ΠΕΡΙΕΧΟΜΕΝΑ

9

Έλεγχος του συστήματος

Σε αυτό το κεφάλαιο εξετάζονται:

- ο έλεγχος λειτουργιών
- ο έλεγχος επίδοσης
- ο έλεγχος αποδοχής συστήματος
- η αξιοπιστία τού λογισμικού, η διαθεσιμότητα και η συντηρησιμότητα
- ο έλεγχος εγκατάστασης
- η έγγραφη τεκμηρίωση ελέγχου
- ο έλεγχος κρισίμων για την ασφάλεια συστημάτων

Ο έλεγχος του συστήματος είναι πολύ διαφορετικός από τον έλεγχο μονάδων και τον έλεγχο ολοκλήρωσης. Όταν πραγματοποιείτε έλεγχο των διαφόρων μονάδων του συστήματος, έχετε πλήρη δικαιοδοσία πάνω στη διαδικασία ελέγχου. Κατασκευάζετε τα δικά σας δεδομένα ελέγχου, σχεδιάζετε τις δικές σας περιπτώσεις ελέγχου και πραγματοποιείτε τους ελέγχους εσείς οι ίδιοι. Όταν πάλι ολοκληρώνετε τα συστατικά ενός λογισμικού, μερικές φορές δουλεύετε μόνοι σας, αλλά συχνά συνεργάζεστε με ένα μικρό τμήμα της ομάδας ελέγχου ή της ομάδας υλοποίησης. Εντούτοις, όταν ελέγχετε ένα σύστημα, δουλεύετε με ολόκληρη την ομάδα υλοποίησης, συντονίζοντας τις εργασίες που πραγματοποιείτε, κατευθυνόμενος ταυτόχρονα από τον υπεύθυνο της ομάδας ελέγχου. Σε αυτό το κεφάλαιο εξετάζουμε τη διαδικασία ελέγχου του συστήματος: το σκοπό της, τα βήματα, τους συμμετέχοντες, τις τεχνικές και τα εργαλεία που χρησιμοποιούνται.

9.1 ΑΡΧΕΣ ΤΟΥ ΕΛΕΓΧΟΥ ΣΥΣΤΗΜΑΤΟΣ

Στόχος του ελέγχου μονάδων και του ελέγχου ολοκλήρωσης ήταν να εξασφαλιστεί ότι ο κώδικας υλοποιούσε σωστά το σχέδιο του λογισμικού· αυτό σημαίνει ότι οι προγραμματιστές έγραψαν τον κώδικα για να υλοποιήσουν αυτό που οι σχεδιαστές είχαν σκοπό να πετύχουν. Στον έλεγχο συστήματος έχουμε ένα δια-

φορετικό στόχο: να εξασφαλίσουμε ότι το σύστημα κάνει αυτό που θέλει να κάνει ο πελάτης. Για να καταλάβουμε πως μπορούμε να επιτύχουμε αυτό το στόχο, πρέπει πρώτα να κατανοήσουμε από πού προέρχονται τα σφάλματα ενός συστήματος.

Πηγές σφαλμάτων λογισμικού

Ας θυμηθούμε ότι ένα σφάλμα λογισμικού προκαλεί αποτυχία μόνο όταν συνοδεύεται από τις κατάλληλες συνθήκες. Αυτό σημαίνει ότι μπορεί να υπάρχει ένα σφάλμα μέσα στον κώδικα του λογισμικού, αλλά εάν ο κώδικας δεν εκτελεστεί ποτέ, ή ο κώδικας δεν εκτελεστεί για αρκετό διάστημα ή με την κατάλληλη διαμόρφωση συνθηκών για να προκαλέσει πρόβλημα, μπορεί το λογισμικό μας να μην αποτύχει ποτέ. Επειδή κατά τον έλεγχο δεν μπορεί να εφαρμοστεί κάθε πιθανή συνθήκη, διατηρούμε σαν σκοπό μας την ανακάλυψη των σφαλμάτων, ελπίζοντας ότι στη διαδικασία ελέγχου θα εξαλείψουμε όλα τα σφάλματα που μπορούν να οδηγήσουν σε αποτυχίες κατά την πραγματική χρήση του υλοποιημένου συστήματος.

Τα σφάλματα του λογισμικού μπορούν να εισαχθούν στις απαιτήσεις, στο σχέδιο ή στον κώδικα, ή στην τεκμηρίωση του λογισμικού, σε κάθε στιγμή κατά τη διάρκεια της υλοποίησης ή της συντήρησης του λογισμικού. Στην Εικόνα 9.1 παρουσιάζονται οι πιθανές αιτίες των σφαλμάτων σε κάθε δραστηριότητα ανάπτυξης. Αν και θα θέλαμε να εντοπίζουμε και να διορθώνουμε τα σφάλματα όσο το δυνατόν νωρίτερα, η διαδικασία ελέγχου του συστήματος λαμβάνει υπόψη ότι μπορεί να υπάρχουν σφάλματα ακόμα και μετά τον έλεγχο ολοκλήρωσης.

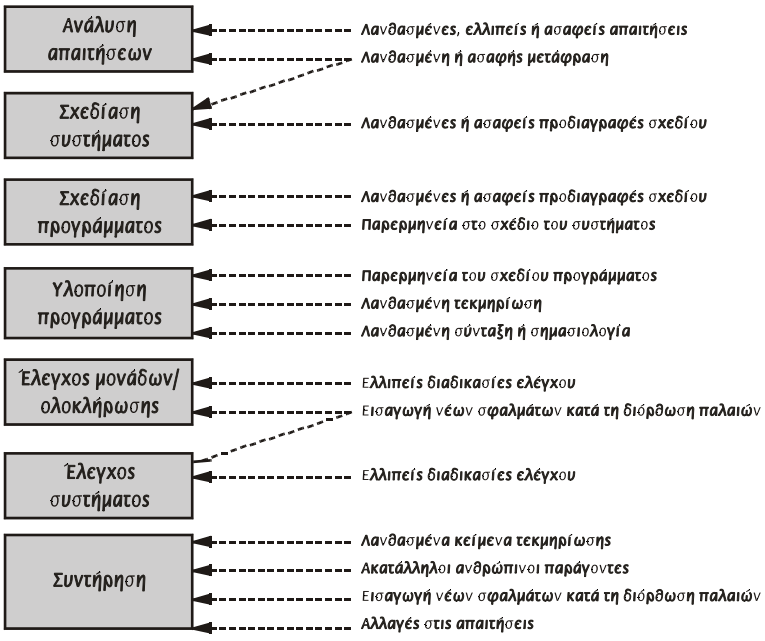
Σφάλματα μπορούν να εισαχθούν στο σύστημα ανά πάσα στιγμή είτε στην αρχή της υλοποίησης, είτε στη διάρκεια, είτε στο τέλος, είτε ακόμα και διορθώνοντας ένα σφάλμα που μόλις ανακαλύφθηκε. Για παράδειγμα, μπορούμε να καταλήξουμε σε ελαττωματικό λογισμικό από σφάλμα στις απαιτήσεις του συστήματος. Είτε μία απαίτηση ήταν ασαφής εξαιτίας της αβεβαιότητας του πελάτη για τη συγκεκριμένη απαίτηση, είτε επειδή έγινε παρερμηνεία των αναγκών του πελάτη, το αποτέλεσμα τελικά είναι το ίδιο: καταλήγουμε σε ένα σύστημα το οποίο δεν λειτουργεί με τον τρόπο κατά τον οποίο ο πελάτης θα ήθελε να λειτουργεί.

Σφάλματα κακής επικοινωνίας του ίδιου τύπου μπορούν να προκύψουν και κατά τη διάρκεια της σχεδίασης του συστήματος. Μπορεί να παρερμηνεύσουμε μία απαίτηση και να καθορίσουμε μία λανθασμένη προδιαγραφή κατά τη σχεδίαση του συστήματος. Μπορεί ακόμη να κατανοήσουμε την απαίτηση αλλά να εκφράσουμε την προδιαγραφή με πολύ φτωχό τρόπο ώστε αυτοί που θα τη διαβάσουν και θα χρησιμοποιήσουν το προτεινόμενο σχέδιο στη συνέχεια να παρερμηνεύσουν το στόχο της προδιαγραφής. Με παρόμοιο τρόπο μπορεί να κάνουμε διάφορες υποθέσεις σχετικά με τα χαρακτηριστικά του συστήματος και τις σχέ-

σεις μεταξύ τους, οι οποίες όμως να μη γίνονται αντιληπτές κατά τον ίδιο τρόπο από τους υπόλοιπους αναγνώστες των προδιαγραφών του συστήματος.

Παρόμοια συμβάντα μπορούν να οδηγήσουν σε σχεδιαστικά σφάλματα του συστήματος. Οι παρερμηνείες είναι συνήθεις όταν το συνολικό σχέδιο του συστήματος μεταφράζεται σε περιγραφές χαμηλότερου επιπέδου για ανάλυση των προδιαγραφών σχεδίου του προγράμματος. Οι προγραμματιστές (υλοποιητές του συστήματος) συνήθως δεν συμμετέχουν στις αρχικές συνομιλίες με τους πελάτες, οι οποίες σχετίζονται με τον καθορισμό των στόχων και των λειτουργιών του συστήματος. Έχοντας την ευθύνη για ένα μόνο "δένδρο" και όχι για όλο το "δάσος", οι προγραμματιστές δεν θα πρέπει να αναμένεται να εντοπίζουν σφάλματα σχεδίου τα οποία έχουν εισαχθεί κατά τα πρώτα βήματα του κύκλου υλοποίησης του λογισμικού. Για το λόγο αυτό, η επανεξέταση των απαιτήσεων και του σχεδίου είναι απαραίτητες για να εξασφαλιστεί η ποιότητα του τελικού συστήματος που θα υλοποιηθεί.

Οι προγραμματιστές και οι σχεδιαστές στην ομάδα υλοποίησης που έχει συσταθεί μπορεί να αποτύχουν και αυτοί να χρησιμοποιήσουν την κατάλληλη σύνταξη και σημασιολογία για την καταγραφή της εργασίας τους. Ένας μεταγλωττιστής ή ένας συμβολομεταφραστής μπορεί να εντοπίσει μερικά από αυτά τα σφάλματα πριν να τρέξει ένα πρόγραμμα, αλλά δεν μπορούν να εντοπιστούν



Εικόνα 9.1 Αιτίες σφαλμάτων κατά την υλοποίηση.

σφάλματα όταν η μορφή μίας εντολής είναι σωστή αλλά δεν ταιριάζει με τις προθέσεις του προγραμματιστή ή του σχεδιαστή του συστήματος.

Όταν ξεκινήσει ο έλεγχος των συστατικών μερών του προγράμματος, μπορεί να εισαχθούν σφάλματα ακουσίως στην προσπάθεια διόρθωσης άλλων προβλημάτων. Αυτά τα σφάλματα είναι συχνά πολύ δύσκολο να εντοπιστούν, επειδή μπορεί να εμφανίζονται μόνο όταν χρησιμοποιούνται συγκεκριμένες λειτουργίες, ή μόνο υπό συγκεκριμένες συνθήκες. Αν αυτές οι λειτουργίες έχουν ήδη ελεγχθεί όταν ένα νέο σφάλμα εισαχθεί ακουσίως, το νέο σφάλμα μπορεί να μην εντοπιστεί για πολύ χρόνο και όταν εντοπιστεί να μην είναι πλέον σαφής η πηγή προέλευσης. Αυτή η κατάσταση είναι πιθανό να συμβεί όταν κάνουμε επαναχρησιμοποίηση κώδικα που γράφτηκε για άλλη εφαρμογή, και τον μετατρέψουμε για να ικανοποιήσει τις τρέχουσες ανάγκες μας. Οι λεπτές διαφορές στο σχέδιο του κώδικα μπορεί να μην είναι εμφανείς και οι αλλαγές μας μπορεί να προκαλέσουν περισσότερο ζημιά παρά καλό.

Για παράδειγμα, ας υποθέσουμε ότι ελέγχετε τα συστατικά A, B και C. Επίσης ας υποθέσουμε ότι τα ελέγχετε ξεχωριστά. Όταν τα ελέγξετε και τα τρία μαζί, παρατηρείτε ότι το A μεταβιβάζει λανθασμένα μία παράμετρο στο C. Διορθώνοντας το A, εξασφαλίζετε ότι η μεταβίβαση της παραμέτρου γίνεται σωστά, αλλά πραγματοποιείτε προσθήκες νέου κώδικα ο οποίος δίνει λανθασμένη τιμή σε ένα δείκτη. Επειδή μπορεί να μην επιστρέψετε πίσω και να κάνετε ξανά ξεχωριστό έλεγχο στο A, μπορεί να μην εντοπίσετε το νέο σφάλμα για πολύ χρόνο και μέχρι να πραγματοποιήσετε κάποιον έλεγχο, οπότε πλέον μπορεί να μην είναι σαφές ότι το A δημιουργεί το πρόβλημα.

Με παρόμοιο τρόπο, η συντήρηση μπορεί να εισάγει νέα σφάλματα. Οι βελτιώσεις του συστήματος απαιτούν αλλαγές στις αρχικές απαιτήσεις σχεδίου, στην αρχιτεκτονική του συστήματος, στο σχέδιο του προγράμματος, και στην υλοποίηση, ώστε πολλοί τύποι σφαλμάτων μπορεί να εισαχθούν καθώς περιγράφεται, σχεδιάζεται, και υλοποιείται η βελτίωση. Επιπρόσθετα, το σύστημα μπορεί να μη λειτουργεί σωστά επειδή οι χρήστες δεν κατανοούν πώς είχε σχεδιαστεί για να λειτουργεί. Εάν το εγχειρίδιο τεκμηρίωσης είναι ασαφές ή περιέχει λάθη, μπορεί να προκύψουν σφάλματα κατά τη χρήση. Ανθρώπινοι παράγοντες, στους οποίους συμπεριλαμβάνονται και η διαφορετική αντίληψη και εξοικείωση του κάθε χρήστη, παίζουν ένα σημαντικό ρόλο στην κατανόηση του συστήματος και στην ερμηνεία των μηνυμάτων του και των απαιτούμενων δεδομένων εισόδου. Χρήστες που δεν έχουν μεγάλη εξοικείωση με το σύστημα μπορεί να μη χρησιμοποιούν κατάλληλα τις λειτουργίες του συστήματος ή να μην εκμεταλλεύονται όλα τα πλεονεκτήματα που παρέχει το λογισμικό.

Οι διαδικασίες ελέγχου θα πρέπει να είναι αρκετά εξονυχιστικές ώστε να ελέγξουν τις λειτουργίες του συστήματος και να ικανοποιηθούν όλοι: οι χρήστες,

οι πελάτες, και οι υλοποιητές. Εάν οι έλεγχοι είναι ατελείς, τα σφάλματα μπορεί να μην εντοπιστούν καθόλου. Όπως έχουμε δει, όσο γρηγορότερα εντοπίσουμε ένα σφάλμα τόσο το καλύτερο· τα σφάλματα που εντοπίζονται νωρίς διορθώνονται ευκολότερα και με μικρότερο κόστος. Συνεπώς, ο πλήρης και έγκαιρος έλεγχος μπορεί να βοηθήσει όχι μόνο στον γρήγορο εντοπισμό των λαθών, αλλά και στην απομόνωση των αιτιών τους πολύ πιο εύκολα.

Στην Εικόνα 9.1 παρουσιάζονται οι λόγοι εμφάνισης των σφαλμάτων και όχι οι ενδείξεις αναγνώρισης αυτών. Επειδή ο έλεγχος έχει σκοπό τον εντοπισμό όσο το δυνατόν περισσότερων σφαλμάτων, ασχολείται με τις περιοχές εντοπισμού τους. Γνωρίζοντας τους τρόπους δημιουργίας των σφαλμάτων έχουμε υπόψη που να ψάξουμε όταν ελέγχουμε ένα σύστημα.

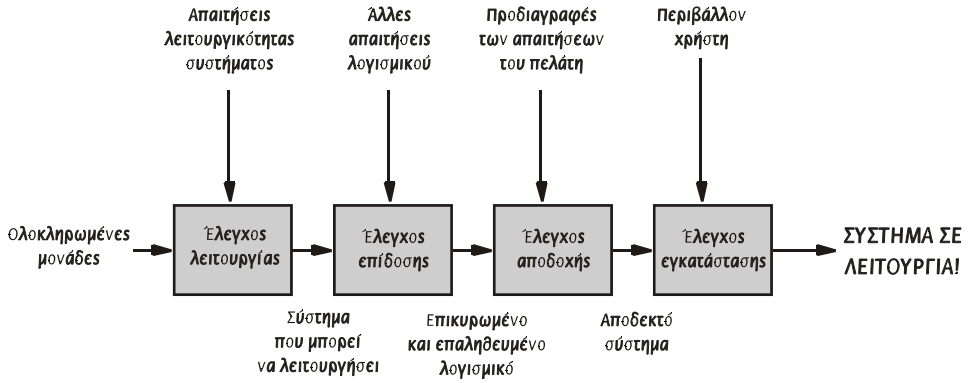
Διαδικασία ελέγχου συστήματος

Υπάρχουν αρκετά βήματα κατά τον έλεγχο ενός συστήματος:

1. Έλεγχος λειτουργιών
2. Έλεγχος επίδοσης
3. Έλεγχος αποδοχής
4. Έλεγχος εγκατάστασης

Τα βήματα παρουσιάζονται στην Εικόνα 9.2. Κάθε βήμα έχει διαφορετική εστίαση, ενώ η επιτυχία ενός βήματος εξαρτάται από το στόχο ή το σκοπό του. Συνεπώς θα ήταν χρήσιμο να εξετάσουμε το σκοπό κάθε βήματος της διαδικασίας ελέγχου του συστήματος.

Στόχοι διεργασιών. Αρχικά ελέγχουμε τις λειτουργίες που εκτελούνται από το σύστημα. Ξεκινάμε με ένα σύνολο συστατικών τα οποία είχαν ελεγχθεί ξεχω-



Εικόνα 9.2 Τα βήματα κατά τη διαδικασία ελέγχου.

ριστά και έπειτα όλα μαζί. Ένας **έλεγχος λειτουργίας** (function test) πραγματοποιείται για να επαληθεύσει ότι το ολοκληρωμένο σύστημα εκτελεί τις λειτουργίες του όπως είναι καθορισμένες στις απαιτήσεις του συστήματος. Για παράδειγμα, ο έλεγχος λειτουργίας ενός πακέτου τραπεζικού λογαριασμού επαληθεύει ότι αυτό ενημερώνει σωστά το υπόλοιπο μετά από μία κατάθεση, δίνει τη δυνατότητα απόσυρσης των χρημάτων, υπολογίζει το επιτόκιο, παρουσιάζει το υπόλοιπο του λογαριασμού και άλλες τραπεζικές λειτουργίες.

Μόλις η ομάδα ελέγχου πειστεί ότι οι λειτουργίες δουλεύουν όπως είναι καθορισμένο να δουλεύουν, ο **έλεγχος επίδοσης** (performance test) συγκρίνει τα ολοκληρωμένα συστατικά με τις μη λειτουργικές απαιτήσεις του συστήματος. Αυτές οι απαιτήσεις, στις οποίες συμπεριλαμβάνονται η ασφάλεια, η ακρίβεια, η ταχύτητα και η αξιοπιστία, περιορίζουν τον τρόπο με τον οποίο εκτελούνται οι λειτουργίες του συστήματος. Για παράδειγμα, ο έλεγχος επίδοσης ενός πακέτου τραπεζικού λογαριασμού εκτιμά την ταχύτητα με την οποία γίνονται οι υπολογισμοί, την ακρίβεια των υπολογισμών, τα μέτρα ασφαλείας που απαιτούνται και τον χρόνο απόκρισης του συστήματος στα ερωτήματα του χρήστη.

Σε αυτό το σημείο, το σύστημα λειτουργεί με τον τρόπο κατά τον οποίο είχε σχεδιαστεί να λειτουργεί. Επομένως, καλούμε αυτό το σύστημα **επαληθευμένο σύστημα** (verified system)· Το σύστημα αυτό εκφράζει την ερμηνεία των προδιαγεγραμμένων απαιτήσεων από το σχεδιαστή. Έπειτα συγκρίνουμε το σύστημα με τις προσδοκίες του πελάτη εξετάζοντας τα έγγραφα καθορισμού απαιτήσεων. Εάν είμαστε ικανοποιημένοι με το σύστημα που κατασκευάσαμε θεωρώντας ότι ανταποκρίνεται στις απαιτήσεις, τότε έχουμε ένα **επικυρωμένο σύστημα** (validated system)· αυτό σημαίνει ότι έχουμε επαληθεύσει ότι το σύστημα ικανοποιεί τις προκαθορισμένες απαιτήσεις.

Μέχρι τώρα όλοι οι έλεγχοι έχουν εκτελεστεί από τους κατασκευαστές του συστήματος και βασίζονται στην κατανόηση που έχουν οι κατασκευαστές σχετικά με το σύστημα και τους στόχους του. Επιπλέον, το σύστημα ελέγχεται και από τους πελάτες, ώστε να βεβαιωθούμε ότι ανταποκρίνεται στις ανάγκες και τις απαιτήσεις τους, οι οποίες μπορεί να είναι διαφορετικές από την άποψη που έχουν σχηματίσει για αυτές οι κατασκευαστές του συστήματος. Ο έλεγχος αυτός, που καλείται **έλεγχος αποδοχής** (acceptance test), διαβεβαιώνει τους πελάτες ότι το σύστημα που ζήτησαν είναι το σύστημα που τελικά κατασκευάστηκε. Ο έλεγχος αποδοχής μερικές φορές εκτελείται στην πραγματική πλατφόρμα λειτουργίας, όμως τις περισσότερες φορές εκτελείται σε πλατφόρμες λειτουργίας διαφορετικές από την πραγματική. Για το λόγο αυτό μπορούμε να εκτελέσουμε ένα τελικό **έλεγχο εγκατάστασης** (installation test) ώστε να δώσουμε τη δυνατότητα στους χρήστες να ελέγξουν τις λειτουργίες του συστήματος και να καταγράψουν επιπρόσθετα προβλήματα που θα εμφανιστούν (τα οποία προέρχονται από το πραγ-

ματικό περιβάλλον λειτουργίας). Για παράδειγμα, ένα σύστημα για χρήση πάνω σε πλοίο μπορεί να σχεδιαστεί, να κατασκευαστεί και να ελεγχθεί στον τόπο κατασκευής του με συνθήκες διαμορφωμένες σαν συνθήκες πλοίου, οι οποίες βεβαίως δεν είναι οι πραγματικές συνθήκες λειτουργίας, απλά μία προσομοίωση. Μόλις ολοκληρωθούν οι έλεγχοι στο χώρο του κατασκευαστή, θα πρέπει να πραγματοποιηθεί ένα επιπλέον σύνολο ελέγχων εγκατάστασης με το σύστημα πάνω στο πραγματικό πλοίο (για το οποίο κατασκευάστηκε το σύστημα).

Πλάνο κατασκευής ή ολοκλήρωσης. Στην ιδανική περίπτωση, μετά τον έλεγχο του προγράμματος, μπορούμε να δούμε τη συλλογή των συστατικών μερών σαν μία ολότητα. Έπειτα, κατά τη διάρκεια των πρώτων βημάτων ελέγχου του συστήματος, η ολοκληρωμένη αυτή συλλογή αξιολογείται από ποικίλες διαφορετικές όψεις όπως περιγράφηκε στις προηγούμενες παραγράφους. Ωστόσο, τα μεγάλα συστήματα μερικές φορές είναι δύσχρηστα όταν ελέγχονται ως μία τεράστια συλλογή συστατικών μερών. Στην πράξη τέτοια συστήματα είναι πολλές φορές υποψήφια για ανάπτυξη σε πολλαπλές φάσεις, απλά επειδή είναι πολύ πιο εύκολο να κατασκευαστούν και να ελεγχθούν ανά μικρότερα λειτουργικά τμήματα. Συνεπώς μπορείτε να επιλέξετε την εκτέλεση του ελέγχου του συστήματος σε πολλαπλές φάσεις. Στο Κεφάλαιο 1* δείξαμε ότι μπορούμε να αντιληφθούμε ένα σύστημα ως ένα σύνολο από φωλιασμένα επίπεδα ή υποσυστήματα. Κάθε επίπεδο είναι υπεύθυνο για την εκτέλεση τουλάχιστον των λειτουργιών των υποσυστημάτων που περιέχει. Παρόμοια, μπορούμε να χωρίσουμε το σύστημα για έλεγχο σε μία ακολουθία φωλιασμένων υποσυστημάτων και να εκτελέσουμε τον έλεγχο συστήματος σε ένα υποσύστημα κάθε φορά.

Ο ορισμός του κάθε υποσυστήματος βασίζεται σε προκαθορισμένα κριτήρια. Συνήθως, η βάση του διαχωρισμού είναι η λειτουργικότητα του κάθε υποσυστήματος. Για παράδειγμα, είδαμε στο Κεφάλαιο 8 ότι η Microsoft διαχωρίζει ένα προϊόν σε τρία υποσυστήματα με βάση τις πιο κρίσιμες λειτουργίες, τις επιθυμητές λειτουργίες και τις λιγότερο απαραίτητες λειτουργίες. Παρόμοια ένα σύστημα τηλεπικοινωνιών το οποίο δρομολογεί κλήσεις μπορεί να χωριστεί σε υποσυστήματα σύμφωνα με τον ακόλουθο τρόπο:

1. Δρομολόγηση κλήσεων μέσα σε ένα τηλεφωνικό κέντρο
2. Δρομολόγηση κλήσεων μέσα σε ένα κωδικό περιοχής
3. Δρομολόγηση κλήσεων μέσα σε ένα νομό, ζώνη ή περιοχή
4. Δρομολόγηση κλήσεων μέσα σε μία χώρα
5. Δρομολόγηση κλήσεων στο εξωτερικό

---

* 1ος Τόμος

Το καθένα από τα μεγαλύτερα υποσυστήματα περιέχει όλα τα προγενέστερά του υποσυστήματα. Ξεκινάμε τον έλεγχο του συστήματός μας ελέγχοντας τις λειτουργίες του πρώτου συστήματος. Όταν όλες οι λειτουργίες του υποσυστήματος δρομολόγησης κλήσεων μέσα στο τηλεφωνικό κέντρο ελεγχθούν επιτυχώς, προχωράμε στον έλεγχο του δευτέρου συστήματος. Παρόμοια, ελέγχουμε το δεύτερο, τρίτο, τέταρτο και πέμπτο σύστημα. Το αποτέλεσμα είναι ένας επιτυχημένος έλεγχος του συνολικού συστήματος, όμως αυτός ο έλεγχος σε επίπεδα καθιστά τις διαδικασίες εντοπισμού και διόρθωσης σφαλμάτων πολύ πιο εύκολες και απλές σε σχέση με την περίπτωση που γινόταν εστίαση και έλεγχος κατευθείαν στο μεγαλύτερο σύστημα. Για παράδειγμα, ένα πρόβλημα που θα εμφανιζόταν κατά τον έλεγχο των λειτουργιών κλήσης σε ένα νομό, ζώνη ή περιοχή είναι πιθανότερο να είναι αποτέλεσμα του κωδικού που εξυπηρετεί το νομό αυτό παρά πρόβλημα του κωδικού περιοχής ή της κλήσης εντός του τηλεφωνικού κέντρου. Με αυτό τον τρόπο μπορούμε να περιορίσουμε την αναζήτηση για την αιτία του σφάλματος στον κωδικό του υποσυστήματος 3 συν τους κωδικούς των υποσυστημάτων 1 και 2 που επηρεάζονται από το υποσύστημα 3. Εάν αυτό το πρόβλημα το ανακαλύπταμε όταν όλα τα υποσυστήματα είχαν ολοκληρωθεί, δεν θα μπορούσαμε εύκολα να εντοπίσουμε την πιθανή πηγή του σφάλματος αυτού.

Ο αυξητικός έλεγχος απαιτεί προσεκτικό σχεδιασμό. Η ομάδα ελέγχου θα πρέπει να δημιουργήσει ένα **πλάνο κατασκευής** (build plan) ή **πλάνο ολοκλήρωσης** (integration plan) για να καθορίσει τα υποσυστήματα τα οποία θα πρέπει να εξεταστούν και να περιγράψει πώς, πού, πότε, και από ποιόν θα πραγματοποιηθούν οι έλεγχοι αυτοί. Πολλά από τα θέματα που συζητήσαμε στον έλεγχο ολοκλήρωσης πρέπει να αντιμετωπισθούν στο πλάνο κατασκευής, στα οποία συμπεριλαμβάνονται η σειρά ολοκλήρωσης και η ανάγκη για στελέχη ή προγράμματα οδηγούς.

Μερικές φορές ένα επίπεδο ή υποσύστημα ενός πλάνου κατασκευής καλείται **στοιβάδα** (spin). Οι στοιβάδες αριθμούνται, με το χαμηλότερο επίπεδο να καλείται **μηδενική στοιβάδα** (zero spin). Για μεγάλα συστήματα η μηδενική στοιβάδα συχνά αποτελεί ένα ελάχιστο σύστημα· μερικές φορές μπορεί να είναι ακόμα και το λειτουργικό σύστημα του υπολογιστή που φιλοξενεί το σύστημα.

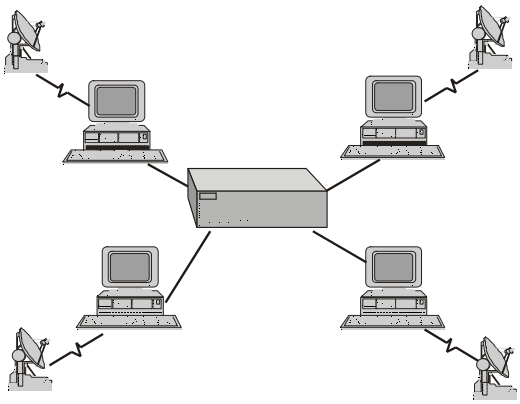
Για παράδειγμα, το πλάνο κατασκευής του τηλεπικοινωνιακού συστήματος μπορεί να περιέχει ένα χρονοδιάγραμμα παρόμοιο με εκείνο του Πίνακα 9.1. Το πλάνο κατασκευής περιγράφει κάθε στοιβάδα με έναν αριθμό, ένα λειτουργικό περιεχόμενο και ένα χρονοδιάγραμμα ελέγχου. Εάν ο έλεγχος της στοιβάδας n είναι επιτυχής και εμφανιστεί ένα πρόβλημα στη στοιβάδα n+1, τότε η πιο πιθανή πηγή του προβλήματος σχετίζεται με τη διαφορά μεταξύ των στοιβάδων n και n+1, δηλαδή με τη λειτουργικότητα που προστίθεται από τη μία στοιβάδα μέχρι να φτάσουμε στην επόμενη. Εάν η διαφορά μεταξύ δύο διαδοχικών στοιβάδων

είναι μικρή, τότε έχουμε σχετικά λίγα μέρη να κοιτάξουμε για τον εντοπισμό της αιτίας του σφάλματος, συνεπώς γίνεται εύκολος και γρήγορος εντοπισμός.

Ο αριθμός των στοιβάδων και ο ορισμός τους εξαρτώνται κυρίως από τους πόρους που διαθέτουμε εμείς καθώς και τους πόρους που διαθέτει ο πελάτης μας. Αυτοί οι πόροι περιλαμβάνουν όχι μόνο το υλικό και το λογισμικό αλλά το διαθέσιμο χρόνο καθώς και το διαθέσιμο προσωπικό. Ένα ελάχιστο σύστημα τοποθετείται στην πιο αρχική στοιβάδα, και οι επόμενες στοιβάδες ορίζονται προσθέτοντας τις επόμενες πιο σημαντικές ή κρίσιμες λειτουργίες όσο το δυνατόν νωρίτερα και φυσικά εφόσον είναι εφικτό. Για παράδειγμα, ας εξετάσουμε το δίκτυο υπολογιστών σε μορφή αστέρα που φαίνεται στην Εικόνα 9.3. Το κέντρο του αστέρα είναι ένας υπολογιστής ο οποίος λαμβάνει μηνύματα από πολλούς μικρότερους υπολογιστές, καθένας από τους οποίους συλλαμβάνει δεδομένα από αισθητήρες και τα μεταδίδει στο κατάλληλο σύστημα για περαιτέρω επεξεργασία. Συνεπώς οι κύριες λειτουργίες του κεντρικού υπολογιστή είναι να μεταφράζει και να κατανοεί τα μηνύματα που στέλνονται από τους υπόλοιπους υπολογιστές. Επειδή αυτές οι λειτουργίες είναι κρίσιμες για το συνολικό σύστημα, θα πρέπει να συμπεριληφθούν σε μία από τις αρχικές στοιβάδες. Πράγματι, μπορούμε να ορίσουμε τις στοιβάδες σύμφωνα με το επόμενο τρόπο:

ΠΙΝΑΚΑΣ 9.1 Το πλάνο κατασκευής για το τηλεπικοινωνιακό σύστημα

Στοιβάδα	Λειτουργίες	Εκκίνηση Ελέγχου	Τερματισμός Ελέγχου
0	Τηλεφωνικό Κέντρο	1 Σεπτεμβρίου	15 Σεπτεμβρίου
1	Κωδικός Περιοχής	30 Σεπτεμβρίου	15 Οκτωβρίου
2	Νομός/ζώνη/περιοχή	25 Οκτωβρίου	5 Νοεμβρίου
3	Χώρα	10 Νοεμβρίου	20 Νοεμβρίου
4	Διεθνείς κλήσεις	1 Δεκεμβρίου	15 Δεκεμβρίου



Εικόνα 9.3 Παράδειγμα δικτύου σε μορφή αστέρα.

- Στοιβάδα 0: έλεγξε τις γενικές λειτουργίες του κεντρικού υπολογιστή
- Στοιβάδα 1: έλεγξε τις λειτουργίες μετάφρασης μηνυμάτων του κεντρικού υπολογιστή
- Στοιβάδα 2: έλεγξε τις λειτουργίες κατανόησης μηνυμάτων του κεντρικού υπολογιστή
- Στοιβάδα 3: έλεγξε κάθε περιφερειακό ηλεκτρονικό υπολογιστή σε κατάσταση αυτόνομης λειτουργίας
- Στοιβάδα 4: έλεγξε τις λειτουργίες μεταβίβασης μηνυμάτων κάθε περιφερειακού υπολογιστή
- Στοιβάδα 5: έλεγξε τις λειτουργίες λήψης μηνυμάτων του κεντρικού υπολογιστή

και τα λοιπά.

Οι ορισμοί των στοιβάδων εξαρτώνται επίσης από την ικανότητα των συστατικών μερών του συστήματος να λειτουργούν με αυτόνομο τρόπο. Μπορεί να είναι πολύ πιο δύσκολο να προσομοιωθεί ένα κομμάτι του συστήματος το οποίο λείπει από το να συμπεριληφθεί μέσα σε μία στοιβάδα, επειδή οι εξαρτήσεις ανάμεσα στα διάφορα συστατικά μερικές φορές απαιτούν τόσο κώδικα προσομοίωσης όσος είναι και ο πραγματικός κώδικας. Θυμηθείτε ότι στόχος μας είναι να ελέγξουμε το σύστημα. Ο χρόνος και η προσπάθεια που πρέπει να καταβληθεί προκειμένου να κατασκευαστούν και να χρησιμοποιηθούν εργαλεία ελέγχου θα ήταν προτιμότερο να δαπανηθούν για τον έλεγχο του πραγματικού συστήματος. Αυτή η ανάγκη εξισορρόπησης είναι παρόμοια με εκείνη που εμπλέκεται στην επιλογή μίας φιλοσοφίας ελέγχου κατά τη διάρκεια του ελέγχου μονάδας και ολοκλήρωσης: Η υλοποίηση πολλών στελεχών και οδηγών μπορεί να απαιτεί τόσο χρόνο κατά τη διάρκεια ελέγχου του προγράμματος όσος θα απαιτούταν για τον έλεγχο των πραγματικών συστατικών στοιχείων των οποίων επιχειρείται η προσομοίωση.

Διαχείριση σχηματισμών

Συχνά ελέγχουμε ένα σύστημα σε στάδια ή κομμάτια, βασιζόμενοι σε στοιβάδες (όπως προηγουμένως) ή σε υποσυστήματα, λειτουργίες, ή άλλου τύπου διασπάσεις που κάνουν τον χειρισμό του έλεγχου πιο εύκολο (εξετάζουμε αυτές τις στρατηγικές ελέγχου σε επόμενη ενότητα του παρόντος κεφαλαίου). Εντούτοις, για τον έλεγχο του συστήματος πρέπει να ληφθούν υπόψη οι πολλαπλές διαφορετικές διαμορφώσεις του συστήματος οι οποίες υλοποιούνται σταδιακά. Ο **σχηματισμός συστήματος** (system configuration) είναι μία συλλογή των συστατικών του συστήματος τα οποία παραδίδονται σε ένα συγκεκριμένο πελάτη. Για παράδειγμα, ένα πακέτο μαθηματικών υπολογισμών μπορεί να πωληθεί με ένα συγκε-

κριμένο σχηματισμό για μηχανές που βασίζονται στο UNIX, με έναν άλλο σχηματισμό για μηχανές που τρέχουν Windows, και ακόμα με άλλο σχηματισμό για συστήματα με Solaris. Για τους σχηματισμούς αυτούς μπορεί να γίνει περαιτέρω διαχωρισμός με βάση το εκάστοτε τσιπ το οποίο περιλαμβάνει ένα σύστημα ή με βάση συγκεκριμένες συσκευές. Συνήθως αναπτύσσουμε έναν πυρήνα λογισμικού που τρέχει στο κάθε διαφορετικό σύστημα και χρησιμοποιούμε τις αρχές που περιγράφονται στα Κεφάλαια 5, 6 και 7* για να απομονώσουμε τις διαφορές ανάμεσα στους διαφορετικούς σχηματισμούς σε ένα μικρό αριθμό ανεξαρτήτων συστατικών. Για παράδειγμα, οι λειτουργίες του πυρήνα μπορεί να περιλαμβάνονται στα συστατικά A, B και C· Τότε ο σχηματισμός 1 θα περιλαμβάνει τα συστατικά A, B, C και D ενώ ο σχηματισμός 2 θα περιλαμβάνει τα συστατικά A, B, C και E.

Η υλοποίηση και ο έλεγχος αυτών των διαφορετικών σχηματισμών απαιτεί τη **διαχείριση σχηματισμών** (configuration management), τον έλεγχο των διαφορών μεταξύ των συστημάτων για την ελαχιστοποίηση των κινδύνων και των σφαλμάτων. Είχαμε δει σε προηγούμενα κεφάλαια τον τρόπο με τον οποίο η ομάδα διαχείρισης σχηματισμών εξασφαλίζει ότι ενδεχόμενες αλλαγές στις απαιτήσεις, στο σχέδιο, ή στον κώδικα αντανακλώνται σωστά στα έγγραφα τεκμηρίωσης και σε άλλα συστατικά που επηρεάζονται από τις αλλαγές αυτές. Κατά τη διάρκεια του ελέγχου, η διαχείριση σχηματισμών είναι μία ιδιαίτερα σημαντική διαδικασία, η οποία συντονίζει τις προσπάθειες ανάμεσα στους ελεγκτές και τους κατασκευαστές.

Εκδοχές και εκδόσεις. Μία διαμόρφωση για ένα συγκεκριμένο σύστημα μερικές φορές καλείται **εκδοχή** (version). Συνεπώς η αρχική διανομή ενός πακέτου λογισμικού μπορεί να αποτελείται από διαφορετικές τέτοιες εκδοχές, μία για κάθε πλατφόρμα ή συσκευή στην οποία θα χρησιμοποιηθεί το λογισμικό αυτό. Για παράδειγμα, μπορεί να κατασκευαστεί λογισμικό αεροσκαφών έτσι ώστε η εκδοχή 1 να τρέχει σε αεροσκάφη του Ναυτικού, η εκδοχή 2 να τρέχει σε αεροσκάφη της πολεμικής αεροπορίας και η εκδοχή 3 να τρέχει σε αεροσκάφη της πολιτικής αεροπορίας.

Καθώς το λογισμικό ελέγχεται και δοκιμάζεται, εντοπίζονται σφάλματα τα οποία πρέπει να διορθωθούν ή να γίνουν μικρές βελτιώσεις στην αρχική λειτουργικότητα του λογισμικού αυτού. Μία νέα **έκδοση** (release) του λογισμικού είναι ένα βελτιωμένο σύστημα το οποίο έχει σκοπό να αντικαταστήσει το παλιό. Συχνά τα συστήματα λογισμικού περιγράφονται με την ονομασία εκδοχή *n*, έκδοση *m* ή ως εκδοχή *n.m*, όπου ο αριθμός αντανακλά τη θέση του συστήματος ανάμεσα στα προηγούμενα συστήματα καθώς το σύστημα αυτό αναπτύσσεται και ωριμάζει. Η

* 1ος Τόμος

εκδοχή *n* συχνά έχει σκοπό να αντικαταστήσει την εκδοχή *n*–1 και η έκδοση *m* αντικαθιστά την έκδοση *m*–1. (Η λέξη "εκδοχή" μπορεί να έχει δύο διαφορετικές έννοιες: μία ως εκδοχή για κάθε τύπο πλατφόρμας ή λειτουργικού συστήματος, ή ως αλυσίδα προϊόντων που αναπτύσσονται σε φάσεις. Η ορολογία αυτή γίνεται συνήθως κατανοητή από τα συμφραζόμενα με τα οποία χρησιμοποιείται. Για παράδειγμα, ένας κατασκευαστής μπορεί να παρέχει την εκδοχή 3 του προϊόντος του για πλατφόρμες UNIX και την εκδοχή 4 για πλατφόρμες Windows, καθεμιά από τις οποίες προσφέρει τις ίδιες λειτουργίες.)

Η ομάδα διαχείρισης σχηματισμών είναι υπεύθυνη να εξασφαλίσει ότι κάθε εκδοχή ή έκδοση είναι σωστή και σταθερή πριν δοθεί στους πελάτες για χρήση, καθώς επίσης να διασφαλίσει ότι οι αλλαγές έγιναν γρήγορα και με ακρίβεια. Η ακρίβεια είναι ένας πολύ κρίσιμος παράγοντας, επειδή θα θέλαμε να αποφύγουμε τη γέννηση νέων σφαλμάτων κατά τη διάρκεια διόρθωσης των παλαιοτέρων. Παρομοίως, η ταχύτητα είναι σημαντική επειδή οι διαδικασίες εντοπισμού και διόρθωσης σφαλμάτων εξελίσσονται παράλληλα με τη διαδικασία εντοπισμού επιπρόσθετων σφαλμάτων από την ομάδα ανάπτυξης του λογισμικού. Συνεπώς, εκείνοι οι οποίοι προσπαθούν να διορθώσουν τα σφάλματα του συστήματος θα πρέπει να δουλέψουν με τα συστατικά και τα έγγραφα τεκμηρίωσης του λογισμικού τα οποία αντανακλούν την τρέχουσα κατάσταση του συστήματος.

Η παρακολούθηση και ο έλεγχος των εκδοχών είναι ιδιαίτερα σημαντικές διαδικασίες όταν εφαρμόζουμε υλοποίηση σε φάσεις. Όπως σημειώσαμε σε προηγούμενα κεφάλαια, ένα **σύστημα παραγωγής** (production system) είναι μία εκδοχή που έχει ελεγχθεί και εκτελεί μόνο ένα υποσύνολο των απαιτήσεων του πελάτη. Η επόμενη εκδοχή, με περισσότερα στοιχεία αναπτύσσεται καθώς οι χρήστες χρησιμοποιούν το σύστημα παραγωγής. Αυτό το **σύστημα ανάπτυξης** (development system) κατασκευάζεται και ελέγχεται· όταν ο έλεγχος ολοκληρωθεί, το σύστημα ανάπτυξης αντικαθιστά το παλιό σύστημα παραγωγής ώστε να γίνει πλέον το νέο σύστημα παραγωγής.

Για παράδειγμα, ας υποθέσουμε ότι μία εγκατάσταση ισχύος αυτοματοποιεί τις λειτουργίες που πραγματοποιούνται στο χώρο ελέγχου. Οι χειριστές της εγκατάστασης ισχύος έχουν εκπαιδευτεί να κάνουν τα πάντα χειρωνακτικά και νιώθουν άβολα αν πρόκειται να δουλέψουν με τον υπολογιστή. Επομένως, αποφασίζουμε να κατασκευάσουμε ένα σύστημα σε φάσεις. Η πρώτη φάση είναι σχεδόν όμοια με το χειροκίνητο σύστημα που γνωρίζει το προσωπικό, αλλά επιτρέπει στους χειριστές της εγκατάστασης ισχύος να δημιουργούν αυτόματα αναφορές για την απόδοση του συστήματος. Στη δεύτερη φάση προστίθενται αρκετές νέες λειτουργίες αυτοματισμού στις λειτουργίες της πρώτης φάσης, αλλά οι μισές από τις λειτουργίες του χώρου ελέγχου συνεχίζουν να γίνονται με χειροκίνητο τρόπο. Κατά τις επόμενες φάσεις συνεχίζεται η αυτοματοποίηση και άλλων επιλεγμένων

λειτουργιών, ενώ κάθε φορά το νέο σύστημα χτίζεται με βάση τις προηγούμενες φάσεις, μέχρι τελικά να αυτοματοποιηθούν όλες οι λειτουργίες του συστήματος. Επεκτείνοντας με αυτό τον τρόπο το αυτοματοποιημένο σύστημα δίνουμε τη δυνατότητα στους χειριστές της εγκατάστασης ισχύος να συνηθίσουν σιγά σιγά και να νιώσουν βολικά με το νέο σύστημα.

Σε κάθε στιγμή της ανάπτυξης του συστήματος σε φάσεις, οι χειριστές της εγκατάστασης ισχύος χρησιμοποιούν ένα σύστημα παραγωγής το οποίο έχει περάσει όλους του ελέγχους. Την ίδια στιγμή, δουλεύουμε στην επόμενη φάση εξέλιξης του συστήματος, ελέγχοντας το σύστημα ανάπτυξης. Όταν το σύστημα ανάπτυξης έχει περάσει από όλους τους προβλεπόμενους ελέγχους και είναι έτοιμο να παραδοθεί στους χειριστές της εγκατάστασης ισχύος, γίνεται πλέον σύστημα παραγωγής (δηλαδή χρησιμοποιείται πλέον από τους χειριστές της εγκατάστασης ισχύος) και συνεχίζουμε με την εξέλιξη του συστήματος στην επόμενη φάση. Καθώς εργαζόμαστε στο σύστημα ανάπτυξης, προσθέτουμε λειτουργίες στο τρέχον σύστημα παραγωγής ή στο σύστημα εργασίας ώστε τελικά να δημιουργήσουμε το νέο σύστημα ανάπτυξης.

Όσο χρόνο ένα σύστημα είναι στη διαδικασία παραγωγής, μπορεί να προκύψουν προβλήματα τα οποία θα πρέπει να αναφερθούν και καταγραφούν. Συνεπώς ένα σύστημα ανάπτυξης συχνά εξυπηρετεί δύο σκοπούς: προσθέτει τη λειτουργικότητα της επόμενης φάσης στο παλιό σύστημα και διορθώνει τα προβλήματα τα οποία εντοπίστηκαν στις προηγούμενες εκδοχές. Επομένως, ένα σύστημα ανάπτυξης μπορεί να περιλαμβάνει τις φάσεις προσθήκης νέων συστατικών μερών καθώς και τις αλλαγές σε υπάρχοντα συστατικά μέρη του λογισμικού. Ωστόσο, αυτή η διαδικασία επιτρέπει να εισαχθούν σφάλματα σε συστατικά τα οποία έχουν παλαιότερα ελεγχθεί. Όταν γράφουμε το πλάνο κατασκευής και τα πλάνα ελέγχου θα πρέπει να λάβουμε υπόψη αυτή την κατάσταση και να εξετάσουμε την ανάγκη για τον έλεγχο αλλαγών οι οποίες υλοποιούνται από τη μία εκδοχή στην επόμενη. Ο συμπληρωματικός έλεγχος μπορεί να διασφαλίσει ότι το σύστημα ανάπτυξης θα λειτουργεί τουλάχιστον όσο καλά λειτουργεί το τρέχον σύστημα παραγωγής. Εντούτοις, πρέπει να φυλάγονται καταγραφές για τις ακριβείς αλλαγές που πραγματοποιούνται στον κώδικα από τη μία εκδοχή στην επόμενη, ώστε να μπορούμε να εντοπίσουμε τις πηγές των προβλημάτων που ενδεχομένως να προκύψουν. Για παράδειγμα, εάν ένας χρήστης αναφέρει ένα πρόβλημα στο σύστημα παραγωγής, θα πρέπει να γνωρίζουμε ποια εκδοχή και έκδοση του κώδικα χρησιμοποιεί. Ο κώδικας μπορεί να διαφέρει σημαντικά από τη μία εκδοχή στην άλλη. Εάν εργαστούμε με λανθασμένη λίστα εντολών, μπορεί να μην εντοπίσουμε ποτέ την αιτία του προβλήματος. Ακόμα χειρότερα, μπορεί να νομίσουμε ότι βρήκαμε την αιτία, και να κάνουμε μία αλλαγή που θα εισάγει ένα νέο σφάλμα χωρίς ταυτόχρονα να διορθωθεί πραγματικά το παλιό σφάλμα!