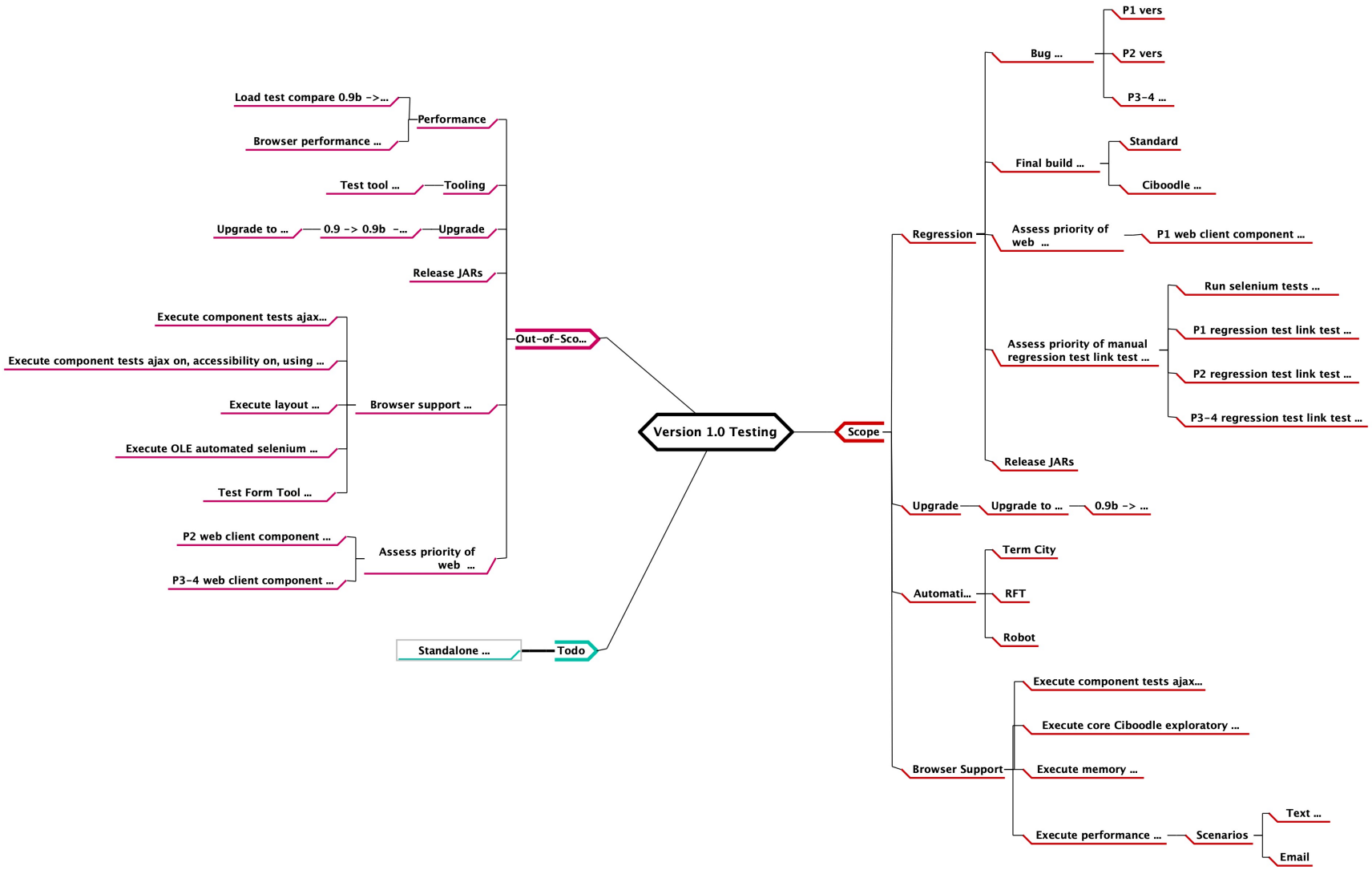
Ενδεικτικό πλάνο ελέγχου

1. Ταυτότητα του εγγράφου
2. Εισαγωγή
3. Οντότητες που θα ελεγχθούν
4. **Χαρακτηριστικά που θα ελεγχθούν**
5. **Χαρακτηριστικά που δεν θα ελεγχθούν**
6. Μέθοδος
7. Κριτήρια επιτυχίας / αποτυχίας ελέγχου
8. Κριτήρια ακύρωσης και προδιαγραφές επανάληψης ελέγχου
9. Παραδοτέα του ελέγχου (logs?)
10. Εργασίες που πρέπει να γίνουν
11. Αναγκαίοι πόροι περιβάλλοντος
12. Κατανομή ευθυνών για την εκτέλεση του ελέγχου
13. Ανάγκες στελέχωσης και εκπαίδευσης προσωπικού
14. Χρονοπρογραμματισμός ελέγχου
15. Κίνδυνοι και απρόοπτα
16. Εγκρίσεις

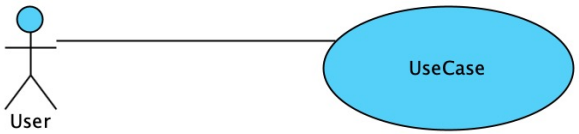
Ενα "πλάνο ελέγχου" μπορεί να αφορά ευρύτερα την πολιτική εκτέλεσης ελέγχων και όχι ένα συγκεκριμένο στοιχείο λογισμικού.

Βλ. και Software Quality Assurance

Test plan brainstorming



Test plan, use-case level



SoftEng2020vescoukis * - Visual Paradigm Professional

Dash | Project | ITSM | Agile | Diagram | View | Team | Tools | Modeling | Window | Help

UseCase > UseCase Details

UseCase

Info | Use Case Notes | Flow of Events | Details | Requirements | Diagrams | Test Plan | References

Rank: Unspecified

ID: UC02

Status: Identify | Next

Justification:

Primary Actors: User ...

Supporting Actors: ...

Task Pool: Link to...

Description:

Abstract | Leaf | Root

Diagram Navigator | Model Explorer | Property | Diagram Backlog | Logical View

SoftEng2020vescoukis * - Visual Paradigm Professional

Dash Project ITSM Agile Diagram View Team Tools Modeling Window Help

Diagram Navigator | Model Explorer | Property | Diagram Backlog | Logical View

UseCase > UseCase Details

UseCase

Info | Use Case Notes | Flow of Events | Details | Requirements | Diagrams | Test Plan | References

Testing Setup:

Testing Configurations:

Τεχνικές ελέγχου

Δοκιμή μονάδας:
Η εκτέλεση μιας μονάδας λογισμικού, με δεδομένα εισόδου για τα οποία τα αποτελέσματα είναι γνωστά και η σύγκριση των αποτελεσμάτων που αυτή παράγει, με τα γνωστά αποτελέσματα.

Περίπτωση ελέγχου: ένα σύνολο από

- Δοκιμαστικά δεδομένα Καλύπτουν όλες τις περιπτώσεις;
- Συνθήκες εκτέλεσης Όχι πάντα εύκολο να δημιουργήσουμε πραγματικές συνθήκες εκτέλεσης
- Αναμενόμενα αποτελέσματα Όχι πάντα εύκολο να γίνεται αυτόματα η σύγκριση

...το οποίο αποσκοπεί στην επαλήθευση ή τον έλεγχο προγράμματος μιας μονάδας λογισμικού

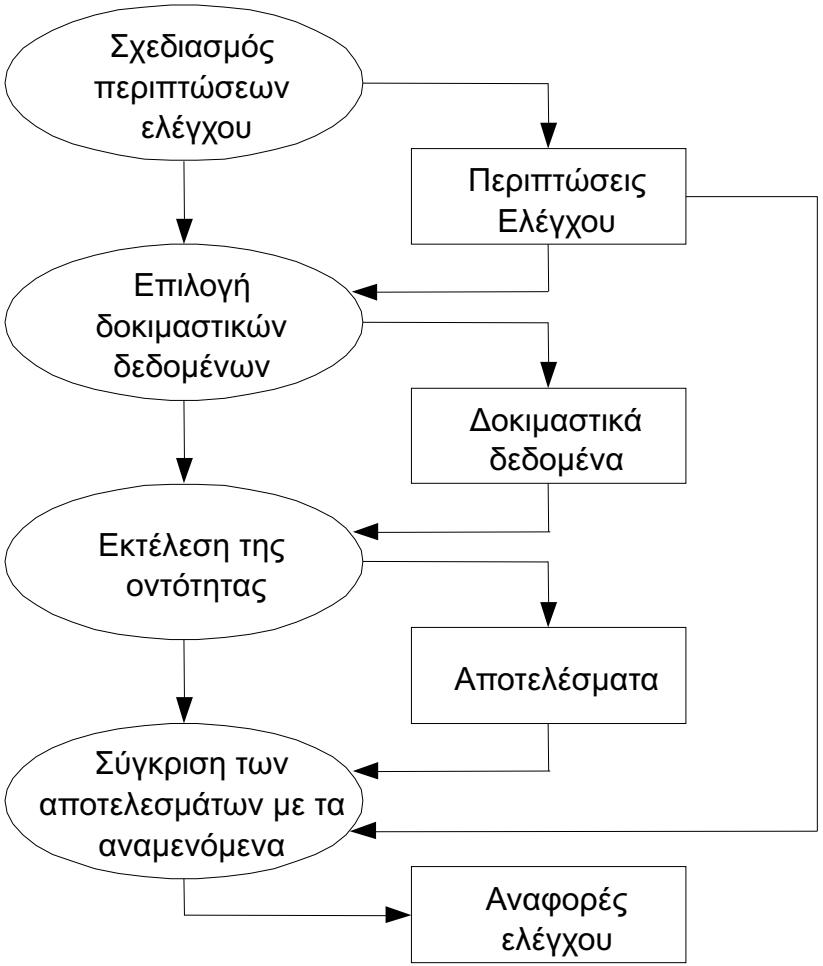
Συνήθως χρησιμοποιείται κάποιο εργαλείο

Τυπική ροή εργασιών ελέγχου

Δεν είναι δυνατόν να ελεγχθούν όλες οι περιπτώσεις

Όταν δεν αποκαλύπτονται σφάλματα, δεν σημαίνει ότι δεν υπάρχουν,
αλλά ότι ο έλεγχος που έγινε δεν τα αποκάλυψε

Οι απαιτήσεις του ελέγχου σε πόρους, ανθρώπους και χρόνο, μπορεί να είναι ιδιαίτερα μεγάλες



Στρατηγικές ελέγχου λογισμικού

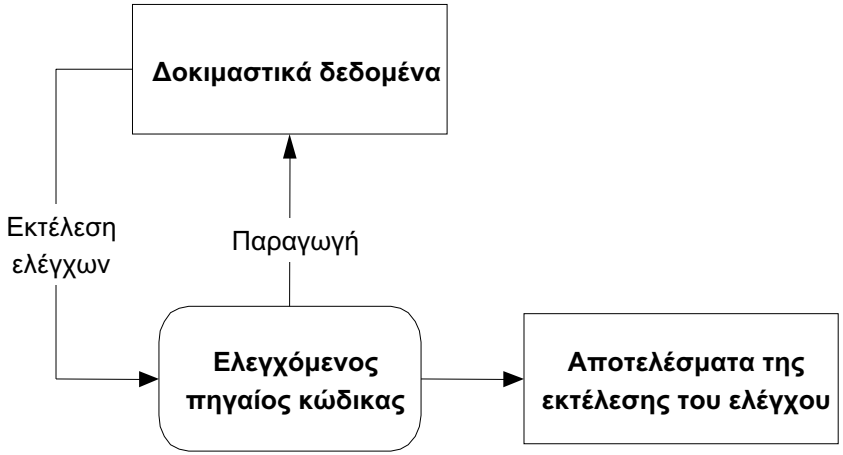
Στρατηγική του "**μαύρου κουτιού**" (black box testing)

- Ο πηγαίος κώδικας δεν είναι γνωστός
- Εκτελούμε το public interface εντός συστατικού

Στρατηγική του "**γυάλινου κουτιού**" (glass box testing)

- Είναι ορατός ο πηγαίος κώδικας και ο έλεγχος αποσκοπεί στην πρόκληση όλων των δυνατών εσωτερικών καταστάσεων: (~ πλήρη κάλυψη κώδικα)

Συνδυασμός (semi-transparent?)



Στρατηγική του μαύρου κουτιού

ΠΡΟΣΕΓΓΙΣΗ ΙΣΟΔΥΝΑΜΗΣ ΔΙΑΜΕΡΙΣΗΣ

Κλάση ισοδυνάμων τιμών

Ενα σύνολο από δοκιμαστικά δεδομένα εισόδου ή δεδομένα εξόδου, τα οποία παράγουν μια ισοδύναμη συμπεριφορά της μονάδας λογισμικού που ελέγχεται
Π.χ: *f(a, b, c) = b*b - 4*a*c* C1 = {(a, b, c), f<0}, C2 = {(a, b, c), f>=0}

Διάκριση κλάσεων ισοδυνάμων τιμών "κατά περίπτωση"

- Από τις συνθήκες εισόδου
- Για κάθε συνδυασμό των μεταβλητών εισόδου που ορίζει ισοδυναμία

Ορισμός περιπτώσεων ελέγχου

- Μέχρι να εξαντληθούν οι κλάσεις ισοδυνάμων τιμών

Στρατηγική του μαύρου κουτιού

ΠΡΟΣΕΓΓΙΣΗ ΣΥΝΟΡΙΑΚΩΝ ΤΙΜΩΝ

Για κάθε μεταβλητή εισόδου

- Ορισμός έγκυρων περιπτώσεων ελέγχου για τα "άκρα" του πεδίου ορισμού
- Ορισμός άκυρων περιπτώσεων, εκτός των άκρων

Διάκριση ορισμών ανάλογα με το είδος του πεδίου ορισμού

- Συνεχείς τιμές
- Σύνολο
- Πίνακας
- Λίστα

Στρατηγική του γυάλινου κουτιού

Ορισμός περιπτώσεων ελέγχου έτσι ώστε να εκτελεστεί όλος ο πηγαίος κώδικας της μονάδας λογισμικού που ελέγχεται

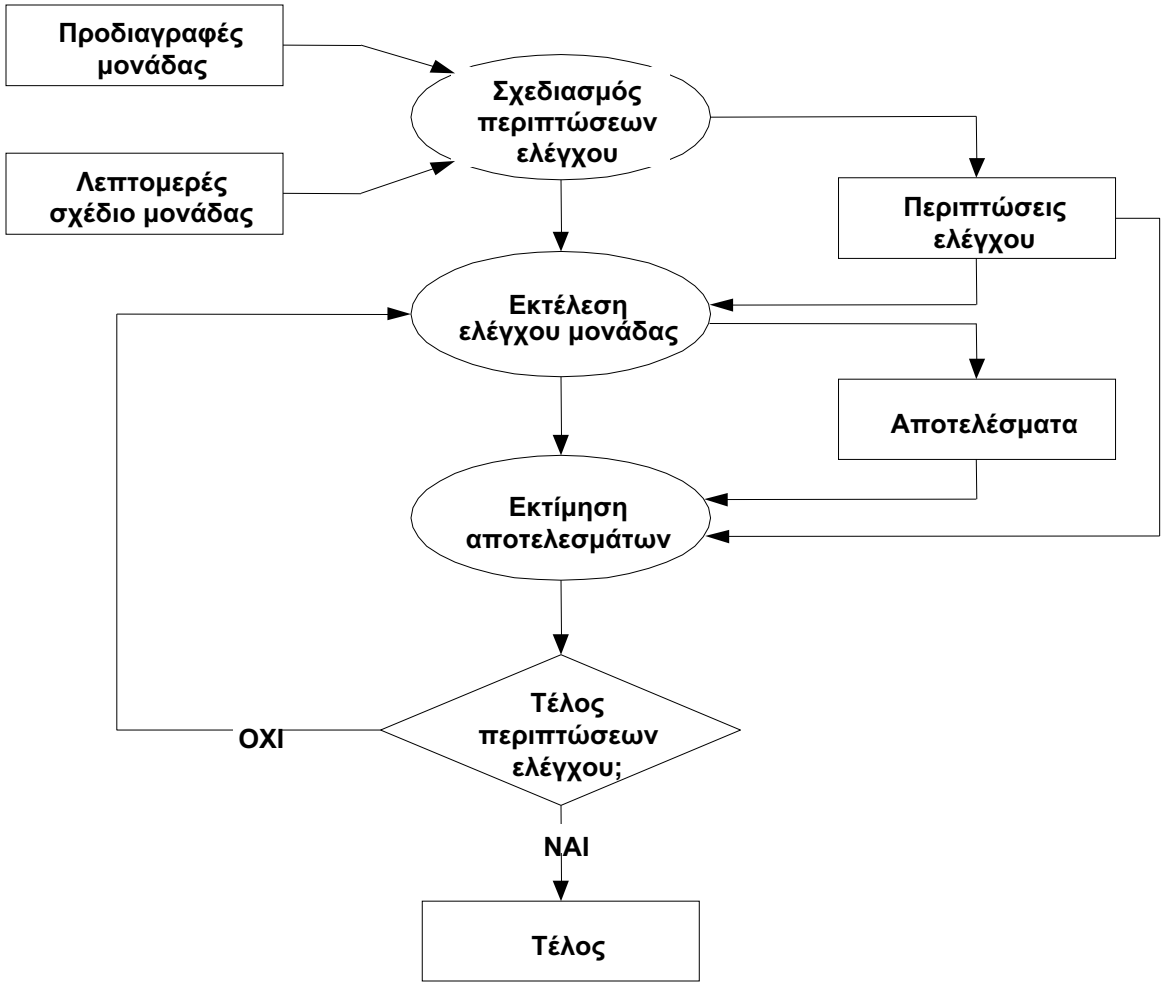
- Όλα τα μονοπάτια εκτέλεσης για τιμές δεδομένων
  - Εντός του διαστήματος ορισμού των συνθηκών επανάληψης
  - Εκτός του διαστήματος
  - Οριακές

Η κάλυψη όλων των εντολών, τουλάχιστον μία φορά δεν σημαίνει ότι η μονάδα που ελέγχεται δεν έχει σφάλματα

Στην πραγματικότητα πρέπει να ελεγχθεί η εκτέλεση του κώδικα σε όλες τις δυνατές συνθήκες (=συνδυασμούς καταστάσεων) αυτού

Ο developer που έγραψε την μονάδα, δεν είναι ο καταλληλότερος για να πραγματοποιήσει τον έλεγχό της (= δεν πρέπει!)

Εκτέλεση του ελέγχου μονάδας (unit testing)



Παράδειγμα ελέγχου μονάδας

Επίλυση του τριωνύμου $a * x^2 + b * x + c = 0$ στο R

```
48      public void Solve() {
49
50          discr=b*b-4*a*c;
51
52          if (discr > 0)
53              flag=0;
54          else
55              if (discr==0)
56                  flag=1;
57              else
58                  flag=2;
59
60          switch (flag) {
61              case 0 :    root1= (-b+Math.sqrt(discr)) / (2*a);
62                          root2= (-b-Math.sqrt(discr)) / (2*a);
63                          break;
64              case 1 : root1= -b / (2*a);
65                          root2=root1;
66                          break;
67              default :
68          }
69      }
```

Παράδειγμα ελέγχου

Μεταβλητές εισόδου

$a \in \Re$, a <> 0

$b \in \Re$

$c \in \Re$

Διαστήματα ενδιαφέροντος

discr = b^2 - 4 * a * c

discr > 0

discr = 0

discr < 0

Παράδειγμα ελέγχου

Περιπτώσεις ελέγχου

a	b	c	Συνθήκες
0	-5	5	a == 0
1	0	-25	a <> 0, discr > 0
1	4	4	a <> 0, discr = 0
1	0	2	a <> 0, discr = 0

Παράδειγμα ελέγχου

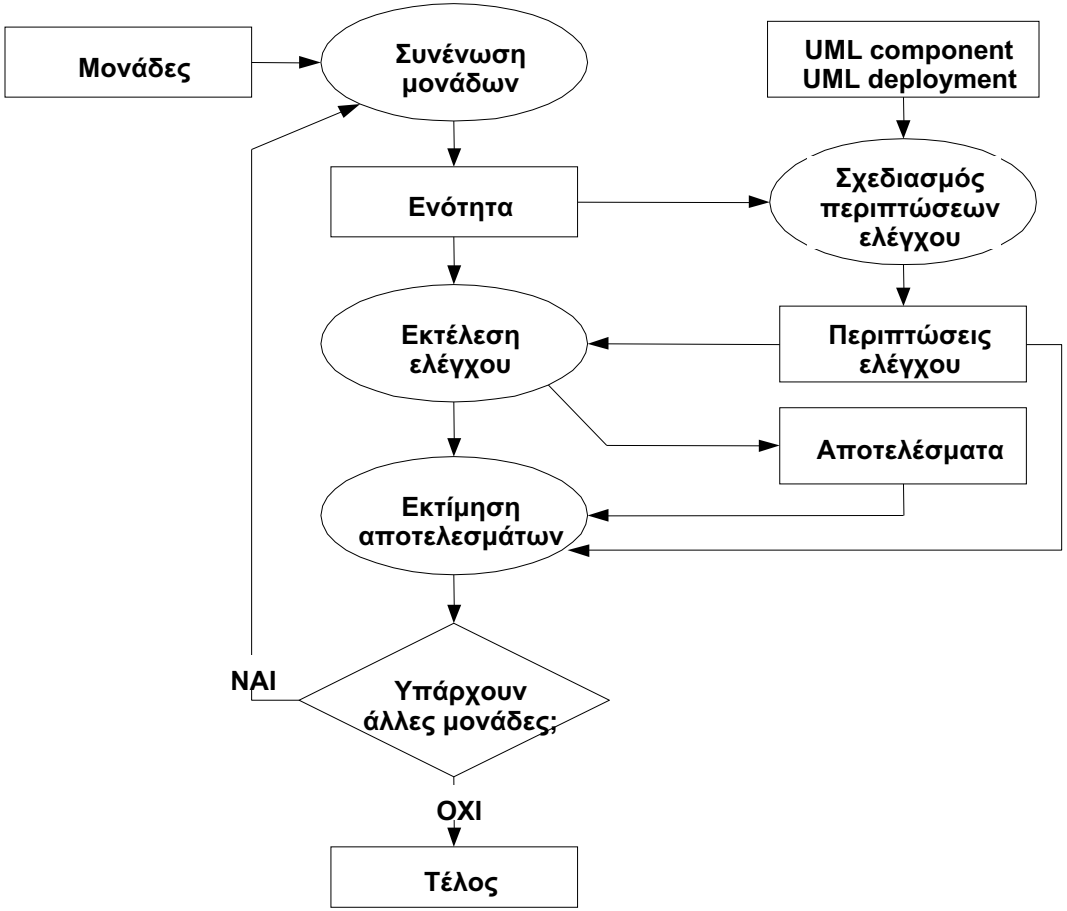
```
48      public void Solve() {
49
50          discr=b*b-4*a*c;
51
52          if (discr > 0)
53              flag=0;
54          else
55              if (discr==0)
56                  flag=1;
57              else
58                  flag=2;
59
60          switch (flag) {
61              case 0 :     root1= (-b+Math.sqrt(discr)) / (2*a);
62                           root2= (-b-Math.sqrt(discr)) / (2*a);
63                           break;
64              case 1 : root1= -b / (2*a);
65                           root2=root1;
66                           break;
67              default :
68          }
69      }
```

a	b	c	ΚΑΛΥΨΗ
0	-5	5	
1	0	-25	
1	4	4	
1	0	2	

Ελεγχος συνένωσης (integration testing)

Δικτυακό λογισμικό:

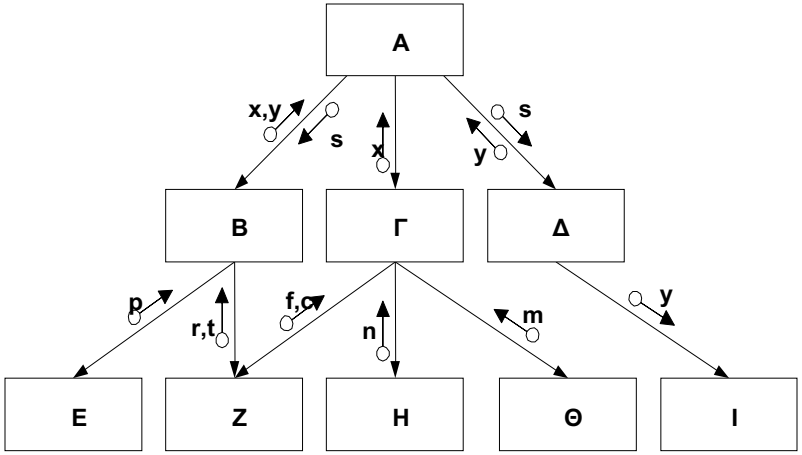
- Συνδέσεις χαμηλού επιπέδου (δίκτυο, session, κλπ)
- Κλήσεις υπηρεσιών, interfaces, APIs κλπ
- Διαχείριση deployments και κοινών ή ανταγωνιστικών πόρων



Εκτέλεση του ελέγχου συνένωσης

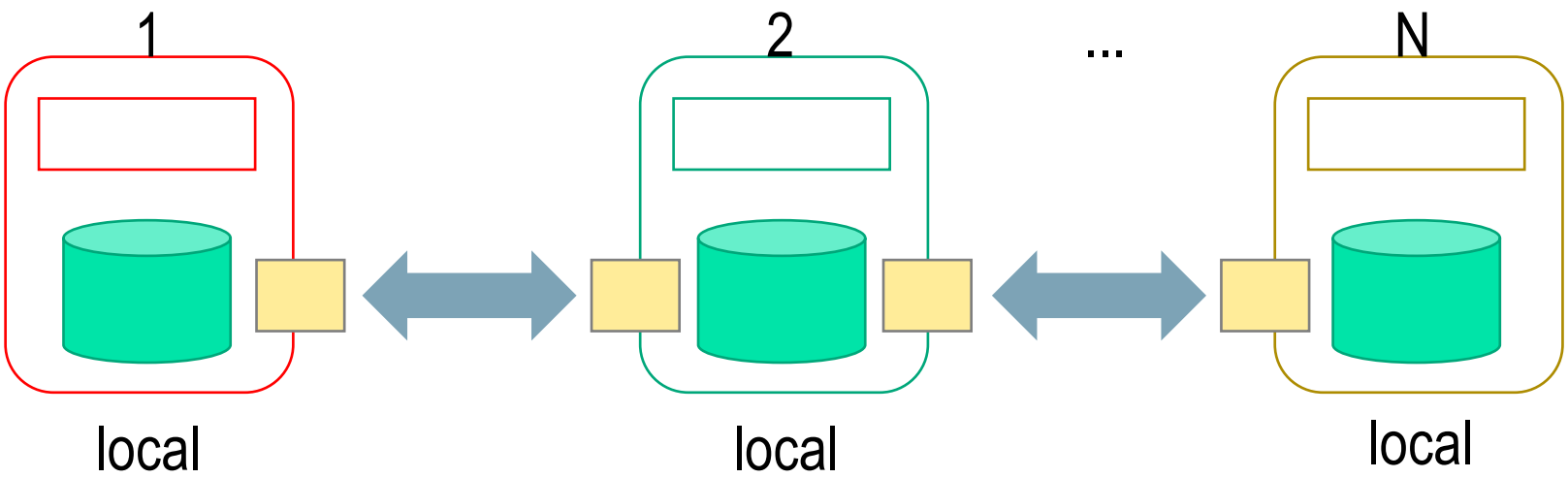
Αυξητική τεχνική

- Από πάνω προς τα κάτω
- Από κάτω προς τα πάνω



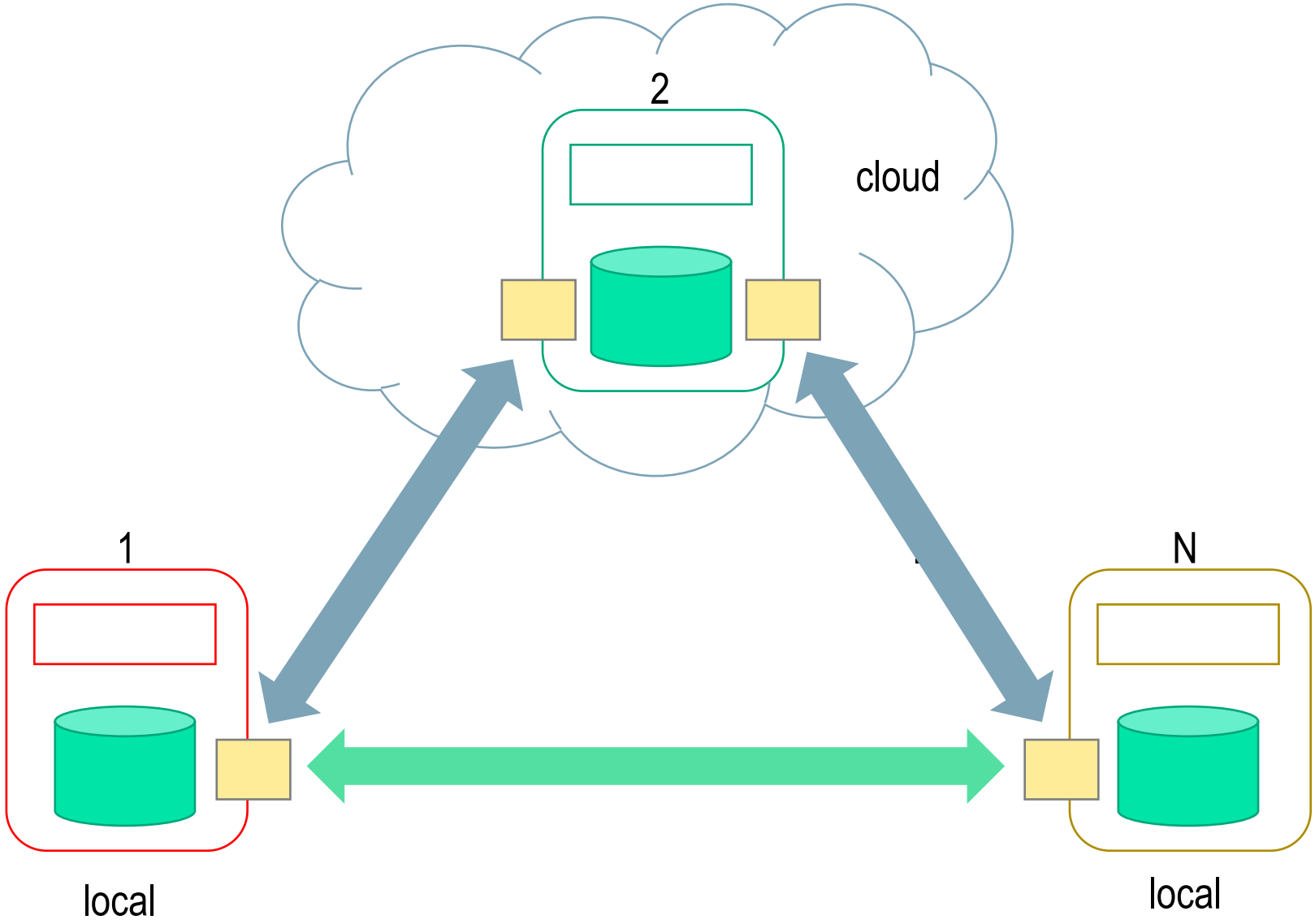
Βήμα	Υποσύστημα που ελέγχεται
1	Α
2	ΑΒ
3	ΑΒΓ
4	ΑΒΓΔ
5	ΑΒΓΔΕ
6	ΑΒΓΔΕΖ
7	ΑΒΓΔΕΖΗ
8	ΑΒΓΔΕΖΗΘ
9	ΑΒΓΔΕΖΗΘΙ

Test vis-a-vis Architecture

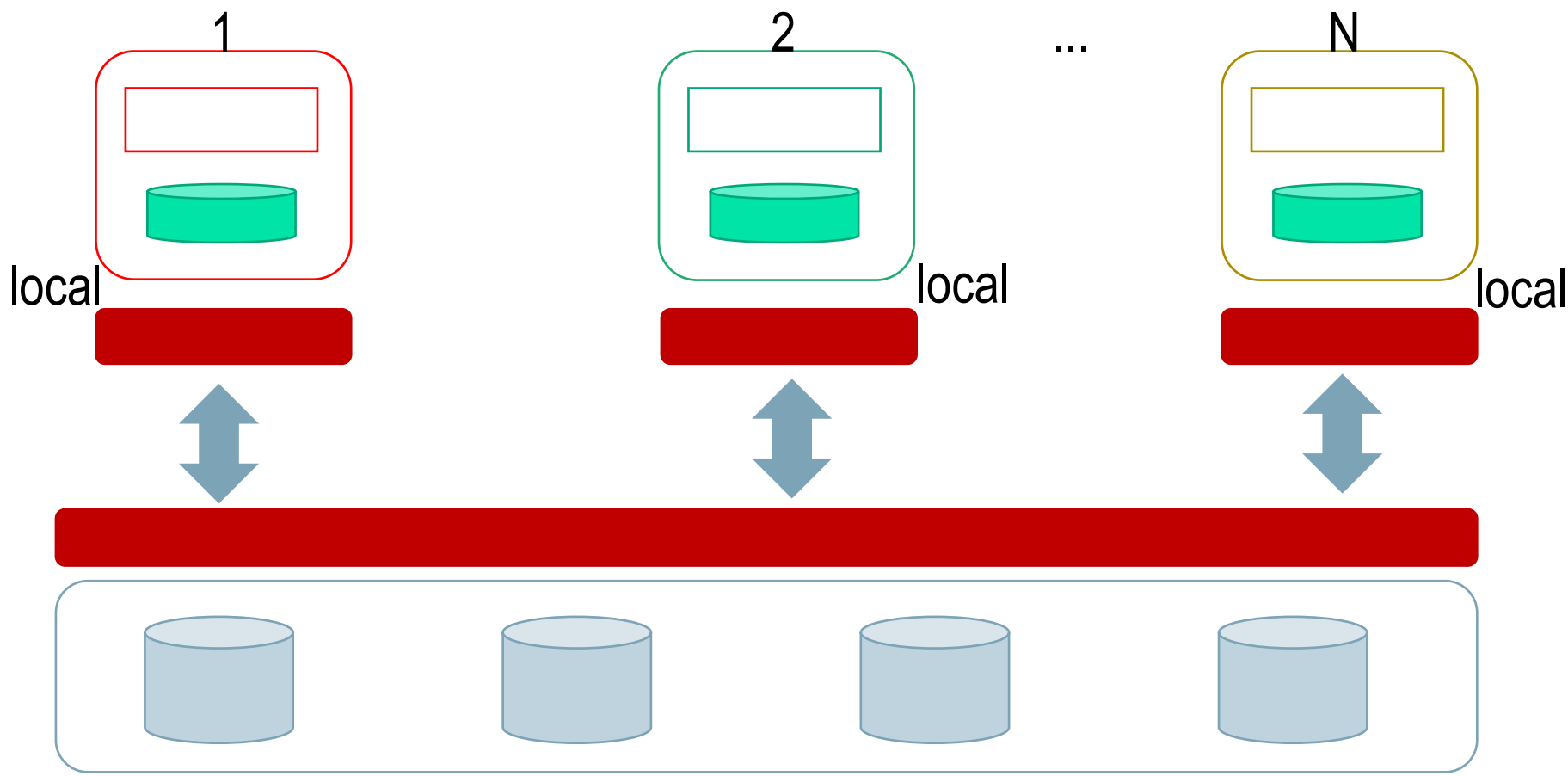


Interoperability interface

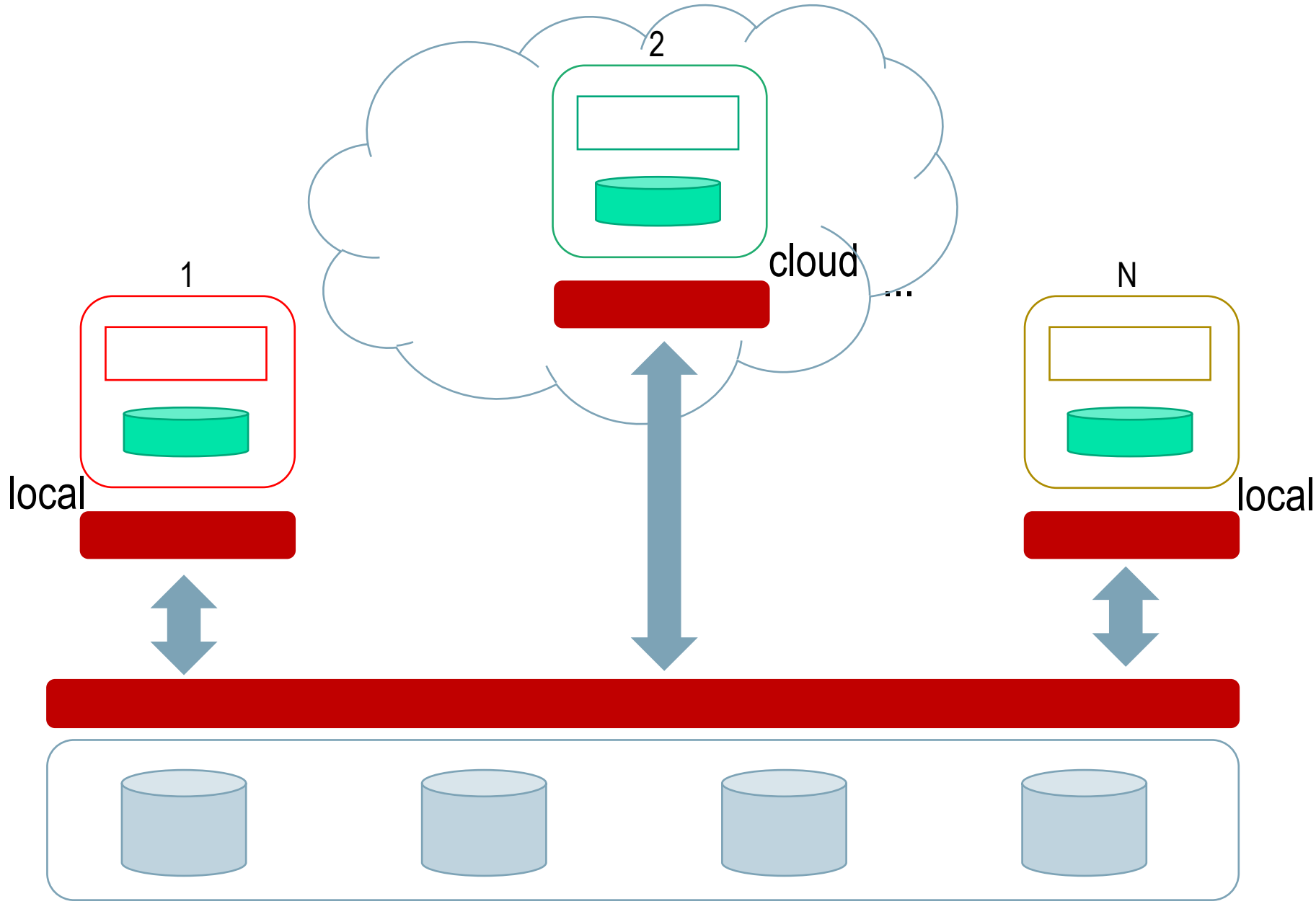
Test vis-a-vis Architecture



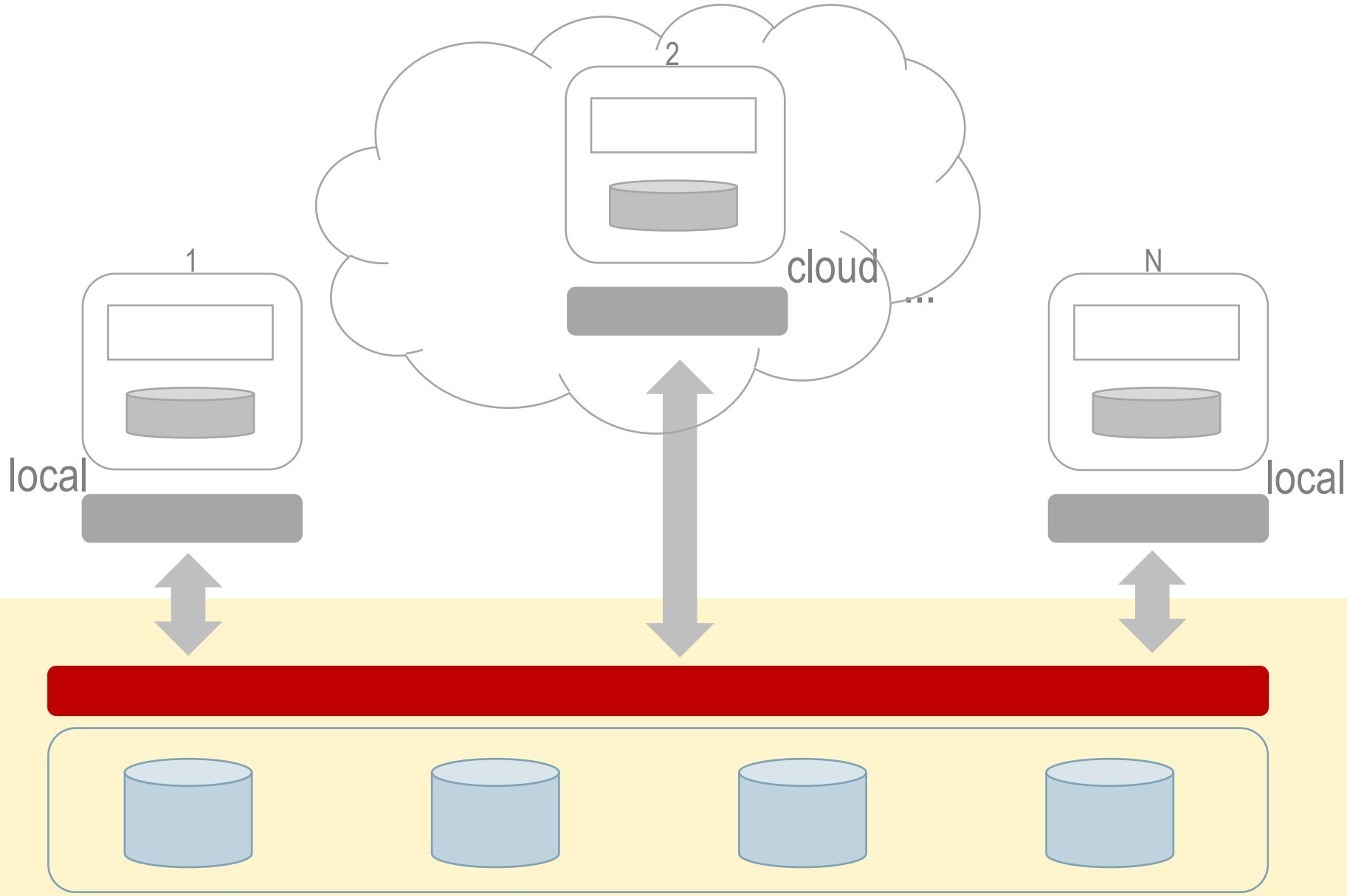
Test vis-a-vis Architecture



Test vis-a-vis Architecture



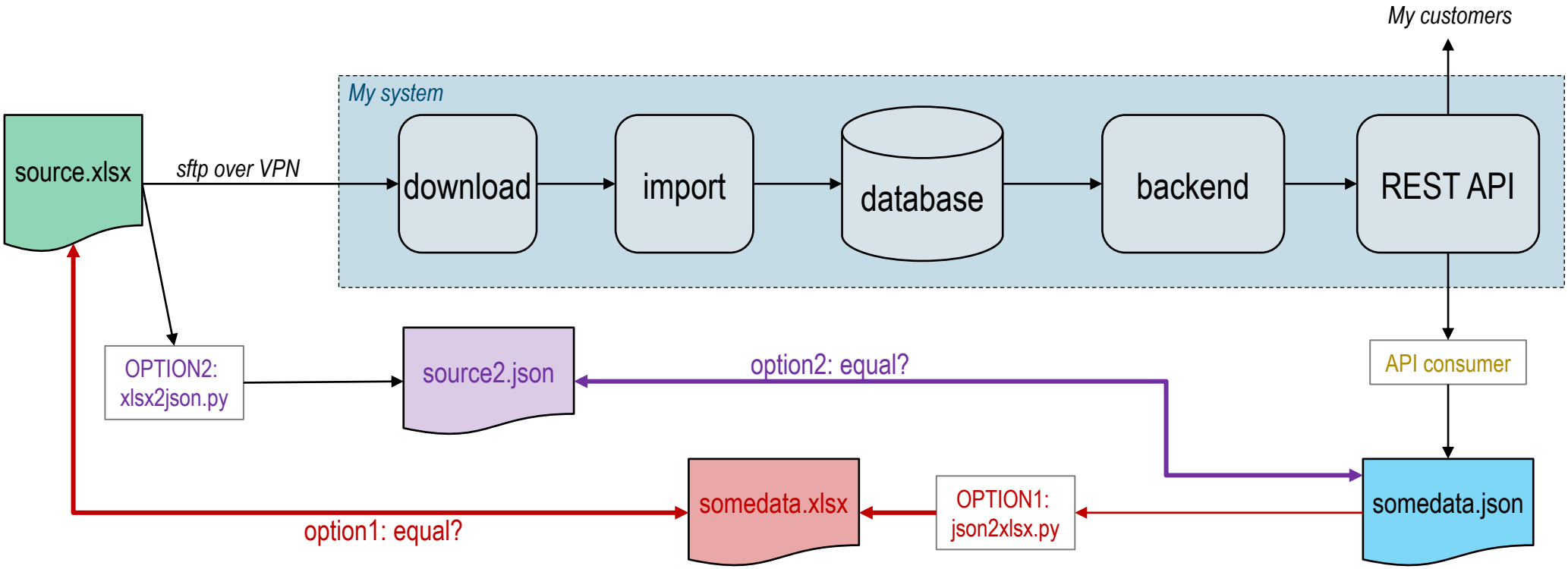
Test vis-a-vis Architecture



Παράδειγμα ελέγχου

Σενάριο:

- Κατεβάζω ένα αρχείο με ανοικτά δεδομένα (πχ excel)
- Το κάνω import σε μια ΒΔ
- Διαθέτω τα περιεχόμενά του μέσω ενός REST API



Ελεγχος συστήματος (system testing)

Ελεγχοι γενικά του συστήματος στο περιβάλλον του

- Επιδόσεις, συμπεριφορά σε φορτίο
- Συμπεριφορά σε ακραίες συνθήκες (partitioning, disaster, κλπ)
- Ογκος δεδομένων
- Ασφάλεια πρόσβασης, δεδομένων, κλπ
- Αποκατάσταση συστήματος (disaster recovery)

Ελεγχος αποδοχής (acceptance testing, validation)

User acceptance testing (UAT) with reference to...

- User expectations
- User needs
- Software/system requirements
- Perceived performance

Important (common) aspects of testing

Smoke testing

Functional testing

Usability testing

Security testing

Performance testing

Stress testing

Spike testing

Soak testing

Regression testing

Smoke testing

Επίπεδα: Integration, System, Acceptance

- Μήπως πήρε φωτιά;
- Δουλεύει τίποτα ή έχουν αποτύχει όλα;
- Κάλυψη όλων των βασικών λειτουργιών οριζόντια, αλλά όχι σε βάθος (κάθετα)
- Να προχωρήσουμε σε επιμέρους δοκιμασίες ή το σύστημα δεν είναι ώριμο ακόμα (θέλει "κι άλλη δουλειά");

Functional testing

Επίπεδα: Unit, System, Acceptance

- Επαλήθευση συμβατότητας με προδιαγραφές και απαιτήσεις
- Καλό είναι να εκτελούνται από τους "χρήστες"
- Domain-driven design, behavior-driven development
- Automated tests/specifications

Usability testing

Επίπεδα: System, Acceptance

- Ad-hoc testing
- A/B testing (σε ευρεία χρήση στις διαδικτυακές εφαρμογές για τη "μέτρηση" της αποτελεσματικότητας μιας UI/UX επιλογής)

Security testing

Μια κατηγορία από μόνο του

Για διαδικτυακές εφαρμογές: owasp.org

Performance / load testing

Μια κατηγορία από μόνο του

- Load testing: stress, spike, soak testing
- Maximum usable capacity (concurrent users, throughput)
- Acceptable response time (latency)
- Παράδειγμα εργαλείου: Apache JMeter

Stress testing

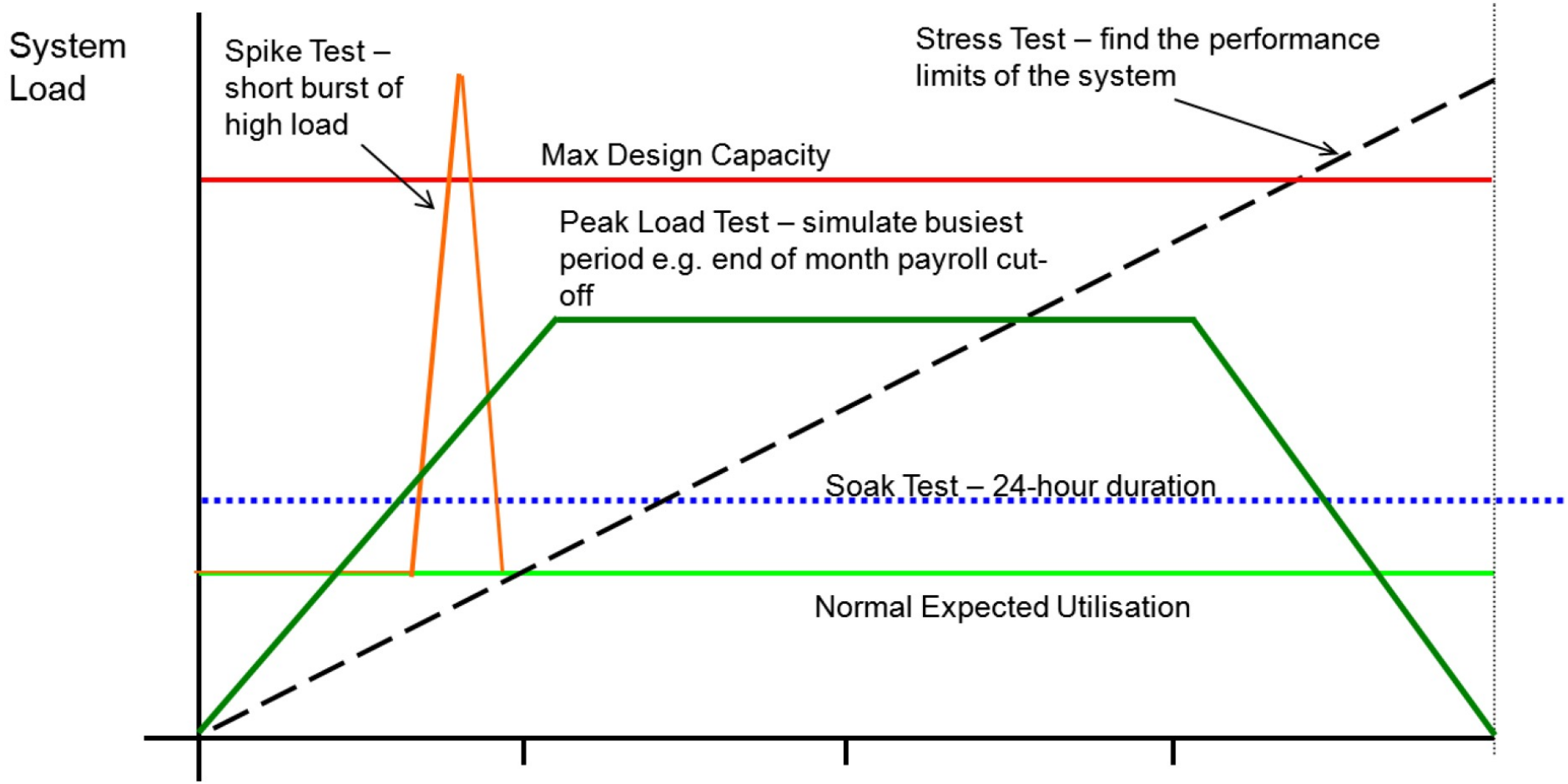
Stress testing (sometimes called torture testing) is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves **testing beyond normal operational capacity**, often to a breaking point, in order to observe the results.

Spike testing

Spike testing verifies a system's stability during **bursts of concurrent user and or system activity** to varying degrees of load over varying time periods.

Soak testing

Soak testing involves testing a system with a typical production load, over a continuous availability period, to validate system behavior under production use.


System Load
Spike Test – short burst of high load
Stress Test – find the performance limits of the system
Max Design Capacity
Peak Load Test – simulate busiest period e.g. end of month payroll cut-off
Soak Test – 24-hour duration
Normal Expected Utilisation

Regression testing

Οπισθοδρόμηση

- Έλεγχος ότι μια νέα έκδοση του λογισμικού (new feature, bug fix, κτλ.) δεν προκαλεί πρόβλημα στην υπάρχουσα λειτουργικότητα
- Από τους βασικούς ελέγχους σε παραγωγικό λογισμικό

Επίσης

Bebugging

- Εκ προθέσεως εισαγωγή σφαλμάτων στον κώδικα με στόχο να ανακαλυφθούν από τις δοκιμασίες

Mutation testing

- Mutation του κώδικα (π.χ. αλλαγή εκφράσεων σε συντακτικό επίπεδο) με αντίστοιχη λογική

Αναφορές ελέγχου

Αναφορά σφάλματος (σε ψηφιακή μορφή, σε κάποιο συνεργατικό περιβάλλον)

1. Ταυτότητα του εγγράφου
2. Ανακεφαλαίωση του ελέγχου που εκτελέστηκε
3. Περιγραφή του ελέγχου και του σφάλματος
4. Επιπτώσεις

Ημερολόγιο ελέγχου

1. Ταυτότητα του ημερολόγιου ελέγχου
2. Περιγραφή
3. Εγγραφές δραστηριοτήτων και γεγονότων
    - 3.1 Περιγραφή εκτέλεσης οντότητας
    - 3.2 Αποτελέσματα εκτέλεσης οντότητας
    - 3.3 Πληροφορίες περιβάλλοντος
    - 3.4 Γεγονότα ανώμαλης συμπεριφοράς
    - 3.5 Ταυτότητες αναφορών σφαλμάτων

Αναφορές ελέγχου

Περιληπτική έκθεση ελέγχου

1. Ταυτότητα της έκθεσης
2. Περίληψη
3. Αποκλίσεις
4. Εκτίμηση περιεκτικότητας
5. Περίληψη αποτελεσμάτων
6. Αξιολόγηση
7. Περίληψη δραστηριοτήτων
8. Εγκρίνοντες

Διόρθωση σφαλμάτων

Στο επίπεδο των απαιτήσεων

- Με οπισθοδρόμηση, τόσο - όσο απαιτείται [μη ικανοποίηση προδιαγραφών, μη υλοποίηση σχεδίου, μη επίτευξη επιχειρηματικού στόχου]
- Κόστος σε χρόνο και όχι μόνο

Στο επίπεδο του πηγαίου κώδικα

- Ανάλογα το βάθος και την εμβέλεια του σφάλματος
- Ενίοτε, η διόρθωση είναι επίπονη, έχει τις δικές της επιπτώσεις και πρέπει να ελεγχθεί επίσης

Η σημασία της τεκμηρίωσης του λογισμικού

- Προδιαγραφές, αρχιτεκτονική, τεκμηρίωση κώδικα
- Υιοθέτηση κανόνων τεκμηρίωσης + κώδικα

Παράδειγμα end-to-end testing με Postman

GET {{protocol}}://{{baseUrl}}/api/v3/data/areareference/GREE Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

```
1  pm.test("Response time is less than 200ms", function () {
2      pm.expect(pm.response.responseTime).to.be.below(200);
3  });
4
5  pm.test("Sampled data is ok", function () {
6      var jsonData = pm.response.json();
7      pm.expect(jsonData[0].AreaRefAbbrev).to.eql("GREECE_GREECE_CTY");
8      pm.expect(jsonData[1].AreaRefAbbrev).to.eql("GREECE_IPTO_BZ_BZN");
9  });
10
11
```

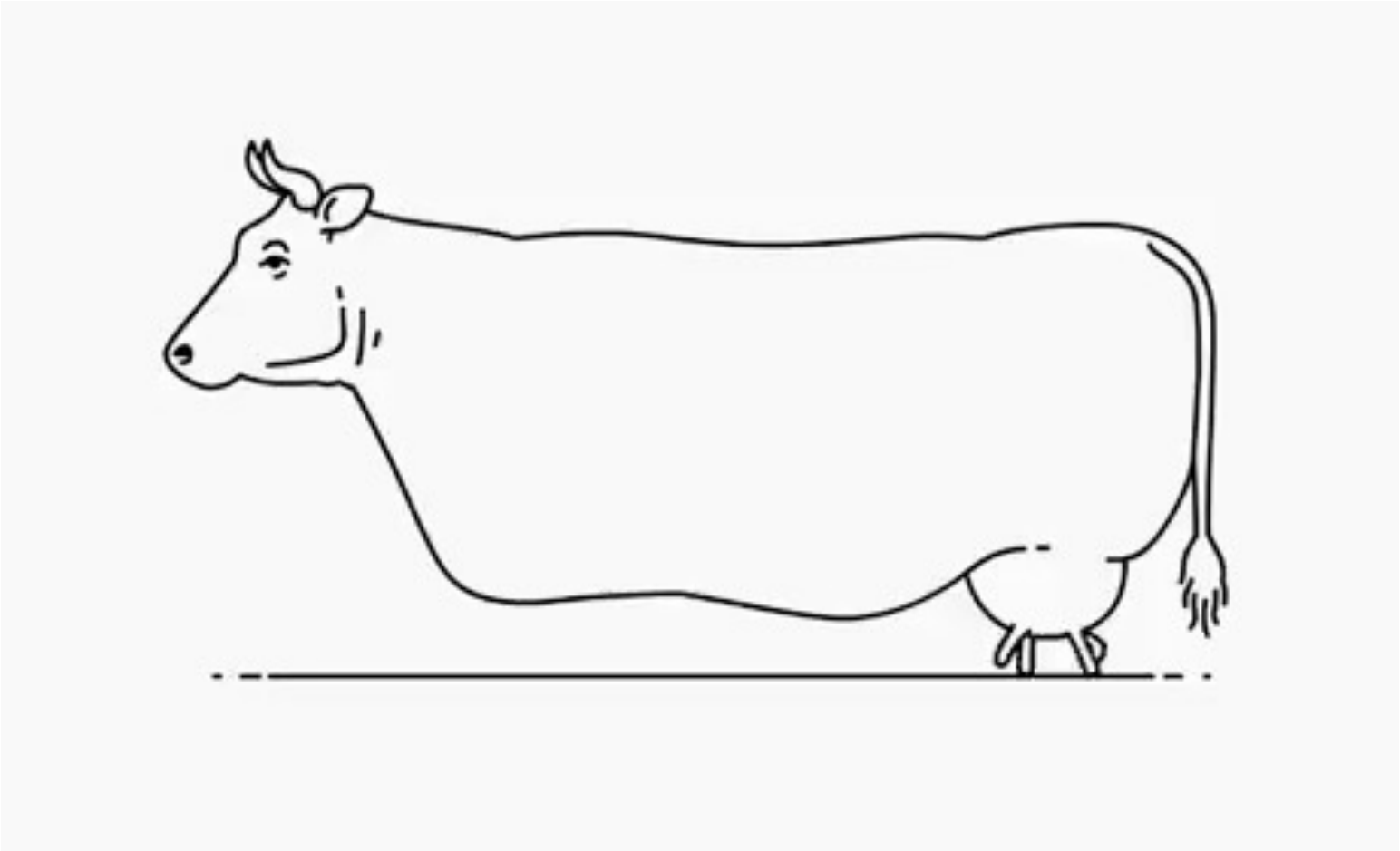
Test scripts are written in JavaScript, and are run after the response is received.
Learn more about tests scripts

SNIPPETS
- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request
- Status code: Code is 200
- Response body: Contains string

Body Cookies (1) Headers (9) Test Results (1/2) Status: 200 OK Time: 310 ms Size: 934 B Save Response

All Passed Skipped Failed

FAIL Response time is less than 200ms | AssertionError: expected 310 to be below 200

PASS Sampled data is ok

Never compromise with this…