ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Μεθοδολογίες και Πρότυπα Ανάπτυξης Λογισμικού

(Methodologies and Patterns in Software Development)

Χατζησταύρου Δημήτριος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Φουντούκης Σπυρίδων
ΠΔ 407/80

Λαμία 2009

Μεθοδολογίες και Πρότυπα Ανάπτυξης Λογισμικού

(Methodologies and Patterns in Software Development)

ΠΡΟΛΟΓΟΣ

Για πρώτη φορά ήρθα σε επαφή με το μάθημα της Τεχνολογίας Λογισμικού κατά το πέμπτο εξάμηνο σπουδών μου στο Τμήμα Πληροφορικής με Εφαρμογές στην Βιοϊατρική. Το μάθημα διδασκόταν από τον Δρ. Σπ. Φουντούκη. Μου προξένησε έντονο ενδιαφέρον το αντικείμενο του μαθήματος και του αφιέρωνα, σχετικά με άλλα μαθήματα, τον περισσότερο χρόνο μελέτης. Όταν ο κ. Φουντούκης κατέθεσε τον τίτλο της παρούσας διπλωματικής εργασίας σαν ένα από τα υποψήφια θέματα για εκπόνηση εργασιών, ενθουσιάστηκα. Επεδίωξα να συναντήσω τον διδάσκοντα του μαθήματος και προσπάθησα να του δώσω να καταλάβει το ενδιαφέρον μου για το συγκεκριμμένο θέμα. Έτσι ανέλαβα την εκπόνηση της διπλωματικής εργασίας με τίτλο: *Μεθοδολογίες και Πρότυπα Ανάπτυξης Λογισμικού*. Μαζί με τον κ. Φουντούκη, ο οποίος διαρκώς με ενεθάρρυνε, τολμήσαμε να ‘παραβιάσουμε’ τα όρια της πτυχιακής εργασίας και να εισέλθουμε σε ερευνητικές περιοχές. Επίσης, μαζί αποφασίσαμε να δώσουμε μορφή παροχής γνώσης προχωρημένων θεμάτων Τεχνολογίας Λογισμικού.σε αυτή την εργασία. Ελπίζουμε ότι στο μέλλον θα μπορέσει να φανεί χρήσιμη, ως πηγή μελέτης, σε μεταπτυχιακούς φοιτητές του Πανεπιστημίου μας με ενδιαφέρον στη συγκεκριμμένη επιστημονική περιοχή.

Θα ήθελα σε αυτό εδώ το σημείο, να ευχαριστήσω τον επιβλέποντα διδάσκοντά μου Δρ. Σπ. Φουντούκη για την καθοδήγηση, την υποστήριξη και την εν γένει πολύτιμη βοήθεια που μου προσέφερε. Μου έδειξε έναν δρόμο και έναν τρόπο κατάκτησης γνώσης που πιο πρίν ήσαν άγνωστα σε εμένα. Επίσης επιθυμώ να ευχαριστήσω την οικογένεια μου και τους φίλους μου για την ηθική συμπαράσταση και την κάθε μορφής στήριξη που μου προσέφεραν κατά την διάρκεια όλου αυτού του χρονικού διάστηματος εντατικής εργασίας για την εκπόνηση του παρόντος έργου.

ΔΗΜΟΣΙΕΥΣΕΙΣ

JOURNALS

[4] S. G. Fountoukis and D. T. Chatzistavrou. *Pattern Based Object Oriented Software System Design for Cardiac Diagnosis and Treatment.* International Journal of Caring Sciences, 2009, [Accepted].

CONFERENCES

[3] S. G. Fountoukis and D. T. Chatzistavrou. *Pattern Oriented Design of Cluster Running Object Medical Information Systems.* In Proceedings of American Institute of Physics – AIP, International Conference of Computational Methods in Sciences and Engineering - ICCMSE, September 29 - October 4, 2009, Rhodes, [Accepted].

[2] S. G. Fountoukis. and D. T. Chatzistavrou. *Pattern Based Object Oriented Software Systems Design for High Performance Computing.* In Proceedings of Hercma, September 24 - 26, 2009, Athens, [Accepted].

[1] S. G. Fountoukis. and D. T. Chatzistavrou. *Advanced Design of Distributed Object Oriented Health Care Information Systems*. International Congress in Nursing Education Research and Practice, October 15 - 17, 2009, Thessaloniki, [Accepted].

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία με τίτλο «Μεθοδολογίες και Πρότυπα Ανάπτυξης Λογισμικού» μελετά μεθοδολογίες που προκύπτουν από σχεδιαστικά πρότυπα, οι οποίες μπορούν να εφαρμοστούν κατά την φάση σχεδιασμού του κύκλου ζωής ανάπτυξης λογισμικού. Ο στόχος των προτύπων είναι να βοηθήσει τους αναλυτές και σχεδιαστές μεγάλων έργων λογισμικού να επιλύσουν συχνά εμφανιζόμενα σχεδιαστικά προβλήματα. Τα πρότυπα μπορούν να χωριστούν σε αρχιτεκτονικά και καθαρά σχεδιαστικά. Η εφαρμογή τους μπορεί να επιταχύνει και να διευκολύνει τις διαδικασίες ανάπτυξης λογισμικού με την χρήση δοκιμασμένων και αποδεδειγμένων μεθόδων. Η μεθοδολογία προτύπων σχεδιασμού που μελετάται στην παρούσα πτυχιακή εργασία, συνίσταται στην επαναχρησιμοποίηση δοκιμασμένων αρχιτεκτονικών σχεδιαστικών λύσεων η οποία αποτρέπει την εμφάνιση σημαντικών προβλημάτων στο αναπτυσσόμενο λογισμικό και διευκολύνει την αναγνωσιμότητα, την συντήρηση και την ενημέρωση των κωδίκων του.

Τα σχεδιαστικά πρότυπα μπορούν να ταξινομηθούν ως προς την σχεδιαστική περιοχή στην οποία δίνουν λύσεις. Τα με αυτόν τον τρόπο κατηγοριοποιούμενα βασικώτερα πρότυπα είναι:

*Θεμελιώδη πρότυπα***:** Διεπαφή, Αφηρημένη Υπερκλάση, Διεπαφή και Αφηρημένη κλάση, Αμετάβλητο, Σήμανση διεπαφής, Πληρεξούσιο.

*Δημιουργικά πρότυπα***:** Μεθόδου Εργοστασίου, Αφηρημένου Εργοστασίου, Οικοδόμησης, Πρωτότυπο, Singleton, Αποθέματος Αντικειμένων.

*Πρότυπα Διαμερισμού***:** Φίλτρου, Διεπαφή μόνο ανάγνωσης.

*Δομικά πρότυπα***:** Μετασχηματιστή, Γέφυρας, Πρόσοψης, Flyweight, Δυναμικού Συνδέσμου, Εικονικού Πληρεξουσίου, Διακοσμητή, Σύνθεσης.

*Πρότυπα Συμπεριφοράς***:** Εντολής, Διερμηνέα, Ενθύμιου, Μεσολαβητή, Iterator, Ειδίκευσης, Παρατηρητή, Κατάστασης, Στρατηγικής, Μεθόδου Template, Επισκέπτη, Αλυσίδας Ευθύνης.

*Πρότυπα Ταυτοχρονισμού***:** Ενεργό Αντικείμενο, Διπλοελεγμένου κλειδώματος, Εμποδισμού, Φρουρούμενη αναβολή, Ηγετών/ακολούθων, Οργάνου Ελέγχου, Ανάγνωσης-γραφής κλειδαριάς, Χρονοπρογραμματιστή, Αποθέματος Νημάτων, Αποθήκευσης Συγκεκριμένου-Νήματος, Αντιδραστήρα.

Αρχιτεκτονικά Πρότυπα: Παρουσίαση-Αφαίρεση–Έλεγχος, Σωλήνωση, Υπονοούμενη επίκληση, Πρότυπο Πίνακα, Peer to peer, Προσανατολισμένη προς τις υπηρεσίες αρχιτεκτονική, Γυμνά Αντικείμενα, Μοντέλο-Όψη-Ελεγκτής.

Παρά την ύπαρξη πληθώρας προτύπων, οι σχεδιαστικές ανάγκες δεν μπορούν να καλυφθούν και συχνά απαιτούνται νέες σχεδιαστικές λύσεις για προβλήματα που ανακύπτουν. Πολλές τέτοιες λύσεις μπορούν να προκύψουν από τους συνδιασμούς των γνωστών προτύπων ιδίως εάν τα προβλήματα είναι σύνθετα.

Στην παρούσα πτυχιακή εργασία, αναζητούνται λύσεις για σύνθετα σχεδιαστικά προβλήματα λογισμικού υψηλών επιδόσεων και στα πλαίσια αυτά αναπτύσσονται ορισμένοι συνδιασμοί προτύπων.

Επιστέγασμα της παρούσας πτυχιακής εργασιάς αποτελεί, η εφαρμογή της μελετηθείσας μεθοδολογίας προτύπων για το σχεδιασμός προτύπου Ιατρικού λογισμικού συστήματος που αφορά την διάγνωση και την θεραπεία προβλημάτων καρδιάς.

ABSTRACT

The present work titled ‘Methodologies and Patterns in Software Development’ studies pattern based methodologies for software systems design. The patterns are defined as tried and proved solutions to repeatedly occurring design problems. The use of patterns at the design phase of the software development life cycle aims to solve design problems in a consistent way avoiding serious structural software errors.

The application of design patterns deters the appearance of important problems in the under development software and facilitates the readability and the maintenance of its code.

The patterns can be classified as architectural and (pure) design. The thus categorized most important patterns are:

Fundamental patterns: Interface, Abstract Hyper Class, Interface and Abstract Class, Immutable, Marker Interface, Proxy.

Creational patterns: Factory method, Abstract Factory, Builder, Prototype, Singleton, Object Pool.

Partitioning patterns: Filter, Read only Interface.

Structural patterns: Adapter, Bridge, Facade, Flyweight, Dynamic Linkage, Virtual Proxy, Decorator, Composite.

Behavioral patterns: Command, Interpreter, Memento, Mediator, Iterator, Specification, Observer, State, Strategy, Template Method, Visitor, Chain of Responsibility.

Concurrency patterns: Active Object, Double Checked Locking, Balking, Guarded suspension, Leaders/Followers, Monitor Object, Read Write Lock, Scheduler, Thread Pool, Thread - Specific Storage, Reactor.

Architectural patterns: Presentation - Abstraction - Control, Pipeline, Implicit invocation, Blackboard, Peer to Peer, Service- Oriented Architecture, Naked Objects, Model-View-Controller.

In spite of the considerable number of the known patterns, the increased software design needs cannot be fulfilled. A solution to complex design problems is the innovation of new patterns based on combinations. In the present work, pattern combination based solutions are studied for high performance computing software systems design problems. Herein also, a pattern based design of a novel medical software system for cardiac diagnosis and treatment is studied.

1. ΠΡΟΤΥΠΑ ΚΑΙ ΛΟΓΙΣΜΙΚΟ

1.1 Εισαγωγή στα πρότυπα

Τα πρότυπα για την ανάπτυξη λογισμικού είναι από τα πιο πρόσφατα «καυτά θέματα» που προέκυψαν από την αντικειμενοστρεφή κοινότητα. Είναι μια λογοτεχνική μορφή επίλυσης προβλημάτων της τεχνολογίας λογισμικού, που έχει τις ρίζες της σε μια σχεδιαστική κίνηση με το ίδιο όνομα στη σύγχρονη αρχιτεκτονική, τον λογοτεχνικό προγραμματισμό (literature programming), και την τεκμηρίωση των καλύτερων πρακτικών που έχουν γίνει γνωστές.

Θεμελιώδες σε οποιαδήποτε επιστήμη ή εφαρμοσμένη μηχανική ή πειθαρχία είναι ένα κοινό λεξιλόγιο για την έκφραση των εννοιών της και μια γλώσσα για να τους συσχετίσει. Ο στόχος των προτύπων μέσα στη κοινότητα λογισμικού είναι να δημιουργηθεί ένας όγκος της βιβλιογραφίας για να βοηθήσει τους προγραμματιστές λογισμικού να επιλύσουν τα επαναλαμβανόμενα προβλήματα που αντιμετωπίζονται σε όλη την ανάπτυξη λογισμικού. Έτσι δημιουργείται μια κοινή γλώσσα για την επικοινωνία και δοκιμάζει αυτά τα προβλήματα και τις λύσεις τους. Κωδικοποιώντας αυτές τις λύσεις και τις σχέσεις τους, γίνεται εύκολα κατανοητή η γνώση που καθορίζει την κατανόηση των καλών αρχιτεκτονικών, που ικανοποιούν τις ανάγκες των χρηστών. Η διαμόρφωση μιας κοινής γλώσσας προτύπων για τη απόδοση των δομών και των μηχανισμών των αρχιτεκτονικών, επιτρέπει να γίνονται τα πρότυπα κατανοητά. Η αρχική εστίαση είναι στη δημιουργία μιας νοοτροπίας για να τεκμηριωθεί και να υποστηριχθεί η υγιής μηχανική αρχιτεκτονική και ο σχεδιασμός.

1.2 Προέλευση προτύπων

Τα πρότυπα λογισμικού έγιναν αρχικά δημοφιλή με την ευρεία αποδοχή του βιβλίου: Design Patterns: Elements of Reusable Object-Oriented Software από τους Erich Gamma, Richard Helm, Ralph Johnson, και John Vlissides (συχνά αποκαλούμενη η συμμορία τεσσάρων ή GoF) [1]. Τα πρότυπα έχουν χρησιμοποιηθεί για πολλά διαφορετικά πεδία που κυμαίνονται από τις οργανώσεις και τις διαδικασίες ως τη διδασκαλία και την αρχιτεκτονική. Αυτή τη στιγμή, η κοινότητα λογισμικού χρησιμοποιεί τα πρότυπα κατά ένα μεγάλο μέρος για την κατασκευή λογισμικού και το σχεδιασμό, και πιο πρόσφατα, για τις διαδικασίες ανάπτυξης λογισμικού και τις οργανώσεις. Άλλα πρόσφατα βιβλία που έχουν βοηθήσει στη διάδοση των προτύπων είναι: Pattern-Oriented Software Architecture: A System of Patterns από τους Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, και Michael Stal (μερικές φορές αποκαλούμενοι συμμορία Siemens των πέντε ή GoV) [2] και τα βιβλία Pattern Languages of Program Design και Pattern Languages of Program Design 2 [3] , τα οποία είναι επιλεγμένα έγγραφα από το πρώτο και δεύτερο συνέδριο σχετικά με τις γλώσσες προτύπων σχεδιασμού προγράμματος (PLoP ή PLoPD). Πολλά από αυτά τα βιβλία είναι μέρος από το: Software Patterns Series από τους Addison-Wesley.

Η τωρινή χρήση του όρου «πρότυπο» προέρχεται από τον αρχιτέκτονα Christopher Alexander που έχει γράψει διάφορα βιβλία με θέματα που αφορούν την αστικό προγραμματισμό και αρχιτεκτονική οικοδόμησης:

- Notes on the Synthesis of Form, Harvard University Press, 1964 [4] (αναφέρεται και ως "*[Notes]*")
- The Oregon Experiment, Oxford University Press, 1975 [5] (αναφέρεται και ως "*[Oregon]*")
- A Pattern Language: Towns, Buildings, Construction, Oxford University Press, 1977 [6] (αναφέρεται και ως "*[APL]*")
- The Timeless Way of Building, Oxford University Press, 1979 [7] (αναφέρεται και ως "*[TTWoB]*")

Αν και αυτά τα βιβλία υπάρχουν φαινομενικά για την αρχιτεκτονική και τον αστικό σχεδιασμό, ισχύουν σε πολλές άλλες περιπτώσεις, συμπεριλαμβανομένης της ανάπτυξης λογισμικού. Στο 'Notes on the Synthesis of Form' [4] ο Alexander υποστηρίζει ότι οι τρέχουσες μέθοδοι αρχιτεκτονικής, οδηγούν σε αποτελέσματα που αποτυγχάνουν να ικανοποιήσουν τις πραγματικές ανάγκες και απαιτήσεις των χρηστών του και της κοινωνίας και είναι ανεπιτυχείς στην πραγματοποίηση του σκοπού των σχεδιαστικών και μηχανικών προσπαθειών. Ο Alexander θέλησε να δημιουργήσει τις δομές που είναι κατάλληλες για τους ανθρώπους και έχουν θετική επιρροή σε αυτούς, βελτιώνοντας την άνεσή τους και την ποιότητα ζωής τους. Κατέληξε στο συμπέρασμα ότι οι αρχιτέκτονες πρέπει συνεχώς να προσπαθούν να παράγουν προϊόντα που προσαρμόζονται καλύτερα στις ανάγκες όλων των κατοίκων και των χρηστών τους και των αντίστοιχων κοινοτήτων τους. Στο 'The Timeless Way of Building' [10] ο Alexander προτείνει ένα παράδειγμα για την αρχιτεκτονική βασισμένη σε τρεις έννοιες: την ποιότητα, την πύλη, και τον τρόπο.

Η ποιότητα

Αυτή είναι η ουσία όλων των πραγμάτων που ζουν και δίνει σε αυτά ιδιότητες όπως: ελευθερία, πληρότητα, ολοκλήρωση, άνεση, αρμονία, διάρκεια, ειλικρίνεια, ανθεκτικότητα, μεταβλητότητα, και προσαρμοστικότητα. Είναι αυτό που μας κάνει να αισθανόμαστε ζωντανοί, μας δίνει ικανοποίηση και βελτιώνει τελικά τον άνθρωπο.

Η πύλη

Είναι ο μηχανισμός που μας επιτρέπει να φθάσουμε στην ποιότητα. Φανερώνεται ως ζωντανή γλώσσα προτύπων, που επιτρέπει να δημιουργηθούν πολύμορφα πρότυπα που εκπληρώνουν τις πολύπλευρες ανάγκες. Είναι ο καθολικός «αιθέρας» των προτύπων και των σχέσεών τους που υπάρχουν σε μια δεδομένη περιοχή. Η πύλη είναι ο αγωγός στην ποιότητα.

Ο τρόπος

Χρησιμοποιώντας τον τρόπο, τα πρότυπα από την πύλη εφαρμόζονται χρησιμοποιώντας τη τεχνική του διαφοροποιημένου διαστήματος (differentiating space) σε μια διατεταγμένη ακολουθία αποσπασματικής αύξησης. Ο Alexander το παρομοιάζει με «μια διαδικασία, όπως την εξέλιξη ενός εμβρύου, στο οποίο το σύνολο προηγείται των κομματιών του, και τα γεννά, με διαχωρισμό.» Με αυτόν τον τρόπο, κάποιος μπορεί να περάσει μέσω της πύλης για να φθάσει στην ποιότητα.

1.3 Ένα κομμάτι της ιστορίας προτύπων

Το 1987, οι Ward Cunningham and Kent Beck αποφάσισαν να χρησιμοποιήσουν μερικές από τις ιδέες του Alexander να αναπτύξουν μια μικρή γλώσσα πέντε προτύπων για την καθοδήγηση των αρχαρίων προγραμματιστών. Ετοίμασαν τα αποτελέσματα και τα παρουσίασαν στο Ορλάντο το "Using Pattern Languages for Object-Oriented Programs"[8] [9].

Αργότερα, ο James Coplien άρχισε να δημιουργεί έναν κατάλογο με ιδιωματισμούς στην C++ (που είναι ένα είδος προτύπου) και τους δημοσίευσε αργότερα ως βιβλίο το 1991, «Advanced C++ Programming Styles and Idioms» [11].

Από το 1990 ως το 1992, διάφορα μέλη της συμμορίας των τεσσάρων είχαν συναντηθεί μεταξύ τους και συνέταξαν έναν κατάλογο με πρότυπα. Το 1991 σε ένα εργαστήριο που έγινε από τον Bruce Andersen και επαναλήφθηκε το 1992 έγιναν πολλές συζητήσεις για τα πρότυπα. Πολλοί διαμορφωτές προτύπων συμμετείχαν σε αυτά τα εργαστήρια, συμπεριλαμβανομένου των James Coplien, Doug Lea, Desmond D'Souza, Norm Kerth, Wolfgang Pree και άλλων.

Τον Αύγουστο του 1993, ο Kent Beck και ο Grady Booch ήταν σπόνσορες στο Κολοράντο, την πρώτη συνεδρίαση που είναι τώρα γνωστή ως Hillside Group. Ένα άλλο εργαστήριο προτύπων διοργανώθηκε, το OOPSLA'93 [12] και έπειτα τον Απρίλιο του 1994, το Hillside Group συναντήθηκε πάλι, αυτή τη φορά και με τον Richard Gabriel, για να προγραμματίσει την πρώτη διάσκεψη PLoP [13].

Σύντομα έκτοτε, το βιβλίο Design Patterns GoF δημοσιεύθηκε και τα υπόλοιπα, είναι ιστορία.

Τι είναι ένα πρότυπο;

Στο «Understanding and Using Patterns in Software Development» [14], ο Dirk Riehle και ο Heinz Zullighoven δίνουν έναν συμπαθητικό ορισμό του όρου «πρότυπο» που ισχύει ευρέως: Ένα πρότυπο είναι η αφαίρεση από μια συγκεκριμένη μορφή που επαναλαμβάνεται σε συγκεκριμένα μη αυθαίρετα πλαίσια.

Οι ανωτέρω συντάκτες επισημαίνουν ότι, εντός της κοινότητας προτύπων λογισμικού, η έννοια ενός προτύπου «συνδέεται με την επίλυση των προβλημάτων στο σχεδιασμό.» Πιο συγκεκριμένα, η συγκεκριμένη μορφή που επαναλαμβάνεται είναι αυτή μιας λύσης σε ένα επαναλαμβανόμενο πρόβλημα. Αλλά ένα πρότυπο είναι περισσότερο από ακριβώς μια αποδεδειγμένη λύση σε ένα επαναλαμβανόμενο πρόβλημα. Η προτεινόμενη λύση περιλαμβάνει κάποιο είδος δομής που ισορροπεί αυτές τις ανησυχίες, ή «τις δυνάμεις», με τον πιο κατάλληλο τρόπο για το δεδομένο πλαίσιο. Χρησιμοποιώντας τη μορφή προτύπων, η περιγραφή της λύσης προσπαθεί να συλλάβει την ουσιαστική διορατικότητα που ενσωματώνει, έτσι ώστε άλλοι μπορούν να μάθουν από αυτήν, και να την χρησιμοποιήσουν σε παρόμοιες καταστάσεις. Στο πρότυπο δίνεται επίσης ένα όνομα, που χρησιμεύει ως μια εννοιολογική λαβή, για να διευκολύνει τη συζήτηση του προτύπου και τις πληροφορίες που αντιπροσωπεύει. Έτσι ένας ορισμός που απεικονίζει περισσότερο τη χρήση του εντός της κοινότητας προτύπων είναι:

Ένα πρότυπο είναι ένα ονομασμένο ψήγμα των διδακτικών πληροφοριών που συλλαμβάνουν την ουσιαστική δομή και την ομάδα αποδεδειγμένων λύσεων σε ένα επαναλαμβανόμενο πρόβλημα που προκύπτει μέσα σε ορισμένα πλαίσιο και ένα σύστημα δυνάμεων.
Ένας ελαφρώς συμπαγέστερος ορισμός που να είναι ευκολότερος να απομνημονευτεί είναι:
Ένα πρότυπο είναι ένα ονομασμένο ψήγμα της διορατικότητας που μεταβιβάζει την ουσία μιας αποδεδειγμένης λύσης σε ένα επαναλαμβανόμενο πρόβλημα μέσα σε ένα ορισμένο πλαίσιο μέσα στις ανταγωνιστικές ανησυχίες.
Τα πρότυπα ενδιαφέρονται συνήθως για κάποιο είδος αρχιτεκτονικής ή οργάνωσης των ιδρυτικών μερών για να παράγουν ένα μεγαλύτερο σύνολο. Ο Richard Gabriel, συντάκτης του «Patterns of Software: Tales From the Software Community» παρέχει έναν σαφή και συνοπτικό ορισμό του όρου προτύπου στο τμήμα ορισμών προτύπων της αρχικής σελίδας των προτύπων:
Κάθε πρότυπο είναι ένας τριμερής κανόνας, που εκφράζει μια σχέση μεταξύ ενός ορισμένου πλαισίου, ένα ορισμένο σύστημα δυνάμεων που εμφανίζεται επανειλημμένα σε εκείνο το πλαίσιο, και μια ορισμένη διαμόρφωση λογισμικού που επιτρέπει σε αυτές τις δυνάμεις για να επιλυθεί [15].

Όπως ο Gabriel εξηγεί, ο Alexander τον περιγράφει λίγο πιο πολύχρωμα στο ‘The Timeless Way of Building’: Κάθε πρότυπο είναι ένας τριμερής κανόνας, ο οποίος εκφράζει μια σχέση μεταξύ ενός ορισμένου πλαισίου, ενός προβλήματος και μιας λύσης.

Όπως ένα στοιχείο στον κόσμο, κάθε πρότυπο είναι μια σχέση μεταξύ ενός ορισμένου πλαισίου, ενός ορισμένου συστήματος δυνάμεων που εμφανίζονται επανειλημμένα σε εκείνο το πλαίσιο και μιας ορισμένης χωρικής διαμόρφωσης που επιτρέπει σε αυτές τις δυνάμεις να επιλυθεί.
Σαν στοιχείο της γλώσσας, ένα πρότυπο είναι μια οδηγία, η οποία επιδεικνύει πώς αυτή η χωρική διαμόρφωση μπορεί να χρησιμοποιηθεί, επανειλημμένως, για να επιλύσει το δεδομένο σύστημα των δυνάμεων, οπουδήποτε το περιεχόμενο το καθιστά σχετικό.
Στο ‘‘Πρότυπα Λογισμικού’’ [16] Ο Jim Coplien γράφει:
« Θα μπορούσα να σας πω πώς να κάνω ένα φόρεμα με τη διευκρίνιση της διαδρομής ενός ψαλιδιού μέσω ενός κομματιού υφάσματος από την άποψη των γωνιών και των μηκών της περικοπής. Ή, θα μπορούσα να σας δώσω ένα πρότυπο. Διαβάζοντας τον προσδιορισμό, δεν θα είχατε καμία ιδέα τι χτιζόταν ή εάν είχατε χτίσει το σωστό πράγμα όταν το τελειώνατε. Το πρότυπο προαναγγέλλει το προϊόν: είναι ο κανόνας για το πράγμα, αλλά είναι επίσης, το ίδιο το πράγμα».
Έτσι γίνεται κατανοητό ότι ένα πρότυπο περιλαμβάνει μια γενική περιγραφή μιας επαναλαμβανόμενης λύσης σε ένα επαναλαμβανόμενο πρόβλημα, κορεσμένο με τους διάφορους στόχους και περιορισμούς. Ένα πρότυπο όχι μόνο προσδιορίζει μια λύση, αλλά εξηγεί γιατί η λύση απαιτείται.

1.4 Παραγωγικά πρότυπα - Generative

Στο ‘‘The Timeless Way of Building’’ [10] ο Alexander εξηγεί ότι τα πιο χρήσιμα πρότυπα είναι τα παραγωγικά. Αυτά τα πρότυπα στο μυαλό μας είναι διανοητικές εικόνες των προτύπων στον κόσμο. Είναι αφηρημένες αναπαραστάσεις των μορφολογικών κανόνων που καθορίζουν τα πρότυπα στον κόσμο. Από μια άποψη

είναι πολύ διαφορετικά. Τα πρότυπα στον κόσμο μόνο υπάρχουν. Αλλά τα ίδια πρότυπα στο μυαλό είναι δυναμικά. Έχουν δύναμη και είναι παραγωγικά. Προσδιορίζουν τι πρέπει να γίνει, λένε πώς μπορούν να παραχθούν και λένε επίσης, ότι υπό ορισμένες συνθήκες πρέπει να δημιουργηθούν. Κάθε πρότυπο είναι ένας κανόνας που περιγράφει τι πρέπει να γίνει για να παραχθεί η οντότητα που καθορίζει.

Τα παραγωγικά πρότυπα ορίζουν πώς θα δημιουργηθεί κάτι και πως μπορεί να παρατηρηθεί στις προκύπτουσες αρχιτεκτονικές συστημάτων που βοήθησαν να διαμορφώσουν. Τα μη-παραγωγικά πρότυπα περιγράφουν τα επαναλαμβανόμενα φαινόμενα χωρίς απαραίτητα να πουν πώς θα αναπαραχθούν. Πρέπει να τεκμηριώνονται τα παραγωγικά πρότυπα, όχι μόνο επειδή εμφανίζουν τα χαρακτηριστικά των καλών συστημάτων, αλλά διδάσκουν πώς θα χτιστούν.

Ο Alexander θέλει τα πρότυπα και ειδικά οι γλώσσες προτύπων να είναι ικανές να δημιουργήσουν ολόκληρες δομές διαβίωσης. Μέρος της επιθυμίας του να δημιουργηθούν οι αρχιτεκτονικές που εξομοιώνουν τη ζωή, βρίσκεται στην μοναδική δυνατότητα των πραγμάτων διαβίωσης να εξελίσσονται και να προσαρμόζονται στα συνεχώς μεταβαλλόμενα περιβάλλοντά τους. Ο Alexander θέλει να μεταδώσει αυτές τις ίδιες ιδιότητες στις αρχιτεκτονικές του. Ομοίως, στο λογισμικό, οι καλές αρχιτεκτονικές λογισμικού είναι όλες προσαρμόσιμες και μπορούν να αλλάξουν. Έτσι μια άλλη πτυχή είναι η προσπάθεια να δημιουργηθούν αρχιτεκτονικές που είναι σε θέση δυναμικά για να εκπληρώσουν τις μεταβαλλόμενες ανάγκες και τις απαιτήσεις.

Υπάρχει μεγάλη παραγωγικότητα. Η διαδοχική εφαρμογή διάφορων προτύπων, ξετυλίγει μια μεγαλύτερη λύση που προκύπτει έμμεσα ως αποτέλεσμα των μικρότερων λύσεων. Έτσι γίνεται κατανοητό τι σημαίνει παραγωγικότητα. Με αυτό τον τρόπο, τα πρότυπα και οι γλώσσες προτύπων πρέπει να καθοδηγήσουν τους χρήστες τους για να παράγουν ολόκληρες τις αρχιτεκτονικές που έχουν ποιότητα.

Ο φθόνος-ζήλια και η ηθική προτύπων

Είναι αναπόφευκτο, ότι η μεγάλη άνοδος στη δημοτικότητα των προτύπων λογισμικού έχει οδηγήσει σε ογκώδη διαφημιστική εκστρατεία. Πολλοί χρησιμοποιούν τη λέξη "pattern" ως μια νέα “hot” λέξη. Τέτοια "διαφημιστική εκστρατεία προτύπων " τελικά προκαλεί την απογοήτευση, τη δυσαρέσκεια και ακόμη και την περιφρόνηση όταν η διαφημιστική εκστρατεία αποδεικνύεται διαφορετική από την πραγματικότητα. Αυτό βλάπτει την αξιοπιστία και τη νομιμότητα εκείνων στην κοινότητα προτύπων που καταβάλλουν γνήσιες προσπάθειες να τεκμηριώσουν γνήσια πρότυπα. Επομένως υπάρχει μια ισχυρή ηθική εντός αυτής της κοινότητας για να αποφύγει τη διαφημιστική εκστρατεία για τα πρότυπα και την σχετική με τα πρότυπα τους εργασία.

1.5 Προτο-πρότυπα και Patternity tests

Είναι σημαντικό, ότι κάθε λύση ή αλγόριθμος, δεν είναι η καλύτερη πρακτική ή αξίωμα από τα οποία αποτελείται ένα πρότυπο. Ακόμα κι αν κάτι εμφανίζεται να έχει όλα τα απαραίτητα στοιχεία προτύπων, δεν πρέπει να θεωρηθεί πρότυπο έως ότου έχει ελεγχθεί για να είναι ένα επαναλαμβανόμενο φαινόμενο (κατά προτίμηση σε τουλάχιστον τρία υπάρχοντα συστήματα - αυτό καλείται συχνά κανόνας τριών).

Μερικοί πιστεύουν ότι δεν είναι σωστό να ονομάζεται κάτι ως πρότυπο, έως ότου έχει υποβληθεί σε κάποιο βαθμό διερεύνησης.

Εάν όπως ο Alexander γράφει, ένα πρότυπο πρέπει να είναι και διαδικασία και ένα πράγμα, τότε ένα πρότυπο πρέπει να περιγράψει όχι μόνο τη διαδικασία που δημιουργεί εκείνο το πράγμα, αλλά και το πράγμα που δημιουργείται με εκείνη την διαδικασία. Για αυτόν τον λόγο, πολλοί υποστηρίζουν ότι τα πρότυπα εξετάζουν τελικά την κάποια οπτικά ευδιάκριτη δομή. Πρέπει να μπορεί να δημιουργηθεί μια εικόνα του είδους- δομής που προκύπτει από τη χρησιμοποίηση του πρότυπου στην πράξη.

Ένα "πρότυπο στην αναμονή" που δεν είναι ακόμα γνωστό να περάσει τις προαναφερθείσες patternity δοκιμές, ή μερικές από αυτές που θα αναφερθούν, καλείται συχνά προτο-πρότυπο. Οι συνοπτικές περιγραφές τέτοιων προτο-προτύπων καλούνται μερικές φορές patlets .

Τεκμηριώνοντας τα καλά πρότυπα μπορεί να είναι μια εξαιρετικά δύσκολη εργασία. Τα καλά πρότυπα κάνουν τα εξής:

1). Λύνουν ένα πρόβλημα. Τα πρότυπα συλλαμβάνουν τις λύσεις, όχι μόνο αφηρημένες αρχές ή στρατηγικές.
2). Είναι μια αποδεδειγμένη έννοια. Τα πρότυπα συλλαμβάνουν τις λύσεις με ένα αρχείο διαδρομής, όχι θεωρίες ή υποθέσεις.
3). Η λύση δεν είναι προφανής. Πολλές τεχνικές επίλυσης προβλημάτων, όπως τα παραδείγματα ή οι μέθοδοι σχεδιασμού λογισμικού, προσπαθούν να αντλήσουν τις λύσεις από τα πρώτα σχέδια. Τα καλύτερα πρότυπα παράγουν μια λύση σε ένα πρόβλημα έμμεσα, που είναι μια απαραίτητη προσέγγιση για τα δυσκολότερα προβλήματα σχεδιασμού.
4). Περιγράφει μια σχέση. Τα πρότυπα δεν περιγράφουν ακριβώς τις ενότητες, αλλά περιγράφουν τις βαθύτερους δομές και τους μηχανισμούς συστημάτων.
5). Το πρότυπο έχει ένα σημαντικό ανθρώπινο συστατικό. Όλο το λογισμικό εξυπηρετεί την ανθρώπινη άνεση ή την ποιότητα της ζωής και τα καλύτερα πρότυπα απευθύνονται ρητά στην αισθητική και τη χρησιμότητα.

Κάτι που δεν είναι πρότυπο, δεν σημαίνει ότι δεν είναι καλό πάντα. Εάν κάτι είναι ένα πρότυπο, ενδεχομένως να είναι καλό. Υπάρχουν πολλά χρήσιμα πράγματα στον κόσμο εκτός από τα πρότυπα. Δεν πρέπει να επηρεάζει η διαφημιστική εκστρατεία των προτύπων και να θεωρούνται πάντα τα πρότυπα ότι είναι καλά.

Σαφώς, η ίδια η λέξη "pattern" προτείνει την επανάληψη. Εάν κάτι δεν επαναλαμβάνεται, δεν μπορεί ενδεχομένως είναι ένα πρότυπο. Αλλά η επανάληψη δεν είναι το μόνο χαρακτηριστικό. Πρέπει επίσης να δειχτεί, ότι ένα πρότυπο είναι κατάλληλο για χρήση, και ότι ταιριάζει στην συγκεκριμένη χρήση. Η επανάληψη είναι καθαρά ποσοτικό χαρακτηριστικό, η ικανότητα και η χρησιμότητα είναι ποιοτικά χαρακτηριστικά. Μπορούμε να εμφανίσουμε επανάληψη απλά με τη χρησιμοποίηση του κανόνα τριών. Χρειάζεται εξήγηση πώς το πρότυπο είναι επιτυχές, γιατί είναι επιτυχές και ευεργετικό.

1.6 Τα πρότυπα είναι χρήσιμα, χρησιμοποιήσιμα και χρησιμοποιημένα

Εκτός από την επανάληψη, ένα πρότυπο πρέπει να περιγράψει πώς η λύση ισορροπεί ή χρησιμοποιεί τις δυνάμεις της. Έτσι το πρότυπο δεν είναι ούτε καθαρή υπόθεση (καθαρή θεωρία), ούτε είναι το πρότυπο που ακολουθεί τυφλά άλλα (rote πρακτική). Θέλουμε να δείξουμε ότι η πρακτική είναι περισσότερο από ακριβώς «θεωρία», και ότι η θεωρία πραγματικά έχει ασκηθεί.
Ένα πρότυπο είναι όπου η θεωρία και η πρακτική συναντιούνται για να ενισχυθούν και να συμπληρώσουν η μια την άλλη, δείχνοντας ότι η δομή που περιγράφει είναι χρήσιμη, χρησιμοποιήσιμη, και χρησιμοποιημένη.
Ένα πρότυπο πρέπει να είναι χρήσιμο, επειδή αυτό επιδεικνύει πώς έχοντας το πρότυπο στο μυαλό, μπορεί να μετασχηματιστεί σε μια περίπτωση του προτύπου στο πραγματικό κόσμο, ως ένα πράγμα που προσθέτει αξία στη ζωή.
Ένα πρότυπο πρέπει επίσης να είναι χρησιμοποιήσιμο, επειδή αυτό επιδεικνύει πώς ένα πρότυπο που περιγράφεται με λογοτεχνική μορφή, μπορεί να μετασχηματιστεί σε ένα σχέδιο που υπάρχει στο μυαλό.
Ένα πρότυπο πρέπει να χρησιμοποιηθεί, επειδή έτσι τα πρότυπα που υπάρχουν στο πραγματικό κόσμο τεκμηριώθηκαν αρχικά ως πρότυπα σε λογοτεχνική μορφή.

Αυτό παράγει έναν κύκλο συνεχούς επανάληψης από τους συγγραφείς προτύπων, στους αναγνώστες προτύπων, στους χρήστες προτύπων. Οι συγγραφείς που τεκμηριώνουν τα πρότυπα με λογοτεχνική μορφή, τα καθιστούν χρησιμοποιήσιμα στους αναγνώστες προτύπων, που μπορούν έπειτα να τα θυμηθούν, κάτι που τα καθιστά χρήσιμα στους επαγγελματίες και τους υπεύθυνους για την ανάπτυξη, που μπορούν να τα χρησιμοποιήσουν στο πραγματικό κόσμο και να ενισχύσουν τη ποιότητα ζωής του χρήστη.

1.7 Αντί-πρότυπα

Εάν ένα πρότυπο αναπαριστά μια «καλύτερη πρακτική», τότε ένα αντι-πρότυπο αναπαριστά μια «εμπειρία» Ο Andrew Koenig στην έκθεση C++ Νοεμβρίου 1995, πρότεινε δύο έννοιες «των αντί-προτύπων.»
1). Εκείνα που περιγράφουν μια κακή λύση σε ένα πρόβλημα που οδήγησε σε μια κακή κατάσταση.
2). Εκείνα που περιγράφουν πώς να βγουν από μια κακή κατάσταση και πώς να προχωρήσουν από εκεί σε μια καλή λύση.
Ένας άλλος ορισμός για τα αντί- πρότυπα είναι: Ένα αντί-πρότυπο αποτελεί έναν μηχανισμό με τον οποίο περιγράφεται μία κοινότυπη λύση για τις ανάγκες της ανάπτυξης μιας εφαρμογής, η οποία όμως λύση, αναπαράγει σημαντικά αρνητικές συνέπειες. Ένα αντί-πρότυπο εξετάζει τις αιτίες, τα συμπτώματα και τις συνέπειες από την εφαρμογή μιας κακής λύσης και προτείνει μια νέα λύση η οποία θα οδηγήσει σε ασφαλή αποτελέσματα.
Ο Alexander λέει στο ‘Notes on the Synthesis of Form’ για ‘δυνάμεις’ και ‘misfits’. Κάθε σχεδιαστικό πρόβλημα αρχίζει με μια προσπάθεια να επιτευχθεί μια συνεργασία μεταξύ δύο οντοτήτων: η μορφή και το πλαίσιό του. Η μορφή είναι η λύση στο πρόβλημα, το πλαίσιο καθορίζει το πρόβλημα. Το πλαίσιο και η μορφή είναι συμπληρωματικά. Μόλις υπάρξει το διάγραμμα των δυνάμεων υπό την κυριολεκτική έννοια, αυτό θα περιγράψει τη μορφή ως συμπληρωματικό διάγραμμα των δυνάμεων.

Αυτά τα misfits είναι οι δυνάμεις που πρέπει να διαμορφώσουν το πρότυπο και δεν πρέπει να μπερδεύονται. Μπορούμε να συνοψίσουμε την κατάσταση κάθε πιθανού misfit από μια μεταβλητή. Κάθε απροσάρμοστη (misfit) μεταβλητή αντιπροσωπεύει

ένα πιθανό είδος απροσάρμοστου μεταξύ της μορφής και του πλαισίου. Αυτές οι απροσάρμοστες μεταβλητές περιγράφουν μια παρούσα κατάσταση που δεν είναι ούτε μόνο στη μορφή, ούτε μόνο στο πλαίσιο, αλλά μια σχέση μεταξύ των δύο. Η κατάσταση αυτής της σχέσης, της τακτοποίησης ή του απροσάρμοστου, περιγράφει μια πτυχή ολόκληρου του συνόλου.

Αυτοί οι «όροι της αρμονίας» των δυνάμεων μεταξύ μιας μορφής και του πλαισίου του είναι αυτό που ο Alexander ήρθε αργότερα να καλέσει «πρότυπα». Τα απροσάρμοστα και οι καταστάσεις στα οποία τα απροσάρμοστα εξουσιάζουν τη «μορφή», αντιστοιχούν σε αυτό που πολλοί καλούν τώρα αντί-πρότυπα. Τα αντί-πρότυπα μπορούν να είναι χρήσιμα, επειδή είναι συχνά σημαντικό να δει κάποιος και να καταλάβει τις κακές λύσεις, δεδομένου ότι πρόκειται να δει και να καταλάβει τις καλές. Αυτό γίνεται γιατί κάποιος μπορεί να βοηθήσει να γίνει κατανοητό γιατί μια ιδιαίτερη εναλλακτική λύση να φανεί λογική, αλλά να μην είναι η καλύτερη διαδρομή που ακολουθεί. Οδηγεί μακριά από τα απροσάρμοστα και προς τα κατάλληλα πρότυπα, που μπορούν να λύσουν το πρόβλημά μας πιο αποτελεσματικά.

Αργότερα, οι Brown, Malveau, McCormick και Mowbray έγραψαν ένα βιβλίο με τίτλο «Anti-Patterns», το οποίο αναφέρεται στο «refactoring λογισμικό, αρχιτεκτονικές και προγράμματα σε κρίση.» Τα αντί-πρότυπα και οι λύσεις που περιγράφηκαν στο βιβλίο είναι παρόμοια με πολλά από τα «προβλήματα» που περιγράφηκαν από τον Bruce Webster στο «Pitfalls of Object-Oriented Development» και των «lessons learned» που περιγράφηκαν από τον Love Tom στο Object Lessons.

Αυτά τα βιβλία έχουν σκοπό να κάνουν πολλά περισσότερα από το να προσδιορίζουν και να εντοπίζουν τις διάφορες «αντι-λύσεις». Προσπαθούν να πάνε ένα βήμα παραπάνω προτείνοντας ένα σχέδιο δράσης για τη διόρθωση, την αποκατάσταση και την πρόληψη. Τα πιο χρήσιμα αντι-πρότυπα είναι εκείνα που δεν επισημαίνουν μόνο παραδείγματα κακής εξάσκησης, αλλά είναι για την αποκατάσταση και την ανασύνθεση του καλού από το κακό. Παρουσιάζουν πώς να μετατρέψουν τα λεμόνια σε λεμονάδα και πώς να αποφύγουν τα χαλασμένα λεμόνια.

1.8 Είδη προτύπων

Το βιβλίο GoF εστιάστηκε πολύ στα σχεδιαστικά πρότυπα (design patterns). Τα πρότυπα στο βιβλίο GoF είναι αντικειμενοστρεφή σχεδιαστικά πρότυπα. Υπάρχουν πολλά άλλα είδη προτύπων λογισμικού εκτός από τα σχεδιαστικά πρότυπα. Ο Martin Fowler έχει γράψει ένα σχετικό βιβλίο ανάλυσης προτύπων. Τα πρότυπα που υποβλήθηκαν στις προηγούμενες διασκέψεις PLoP έχουν καλύψει όλες τις πτυχές της τεχνολογίας λογισμικού συμπεριλαμβανομένων: οργάνωση ανάπτυξης, δημιουργία λογισμικού, προγραμματισμός προγράμματος, εφαρμοσμένη μηχανική απαιτήσεων και διαχείριση διαμόρφωσης λογισμικού. Τα σχεδιαστικά πρότυπα φαίνεται ακόμα οτι είναι τα δημοφιλέστερα.

Είδη σχεδιαστικών προτύπων

Το βιβλίο GoF καθορίζει τα σχεδιαστικά πρότυπα ως «περιγραφές της επικοινωνίας των αντικειμένων και των κλάσεων που προσαρμόζονται για να λύσουν ένα γενικό σχεδιαστικό πρόβλημα σε ένα ιδιαίτερο πλαίσιο.

Ένα σχεδιαστικό πρότυπο ονοματίζει περιλήψεις και προσδιορίζει τις πολύ βασικές πτυχές μιας κοινής δομής σχεδιασμού που το καθιστούν χρήσιμο για τη δημιουργία ενός χρήσιμου αντικειμενοστρεφούς προτύπου. Το σχεδιαστικό πρότυπο προσδιορίζει τις συμμετέχουσες κλάσεις και τα αντικείμενά τους, τους ρόλους και τις συνεργασίες τους και την κατανομή των αρμοδιοτήτων. Κάθε σχεδιαστικό πρότυπο εστιάζει σε ένα ιδιαίτερο αντικειμενοστραφές πρόβλημα ή θέμα. Περιγράφει πότε ισχύει, εάν μπορεί ή όχι να εφαρμοστεί λαμβάνοντας υπόψη άλλους περιορισμούς σχεδιασμού, και συνέπειες. Αφού πρέπει τελικά να εφαρμοστούν τα πρότυπα, ένα σχεδιαστικό πρότυπο παρέχει επίσης τον κώδικα για να επεξηγήσει μια εφαρμογή. Αν και τα σχεδιαστικά πρότυπα περιγράφουν αντικειμενοστρεφή σχεδιασμό, είναι βασισμένα σε πρακτικές λύσεις που εφαρμόζονται στις επικρατούσες αντικειμενοστρεφείς γλώσσες προγραμματισμού.

Η ανωτέρω περιγραφή συγκλίνει προς τον αντικειμενοστρεφή σχεδιασμό, αλλά με ελάχιστες αλλαγές, θα μπορούσε να ρυθμιστεί εύκολα για να περιγράψει τα σχεδιαστικά πρότυπα λογισμικού. Δεδομένου ότι το βιβλίο GoF ήταν το πρώτο και το δημοφιλέστερο των βιβλίων προτύπων λογισμικού, ο όρος «σχεδιαστικό πρότυπο» χρησιμοποιείται συχνά για να αναφερθεί σε οποιοδήποτε πρότυπο, το οποίο αντιμετωπίζει άμεσα τα ζητήματα της αρχιτεκτονικής λογισμικού, σχεδιασμό, ή εφαρμογή προγραμματισμού. Πολλοί επιλέγουν να κάνουν μια σημαντική διάκριση μεταξύ αυτών των τριών εννοιολογικών επιπέδων με την ταξινόμηση τους στα αρχιτεκτονικά πρότυπα, τα σχεδιαστικά πρότυπα και τους ιδιωματισμούς (οι ιδιωματισμοί καλούνται μερικές φορές σχέδια κωδικοποίησης). Οι συντάκτες του «Patterns of Software Architecture [POSA]» [17] [18] [19] [20] [21] [22] καθορίζουν αυτούς τους τρεις τύπους προτύπων ως εξής:

Αρχιτεκτονικά πρότυπα
Ένα αρχιτεκτονικό πρότυπο εκφράζει μια θεμελιώδη δομική οργάνωση ή ένα σχήμα για τα συστήματα λογισμικού. Παρέχει ένα σύνολο προκαθορισμένων υποσυστημάτων, διευκρινίζει τις ευθύνες τους και περιλαμβάνει τους κανόνες και τις οδηγίες για την οργάνωση των σχέσεων μεταξύ τους.

Σχεδιαστικά πρότυπα
Ένα σχεδιαστικό πρότυπο παρέχει ένα σχέδιο για τον καθορισμό των υποσυστημάτων ή των συστατικών ενός συστήματος λογισμικού, ή τις σχέσεις μεταξύ τους. Περιγράφει τη συνήθως επαναλαμβανόμενη δομή της επικοινωνίας των συστατικών που λύνει ένα γενικό πρόβλημα σχεδιασμού μέσα σε ένα ιδιαίτερο πλαίσιο.

Ιδιωματισμοί
Ένας ιδιωματισμός είναι ένα χαμηλού επιπέδου πρότυπο συγκεκριμένο για μια γλώσσα προγραμματισμού. Ένας ιδιωματισμός περιγράφει πώς θα εφαρμοστούν οι ιδιαίτερες πτυχές των συστατικών ή οι σχέσεις μεταξύ τους, χρησιμοποιώντας τα χαρακτηριστικά γνωρίσματα της δεδομένης γλώσσας.

Η διαφορά μεταξύ αυτών των τριών ειδών προτύπων είναι στα αντίστοιχα επίπεδά της αφαίρεσης και λεπτομέρειας.
Τα αρχιτεκτονικά πρότυπα είναι υψηλού επιπέδου στρατηγικές που αφορούν τα μεγάλης κλίμακας συστατικά και τις σφαιρικές ιδιότητες και τους μηχανισμούς ενός συστήματος. Έχουν τις επιπτώσεις που έχουν επιρροή στη γενική σκελετική δομή και την οργάνωση ενός συστήματος λογισμικού.

Τα σχεδιαστικά πρότυπα είναι μεσαίας κλίμακας τακτική που σκληραγωγούν ορίζουν τη δομή και τη συμπεριφορά των οντοτήτων και των σχέσεών τους. Δεν επηρεάζουν τη γενική δομή συστημάτων, αλλά καθορίζουν τις μικρό-αρχιτεκτονικές των υποσυστημάτων και των συστατικών.
Οι ιδιωματισμοί είναι συγκεκριμένοι με παραδείγματα και συγκεκριμένες γλωσσικά τεχνικές προγραμματισμού που συμπληρώνουν τις χαμηλού επιπέδου εσωτερικές ή εξωτερικές λεπτομέρειες της δομής ή της συμπεριφοράς ενός συστατικού.

Στο "Understanding and Using Patterns in Software Development" [14] οι Riehle και Zullighoven κάνουν τις παρόμοιες διακρίσεις, αλλά φαίνονται να χωρίζουν τα διαφορετικά είδη προτύπων μεταξύ της ανάλυσης, του σχεδιασμού και της εφαρμογής. Καθορίζουν τους όρους των εννοιολογικών προτύπων, τα σχεδιαστικά πρότυπα, και τα πρότυπα προγραμματισμού όπως φαίνεται πιο κάτω:

Εννοιολογικά πρότυπα
Ένα εννοιολογικό πρότυπο είναι ένα πρότυπο του οποίου η μορφή περιγράφεται με τη βοήθεια των όρων και των εννοιών από μια περιοχή εφαρμογής.

Σχεδιαστικά πρότυπα
Ένα σχεδιαστικό πρότυπο είναι ένα πρότυπο του οποίου η μορφή περιγράφεται με τη βοήθεια των κατασκευασμάτων σχεδίου λογισμικού, για παράδειγμα, αντικείμενα, κλάσεις, κληρονομικότητα και συνάθροιση.

Πρότυπα προγραμματισμού
Ένα πρότυπο προγραμματισμού είναι ένα πρότυπο του οποίου η μορφή περιγράφεται με τη βοήθεια των κατασκευασμάτων γλώσσας προγραμματισμού.

Χρησιμοποιώντας αυτούς τους ορισμούς, τα εννοιολογικά πρότυπα είναι βασισμένα στις μεταφορές σε μια περιορισμένη περιοχή εφαρμογής. Τα σχεδιαστικά πρότυπα συμπληρώνουν ή διαμορφώνουν τα εννοιολογικά πρότυπα με την έρευνα στην εφαρμογή των στοιχείων από το εννοιολογικό διάστημα. Και τα πρότυπα προγραμματισμού προχωρούν πιο πολύ στις λεπτομέρειες της εκτέλεσης χρησιμοποιώντας μια συγκεκριμένη γλώσσα εφαρμογής.

Συγκρίνοντας τους ορισμούς αυτούς, φαίνεται ότι τα πρότυπα προγραμματισμού είναι ισοδύναμα με τους ιδιωματισμούς. Για τους άλλους τύπους προτύπων που περιγράφονται πιο πάνω, το ένα σύνολο συντακτών επιλέγει να τους σκιαγραφήσει από το αρχιτεκτονικό πεδίο τους, ενώ το άλλο σύνολο συντακτών επιλέγει να τους σκιαγραφήσει, εάν υιοθετούν τη γλώσσα από το διάστημα προβλήματος ή το διάστημα λύσης.

Τελευταία αναπτύσσονται και συνδυασμοί προτύπων ανάλογα με το πρόβλημα που πρέπει να λυθεί σε κάθε περίπτωση. Αυτά ονομάζονται εξειδικευμένα πρότυπα.

1.9 Στοιχεία ενός προτύπου

Ο Alexander αναφέρει ότι «κάθε πρότυπο που καθορίζουμε πρέπει να διατυπωθεί υπό μορφή κανόνα, που καθιερώνει μια σχέση μεταξύ ενός πλαισίου, ενός συστήματος των δυνάμεων που προκύπτουν σε εκείνο το πλαίσιο και μιας διαμόρφωσης, η οποία επιτρέπει σε αυτές τις δυνάμεις για να επιλυθεί σε εκείνο το πλαίσιο. Προτείνει

επίσης τα εικονογραφικά παραδείγματα: «Πρώτα υπάρχει μια εικόνα που παρουσιάζει αρχέτυπο παράδειγμα του προτύπου.» Διάφορα διαφορετικά σχήματα έχουν χρησιμοποιηθεί για την περιγραφή των προτύπων. Το σχήμα περιγραφής προτύπων που χρησιμοποιήθηκε στην εργασία του Alexander καλείται «Alexandrian form». Το σχήμα χρησιμοποιούμενο μέσα στο GoF αναφέρεται ως «σχήμα GoF». Οι τίτλοι τμημάτων των παραγράφων που ακολουθούν αμέσως, αποτελούν αυτό που καλείται «κανονική μορφή» και μερικές φορές καλείται «Alexandrian form». Παρά τη χρήση αυτών των διαφορετικών σχημάτων προτύπων, είναι γενικά αναγνωρισμένο ότι ένα πρότυπο πρέπει να περιλάβει ορισμένα απαραίτητα στοιχεία. Τα ακόλουθα απαραίτητα στοιχεία πρέπει να είναι σαφώς αναγνωρίσιμα επάνω στην ανάγνωση ενός προτύπου:

Όνομα
Πρέπει να έχει ένα όνομα με νόημα. Αυτό επιτρέπει να χρησιμοποιηθεί μια μεμονωμένη λέξη ή μια σύντομη φράση για να αναφερθεί στο πρότυπο και στη γνώση και δομή που περιγράφει. Θα ήταν πολύ αδέξιο να πρέπει να περιγραφτεί ή ακόμα και να συνοψιστεί το πρότυπο κάθε φορά που χρησιμοποιείται σε μια συζήτηση. Τα καλά ονόματα προτύπων διαμορφώνουν ένα λεξιλόγιο για τη συζήτηση των εννοιολογικών αφαιρέσεων. Μερικές φορές ένα πρότυπο μπορεί να έχει περισσότερα από ένα συνήθως χρησιμοποιημένο ή αναγνωρίσιμο όνομα στη λογοτεχνία. Μερικές μορφές προτύπων παρέχουν επίσης μια «ταξινόμηση» του προτύπου εκτός από το όνομά της.

Πρόβλημα
Μια δήλωση του προβλήματος που περιγράφει την πρόθεσή του. Οι σκοποί και οι στόχοι που θέλει να επιτύχει μέσα στο δεδομένο πλαίσιο και τις δυνάμεις. Συχνά οι δυνάμεις αντιτάσσουν αυτούς τους στόχους καθώς επίσης και η μια την άλλη.

Πλαίσιο
Οι προϋποθέσεις κάτω από τις οποίες το πρόβλημα και η λύση του φαίνονται να επαναλαμβάνονται και για το οποίο η λύση είναι επιθυμητή. Αυτό μας λέει τη δυνατότητα εφαρμογής του προτύπου. Μπορεί να θεωρηθεί ως αρχική διαμόρφωση του συστήματος προτού να εφαρμοστεί το πρότυπο σε αυτό.

Δυνάμεις
Μια περιγραφή των σχετικών δυνάμεων και των περιορισμών και πώς αλληλεπιδρούν ή συγκρούονται το ένα με ένα άλλο και με τους στόχους που είναι επιθυμητό να επιτευχθούν. Ένα συγκεκριμένο σενάριο που χρησιμεύει ως το κίνητρο για το πρότυπο υιοθετείται συχνά. Οι δυνάμεις αποκαλύπτουν τα στοιχεία ενός προβλήματος και καθορίζουν τα είδη ανταλλαγών που πρέπει να εξεταστούν παρουσία της έντασης που δημιουργούν. Μια καλή περιγραφή προτύπων πρέπει να περιλαμβάνει όλες τις δυνάμεις που ασκούν επίδραση επάνω σε αυτό.

Λύση
Στατικές σχέσεις και δυναμικοί κανόνες περιγράφουν πώς θα πραγματοποιηθεί η επιθυμητή έκβαση. Αυτό είναι συχνά ισοδύναμο με την παροχή των οδηγιών που περιγράφουν πώς να κατασκευάσουν τα απαραίτητα προϊόντα εργασίας. Η περιγραφή μπορεί να καλύψει τις εικόνες, τα διαγράμματα και την πεζογραφία που προσδιορίζουν τη δομή του προτύπου, τους συμμετέχοντες και τις συνεργασίες τους,

για να επιδείξει πώς το λύνεται πρόβλημα. Η λύση πρέπει να περιγράψει όχι μόνο τη στατική δομή αλλά και τη δυναμική συμπεριφορά. Η στατική δομή μας λέει τη μορφή και την οργάνωση του προτύπου, αλλά συχνά είναι η δυναμική που κάνουν το πρότυπο «ζωντανό». Η περιγραφή της λύσης του προτύπου μπορεί να δείξει τις οδηγίες που λαμβάνονται υπόψη, τις παγίδες που αποφεύγονται κατά την προσπάθεια μιας συγκεκριμένης εφαρμογής της λύσης. Μερικές φορές περιγράφονται οι πιθανές παραλλαγές ή οι ειδικεύσεις της λύσης.

Παραδείγματα
Μια ή περισσότερες εφαρμογές δειγμάτων του προτύπου επεξηγούν: ένα συγκεκριμένο αρχικό πλαίσιο για το πώς εφαρμόζεται το πρότυπο και μετατρέπει εκείνο το πλαίσιο και το προκύπτον πλαίσιο. Τα παραδείγματα βοηθούν τον αναγνώστη να καταλάβει τη χρήση και τη δυνατότητα εφαρμογής του προτύπου. Τα οπτικά παραδείγματα και οι αναλογίες μπορούν συχνά να είναι ιδιαίτερα ωφέλιμα. Ένα παράδειγμα μπορεί να συμπληρωθεί από μια εφαρμογή δειγμάτων για να παρουσιάσει τη μόνη λύση που πρέπει να πραγματοποιηθεί.

Προκύπτουν πλαίσιο
Η κατάσταση ή η διαμόρφωση του συστήματος μετά την εφαρμογή του προτύπου, συμπεριλαμβανομένων των συνεπειών, εφαρμόζοντας το πρότυπο και άλλα προβλήματα και πρότυπα που μπορούν να προκύψουν από το νέο πλαίσιο. Περιγράφει τις συνέπειες και τις παρενέργειες του προτύπου. Αυτό καλείται μερικές φορές ψήφισμα των δυνάμεων, επειδή περιγράφει ποιες δυνάμεις έχουν επιλυθεί, ποιες παραμένουν εκκρεμείς και πού μπορούν να ισχύουν τα πρότυπα. Τεκμηριώνοντας το προκύπτον πλαίσιο που παράγεται από ένα πρότυπο, βοηθάει να συσχετιστεί με το αρχικό πλαίσιο άλλων προτύπων. (ένα ενιαίο πρότυπο είναι συχνά μόνο ένα βήμα προς την ολοκλήρωση κάποιου μεγαλύτερου στόχου ή προγράμματος).

Λογική
Μια εξήγηση των βημάτων ή των κανόνων του προτύπου και επίσης του προτύπου συνολικά από την άποψη πώς και γιατί επιλύει τις δυνάμεις του με έναν ιδιαίτερο τρόπο για να είναι κοντά στους επιθυμητούς στόχους, τις αρχές, και τη φιλοσοφία. Εξηγεί πώς οι δυνάμεις και οι περιορισμοί είναι ενορχηστρωμένοι στη συναυλία για να επιτύχουν αρμονία. Αυτό λέει πώς το πρότυπο λειτουργεί πραγματικά, γιατί λειτουργεί και γιατί είναι «καλό». Το τμήμα λύσης ενός προτύπου μπορεί να περιγράψει τη φαινομενικά ορατή δομή και τη συμπεριφορά του προτύπου, αλλά η λογική είναι τι παρέχουν οι βαθιές δομές και οι βασικοί μηχανισμοί κάτω από την επιφάνεια του συστήματος.

Σχετικά πρότυπα
Τα σχετικά πρότυπα μοιράζονται συχνά τις κοινές δυνάμεις. Συχνά έχουν ένα αρχικό ή προκύπτον πλαίσιο που είναι συμβατό με την κατάληξη ή το αρχικό πλαίσιο ενός άλλου προτύπου. Τέτοια πρότυπα μπορεί να είναι πρότυπα προγενέστερων, των οποίων η εφαρμογή οδηγεί σε αυτό το πρότυπο. Πρότυπα μεταγενέστερων, η εφαρμογή των οποίων προκύπτει από αυτό το πρότυπο. Εναλλακτικά πρότυπα που περιγράφουν μια διαφορετική λύση στο ίδιο πρόβλημα αλλά κάτω από τις διαφορετικές δυνάμεις και τους περιορισμούς που μπορούν ή πρέπει να εφαρμοστούν ταυτόχρονα με αυτό το πρότυπο.

Τα καλά πρότυπα αρχίζουν συχνά με μια περίληψη που παρέχει μια σύντομη περίληψη ή μια επισκόπηση. Αυτό δίνει στους αναγνώστες μια σαφή εικόνα του προτύπου και τους ενημερώνει γρήγορα για τη σχετικότητά του στα προβλήματα που επιθυμούν να λύσουν. Ένα πρότυπο πρέπει να προσδιορίσει το συγκεκριμένο φορέα του και να καταστήσει σαφές στον αναγνώστη τι προϋποθέτει.

1.10 Ιδιότητες ενός προτύπου

Εκτός από τα προαναφερθέντα στοιχεία, ένα καλογραμμένο πρότυπο πρέπει να εκθέσει διάφορες επιθυμητές ιδιότητες. Ο Lea Doug, στο έγγραφό του «Christopher Alexander: an Introduction for Object-Oriented Designers» [23] [24] παρέχει μια λεπτομερή περιγραφή αυτών των ιδιοτήτων και συνοψίζονται πιο κάτω.

Ενθυλάκωση και αφαίρεση
Κάθε πρότυπο τοποθετεί ένα πρόβλημα και τη λύση του μαζί σε μια ιδιαίτερη περιοχή. Τα πρότυπα πρέπει να παρέχουν τα σαφή όρια που βοηθούν να ξεκαθαριστεί το διάστημα προβλήματος και το διάστημα λύσης. Χρησιμεύουν επίσης ως αφαιρέσεις που ενσωματώνουν τη γνώση και την εμπειρία περιοχών και μπορούν να εμφανιστούν σε ποικίλα ιεραρχικά εννοιολογικά επίπεδα μέσα στην περιοχή αυτή.

Ανοικτότητα και μεταβλητότητα
Κάθε πρότυπο πρέπει να είναι ανοικτό για την επέκταση ή την παραμετροποίηση από άλλα πρότυπα έτσι ώστε να μπορούν να λειτουργήσουν μαζί για να λύσουν ένα μεγαλύτερο πρόβλημα. Μια λύση προτύπων πρέπει να είναι ικανή να πραγματοποιηθεί από μια άπειρη ποικιλία εφαρμογών.

Generativity και ικανότητα σύνθεσης.
Κάθε πρότυπο μόλις εφαρμοστεί, παράγει ένα προκύπτον πλαίσιο που ταιριάζει στο αρχικό πλαίσιο ενός ή περισσότερων άλλων προτύπων σε μια γλώσσα προτύπων. Αυτά τα επόμενα πρότυπα μπορούν έπειτα να εφαρμοστούν για να προχωρήσουν περαιτέρω προς τον τελικό στόχο ολοκλήρωσης της γενικής λύσης. Τα πρότυπα εφαρμόζονται με τα μέσα της αποσπασματικής αύξησης (piecemeal growth). Η εφαρμογή ενός προτύπου παρέχει ένα πλαίσιο για την εφαρμογή του επόμενου προτύπου. Αλλά τα πρότυπα δεν είναι απλά γραμμικής φύσης και μπορούν να συντεθούν με άλλα πρότυπα σε ποικίλα επίπεδα κλίμακας.

Ισορροπία
Κάθε πρότυπο πρέπει να πραγματοποιήσει κάποιο είδος ισορροπίας μεταξύ των δυνάμεων και των περιορισμών του. Αυτό μπορεί να οφείλεται σε μια ή περισσότερες σταθερές ή ευρετικές (heuristics) που χρησιμοποιούνται για να ελαχιστοποιήσουν τη σύγκρουση μέσα στο διάστημα λύσης. Ευρετικές είναι όρος που χρησιμοποιείται στα μαθηματικά ή την πληροφορική από το αρχαίο ευρίσκω. Οι σταθερές απεικονίζουν συχνά μια ελλοχεύουσα αρχή ή μια φιλοσοφία επίλυσης προβλήματος για την ιδιαίτερη περιοχή και παρέχουν μια λογική για κάθε βήμα/τον κανόνα στο πρότυπο.

Ο στόχος είναι ότι, εάν γράφεται καλά, κάθε πρότυπο περιγράφει ένα σύνολο που είναι μεγαλύτερο από το άθροισμα των μερών του, λόγω της ικανότητας των στοιχείων του που λειτουργούν μαζί για να ικανοποιήσει όλες τις ποικίλες απαιτήσεις του.

1.11 Πρότυπα, κανόνες και δημιουργικότητα

Είναι η συνδυασμένη παρουσία όλων αυτών των στοιχείων προτύπων και ιδιοτήτων που καθιστούν τα πρότυπα κάτι πολύ περισσότερο από heuristics, κανόνες, ή αλγορίθμους. Τα Heuristics και οι αρχές συμμετέχουν συχνά στις δυνάμεις ή και τη λογική ενός προτύπου, αλλά είναι μόνο ένα στοιχείο του προτύπου. Επιπλέον, όπως ο Cope γράφει στο «Software Design Patterns: Common Questions and Answers»:
Οι κανόνες δεν υποστηρίζονται από μια λογική, ούτε τοποθετούνται σε πλαίσιο. Ένας κανόνας μπορεί να είναι μέρος της λύσης σε μια περιγραφή προτύπου, αλλά μια λύση κανόνα να μην είναι ούτε ικανοποιητική ούτε απαραίτητη. Τα πρότυπα δεν σχεδιάζονται για να εκτελεστούν ή να αναλυθούν από τους υπολογιστές, όπως κάποιος μπορεί να φανταστεί. Τα πρότυπα πρόκειται να εκτελεστούν από αρχιτέκτονες με διορατικότητα, προτίμηση, εμπειρία, και αίσθηση της αισθητικής.

Ένα πρότυπο είναι η διαδικασία που παράγει μια λύση, αλλά αυτό μπορεί να παράγει έναν τεράστιο αριθμό των διάφορων λύσεων. Το ανθρώπινο στοιχείο των προτύπων είναι τι συμβάλλει κυρίως στη μεταβλητότητα και την προσαρμοστικότητά τους και απαιτεί συνήθως έναν μεγαλύτερο βαθμό δημιουργικότητας στην εφαρμογή και το συνδυασμό τους. Όσο οι διαδικασίες της αρχιτεκτονικής και του προτύπου είναι δημιουργικές προσπάθειες, τόσο είναι και η εφαρμογή των προτύπων. Στο ίδιο έγγραφο που αναφέρεται ανωτέρω, ο Cope αναφέρει:
Η δημιουργικότητα απαιτείται ακόμα και για να διαμορφώσει τα πρότυπα σε ένα δεδομένο πλαίσιο. Ακριβώς όπως μια μοδίστρα προσαρμόζει ένα πρότυπο σε έναν μεμονωμένο πελάτη, και ίσως σε ένα συγκεκριμένο γεγονός όπου το φόρεμα πρόκειται να φορεθεί, έτσι οι σχεδιαστές πρέπει να είναι δημιουργικοί κατά χρησιμοποίηση των προτύπων.

1.12 Πρότυπα και αλγόριθμοι

Το προηγούμενο τμήμα για τα πρότυπα εναντίον των κανόνων ισχύει επίσης σε μεγάλο βαθμό για τους αλγορίθμους και τις δομές δεδομένων τους. Βεβαίως, οι αλγόριθμοι και οι δομές δεδομένων μπορούν να χρησιμοποιηθούν στην εφαρμογή ενός ή περισσότερων προτύπων, αλλά οι αλγόριθμοι και οι δομές δεδομένων λύνουν γενικά υπολογιστικά προβλήματα όπως η ταξινόμηση και η αναζήτηση. Τα πρότυπα ενδιαφέρονται για αρχιτεκτονικά ζητήματα που έχουν μεγαλύτερης κλίμακας αποτελέσματα. Τα πρότυπα έχουν ορισμένα ζητήματα ανάπτυξης όπως η συντηρησιμότητα, η ικανότητα επαναχρησιμοποίησης και η παραλλαγή ενθυλάκωσης. Αυτά είναι ζητήματα που απασχολούν τους ανθρώπους και πρέπει να εξελιχθούν με την πάροδο του χρόνου.

Οι αλγόριθμοι και οι δομές δεδομένων ενδιαφέρονται συνήθως σχεδόν αποκλειστικά για τη βελτιστοποίηση του χώρου ή του χρόνου ή κάποιας άλλης πτυχής της υπολογιστικής κατανάλωσης πολυπλοκότητας και των πόρων. Είναι για την εύρεση των συμπαγέστερων και αποδοτικών μέσων να εκτελεστεί κάποιος σημαντικός υπολογισμός ή να αποθηκευτούν και να ανακληθούν τα αποτελέσματά του. Τέτοιοι αλγόριθμοι και δομές δεδομένων ενδιαφέρονται για άλλες ανησυχίες με πράγματα όπως η απόδοση ή η μνήμη και λιγότερο με τη συντηρησιμότητα και την προσαρμοστικότητα και τη δυνατότητα χρησιμοποίησης της αρχιτεκτονικής.

Υπάρχουν πολλά βιβλία των αλγορίθμων και των δομών δεδομένων που παρέχουν το πηγαίο κώδικα και την ποσοτική ανάλυση για τις δομές όπως τα δέντρα AVL ή τα splay-δέντρα. Αλλά σε πολλά από αυτά τα ίδια εγχειρίδια υπάρχει μικρή αναφορά για το πώς θα εφαρμοστούν αυτές οι δομές με τρόπους που υπογραμμίζουν τη συντηρησιμότητα , την προσαρμοστικότητα και την επαναχρησιμοποίηση. Η εξέταση των υπολογιστικών πτυχών χρόνου και χώρου απλά δεν εξετάζει το πληρέστερο σύνολο δυνάμεων που έχουν επιπτώσεις στους αρχιτέκτονες και τους εφαρμοστές καθώς επίσης και τους χρήστες.
Για αυτό οι αλγόριθμοι και οι δομές δεδομένων αντιμετωπίζουν συνήθως τα ζητήματα της υπολογιστικής πολυπλοκότητας και όχι τόσο πολύ τα υπόλοιπα ζητήματα των ανθρώπων που χρησιμοποιούν και δημιουργούν το λογισμικό. Τα πρότυπα απευθύνονται στους ανθρώπους για θέματα, όπως τη συντηρησιμότητα, αντί για τα απλά ζητήματα αποδοτικότητας υλικού και λογισμικού/μνήμης.

Φυσικά οι προγραμματιστές λογισμικού πρέπει να ενδιαφερθούν και για την εύρεση των κατάλληλων αρχιτεκτονικών και για την εύρεση των κατάλληλων λύσεων στα υπολογιστικά προβλήματα. Έτσι θα υπάρξει πάντα μια ανάγκη για τα πρότυπα καθώς και για τους αλγορίθμους, τις δομές δεδομένων και τη χρήση τους μαζί.

1.13 Πρότυπα και πλαίσια

Ένα πράγμα στενά συνδεδεμένο με το σχεδιασμό προτύπων και τον αντικείμενο-προσανατολισμό είναι ένα πλαίσιο λογισμικού:
Ένα πλαίσιο λογισμικού είναι μια επαναχρησιμοποιήσιμη μίνι-αρχιτεκτονική που παρέχει τη γενική δομή και τη συμπεριφορά για μια οικογένεια αφαιρέσεων λογισμικού και ένα πλαίσιο μεταφορών που διευκρινίζει τη συνεργασία και τη χρήση τους μέσα σε μια δεδομένη περιοχή.
Το πλαίσιο το ολοκληρώνει αυτό, μετατρέποντας τα συμφραζόμενα σε ένα είδος «εικονικής μηχανής», σχεδιάζοντας τις αφαιρέσεις με συγκεκριμένα plug-points (επίσης αποκαλούμενα hot spots). Αυτά τα plug-points, επιτρέπουν στο πλαίσιο να προσαρμοστεί και να επεκταθεί για να ανταποκριθεί στις ποικίλες ανάγκες και να συντεθεί επιτυχώς σε άλλα πλαίσια. Ένα πλαίσιο δεν είναι συνήθως μια πλήρης εφαρμογή. Στερείται συχνά την απαραίτητη λειτουργικότητα που ορίζεται από την εφαρμογή. Μια εφαρμογή μπορεί να κατασκευαστεί από ένα ή περισσότερα πλαίσια, βάζοντας την λειτουργικότητα στις έτοιμες προς χρήση «εξόδους» που παρέχονται από τα πλαίσια. Κατά συνέπεια, ένα πλαίσιο παρέχει την υποδομή και τους μηχανισμούς για την αλληλεπίδραση μεταξύ των αφηρημένων συστατικών με τις ανοικτές εφαρμογές.

Ένας ορισμός ενός αντικειμενοστρεφούς πλαισίου λογισμικού δίνεται από το GoF:
Ένα πλαίσιο είναι ένα σύνολο συνεργαζόμενων κλάσεων που αποτελούν ένα επαναχρησιμοποιήσιμο σχεδιασμό για μια συγκεκριμένη κλάση λογισμικού. Ένα πλαίσιο παρέχει τις αρχιτεκτονικές οδηγίες με το χωρισμό του προτύπου στις αφηρημένες κλάσεις και τον ορισμό των ευθυνών και των συνεργασιών τους. Ένας υπεύθυνος για την ανάπτυξη προσαρμόζει ένα πλαίσιο σε μια ιδιαίτερη εφαρμογή με το χωρισμό σε υποκλάσεις και τη σύνθεση των αντικειμένων των κλάσεων πλαισίου.
Ένα πλαίσιο υπαγορεύει την αρχιτεκτονική της αίτησης. Θα ορίσει τη γενική δομή, το χωρισμό του σε κλάσεις και αντικείμενα, τις βασικές ευθύνες, πώς συνεργάζονται οι κλάσεις και τα αντικείμενα και το νήμα του ελέγχου. Ένα πλαίσιο προκαθορίζει αυτές τις παραμέτρους προτύπου έτσι, ώστε ο σχεδιαστής εφαρμογής ή ο εφαρμοστής, να

μπορεί να επικεντρωθεί στις λεπτομέρειες της αίτησης. Το πλαίσιο συλλαμβάνει τις αποφάσεις σχεδιασμού που είναι κοινές για την περιοχή εφαρμογής. Τα πλαίσια υπογραμμίζουν έτσι την επαναχρησιμοποίηση προτύπου πέρα από την επαναχρησιμοποίηση κώδικα, αν και ένα πλαίσιο θα περιλάβει τις συγκεκριμένες υποκλάσεις που μπορεί να μπουν στην εργασία

Η διαφορά μεταξύ ενός πλαισίου και μιας συνηθισμένης βιβλιοθήκης προγραμματισμού είναι ότι ένα πλαίσιο υιοθετεί μια ροή του ελέγχου μεταξύ αυτού και των πελατών του. Κατά τη χρησιμοποίηση ενός πλαισίου, συνήθως εφαρμόζει μερικές λειτουργίες επανάκλησης ή ειδικεύει μερικές κλάσεις και επικαλείται έπειτα μια ενιαία μέθοδο ή μια διαδικασία. Σε αυτό το σημείο, το πλαίσιο κάνει το υπόλοιπο της εργασίας για αυτόν, που επικαλείται οποιεσδήποτε απαραίτητες επανακλήσεις ή μεθόδους πελατών με την πρώτη ευκαιρία χρόνου και χώρου. Για αυτόν τον λόγο, τα πλαίσια λέγονται συχνά ότι τηρούν την αρχή Hollywood («μην μας καλέστε, θα σας καλέσουμε. ») ή την Greyhound αρχή («Αφήστε την καθοδήγηση σε εμάς »).

Τα σχεδιαστικά πρότυπα μπορούν να χρησιμοποιηθούν και στο σχεδιασμό και στην τεκμηρίωση ενός πλαισίου. Ένα ενιαίο πλαίσιο καλύπτει χαρακτηριστικά διάφορα σχεδιαστικά πρότυπα. Στην πραγματικότητα, ένα πλαίσιο μπορεί να αντιμετωπισθεί ως εφαρμογή ενός συστήματος των σχεδιαστικών προτύπων. Παρά το γεγονός ότι συσχετίζονται κατά αυτόν τον τρόπο, είναι σημαντικό να αναγνωριστεί ότι τα πλαίσια και τα σχεδιαστικά πρότυπα είναι χωριστά. Ένα πλαίσιο είναι εκτελέσιμο λογισμικό, ενώ τα σχεδιαστικά πρότυπα αναπαριστούν τη γνώση και την εμπειρία για το λογισμικό. Από αυτή την άποψη, τα πλαίσια είναι φυσικού χαρακτήρα, ενώ τα πρότυπα είναι λογικής φύσης. Τα πλαίσια είναι η φυσική πραγματοποίηση μιας ή περισσότερων λύσεων προτύπων λογισμικού, τα πρότυπα είναι οι οδηγίες για το πώς θα εφαρμοστούν εκείνες τις λύσεις.

Το βιβλίο GoF περιγράφει τις μεγαλύτερες διαφορές μεταξύ των σχεδιαστικών προτύπων και των πλαισίων ως εξής:
1). Τα σχεδιαστικά πρότυπα είναι πιο περιληπτικά από τα πλαίσια. Τα πλαίσια μπορούν να ενσωματωθούν στον κώδικα, αλλά μόνο τα παραδείγματα των προτύπων μπορούν να ενσωματωθούν στον κώδικα. Μια δύναμη των πλαισίων είναι ότι μπορούν να γραφτούν στις γλώσσες προγραμματισμού και όχι μόνο να μελετηθούν αλλά να εκτελεστούν και να επαναχρησιμοποιηθούν άμεσα. Αντίθετα, τα σχεδιαστικά πρότυπα πρέπει να εφαρμοστούν κάθε φορά που χρησιμοποιούνται. Τα σχεδιαστικά πρότυπα εξηγούν επίσης την πρόθεση, τις ανταλλαγές και τις συνέπειες ενός προτύπου.
2). Τα σχεδιαστικά πρότυπα είναι μικρότερα αρχιτεκτονικά στοιχεία από τα πλαίσια. Ένα χαρακτηριστικό πλαίσιο περιέχει διάφορα σχεδιαστικά πρότυπα αλλά το αντίστροφο δεν ισχύει ποτέ.
3). Τα σχεδιαστικά πρότυπα είναι λιγότερο εξειδικευμένα από τα πλαίσια. Τα πλαίσια έχουν πάντα μια ιδιαίτερη περιοχή εφαρμογής. Σε αντίθεση, τα σχεδιαστικά πρότυπα μπορούν να χρησιμοποιηθούν σχεδόν σε οποιοδήποτε είδος εφαρμογής. Ενώ τα πιο εξειδικευμένα σχεδιαστικά πρότυπα είναι πιθανά, ακόμη και αυτά δεν θα υπαγόρευαν μια αρχιτεκτονική εφαρμογής.

1.14 Η ποιότητα χωρίς όνομα

Η «ποιότητα χωρίς όνομα» «QWAN-Quality Without A Name» είναι «η ποιότητα» που μεταδίδει μη ανακοινώσιμη ομορφιά και ανυπολόγιστη αξία σε μια δομή. Καλύπτει όλα τα ακόλουθα:

- παγκοσμίως αναγνωρίσιμη αισθητική ομορφιά και σειρά
- κατ' επανάληψη τοποθετημένα κέντρα συμμετρίας και ισορροπίας
- ζωή και πληρότητα
- ανθεκτικότητα, προσαρμοστικότητα και διάρκεια
- ανθρώπινη άνεση και ικανοποίηση
- συναισθηματική και γνωστική αντήχηση

Ο Alexander προτείνει την ύπαρξη μιας αντικειμενικής ποιότητας της αισθητικής ομορφιάς που είναι παγκοσμίως αναγνωρίσιμη. Υποστηρίζει ότι υπάρχουν ορισμένες ιδιότητες χωρίς ιδιαίτερη σημασία και ιδιότητες που θεωρούνται όμορφες και προκαλούν ευχάριστη αίσθηση σε όλους τους ανθρώπους σε όλους τους πολιτισμούς. Είναι αυτές οι θεμελιώδεις ιδιότητες που συνδυάζουν να παραγάγουν το QWAN, και που κάνουν μια δομή να αισθάνεται «ολόκληρη» και «ζωντανή».

Ο Alexander πραγματοποίησε μερικά πειράματα χρησιμοποιώντας διαμορφώσεις των χρωματισμένων χαντρών και τα πρότυπα των ταπήτων. Εκείνα τα αποτελέσματα, συνδυαζόμενα με την αρχιτεκτονική εμπειρία των χρηστών των δημιουργιών του, πρότειναν ότι η παρουσία αυτής της «αντικειμενικής ομορφιάς» δέθηκε πολύ με την παρουσία συμμετριών και ότι ισορρόπησε τη χρήση του αντιπαραβαλλόμενου διαστήματος, του φωτός, και του χρώματος για να διαμορφώσει τους τομείς των οπτικών κέντρων. Τα συναισθήματα της ομορφιάς και της σειράς θα αυξάνονταν όταν τα οπτικά κέντρα έρχονταν σε επαφή με πολλαπλάσια ιεραρχικά επίπεδα σε ένα σχεδιασμό. Εντούτοις, εάν οι συμμετρίες είναι πάρα πολύ καθαρές ή τέλειες, φαίνεται να είναι λιγότερο επιθυμητό, από το εάν υπάρχουν μικρές παρατυπίες και ατέλειες. Προφανώς μερικές ατέλειες μπορούν να είναι χρηστικές.

Στο «The Laws of Architecture from a Physicist's Perspective» [25], ο καλλιτέχνης και μαθηματικός Νίκος Α. Σαλίγκαρος, που έχει συνεργαστεί στενά με τον Alexander κατά τη διάρκεια των προηγούμενων δέκα ετών, συζητούν τους κανόνες της ομορφιάς και της διάταξης και προτείνουν τους ακόλουθους τρεις νόμους της αρχιτεκτονικής:
1). Η διάταξη με τη μικρότερη κλίμακα καθιερώνεται από τα ταξινομημένα κατά ζεύγος αντιπαραβαλλόμενα στοιχεία, που υπάρχουν σε μια ισορροπημένη οπτική ένταση.
2). Η μεγάλης κλίμακας διάταξη εμφανίζεται όταν κάθε στοιχείο σχετίζεται με άλλο στοιχείο σε μια απόσταση που με τέτοιο τρόπο, ώστε μειώνεται η εντροπία.
3). Η μικρή κλίμακα συνδέεται με τη μεγάλη κλίμακα μέσω μιας συνδεμένης ιεραρχίας των ενδιάμεσων κλιμάκων με τον παράγοντα κλίμακας περίπου ίσο με ε = 2.718.

Ο Σαλίγκαρος υποστηρίζει, ότι αυτοί οι νόμοι δεν κυβερνούν την ομορφιά στην αρχιτεκτονική, κυβερνούν τη ζωή στην αρχιτεκτονική: «Εξάγονται από τη φυσική και τα μαθηματικά ερευνώντας το πώς η φύση συγκροτείται. Ανακάλυψε αυτούς τους νόμους παρατηρώντας πώς τα θεμελιώδη μόρια ενώνονται να διαμορφώσουν μια δομή.» Λέει, ότι όταν προσαρμόζονται οι δομές σε αυτούς τους αρχιτεκτονικούς νόμους, εκείνοι που τους εισάγουν αισθάνονται ένα είδος ηχηρής αρμονίας, μια σχεδόν συναισθηματική σύνδεση στη δομή. Αυτή η σύνδεση μεταξύ της ζωής και της

αρχιτεκτονικής οφείλεται στη θερμοδυναμική των μορφών διαβίωσης, αλλά η ιδέα του κτισίματος δομών που κατέχουν αυτές τις εγγενείς ιδιότητες των βιολογικών οργανισμών διαβίωσης προέρχεται από τον Alexander.

Στο "Reengineering the Application Development Process" [26], ο Michael Beedle προσπαθεί να περιγράψει περιληπτικά το QWAN (Quality Without a Name) ως κάτι που δημιουργείται, όταν οι ιδιότητες στο πρότυπο κάνουν το πρότυπο «ζωντανό». Δηλαδή σχεδιάζει κάτι που είναι εύκαμπτο, προσαρμόσιμο, επαναχρησιμοποιήσιμο και έχει άλλες ιδιότητες των πραγμάτων διαβίωσης.

Σύμφωνα με αυτόν τον ορισμό, το QWAN επιτυγχάνεται όταν παράγει το πρότυπο ή η γλώσσα προτύπων παράγει μια «ζωντανή» αρχιτεκτονική λύση. Κάτι που μιμείται την ικανότητα των πραγμάτων διαβίωσης, προσαρμόζεται δυναμικά για να εκπληρώσει τις μεταβαλλόμενες ανάγκες και τις απαιτήσεις. Αυτό είναι στενά συνδεδεμένο στην ιδέα του Alexander της generativity. «Στο Christopher Alexander: an Introduction for Object-Oriented Designers» [24], ο Lea Doug προσπαθεί να περιγράψει το QWAN ως εξής:
«η ποιότητα χωρίς όνομα», η κατοχή του οποίου είναι ο τελικός σκοπός οποιουδήποτε προϊόντος προτύπου. Είναι αδύνατο να συνοψιστεί εν συντομία αυτό. Ο Alexander παρουσιάζει διάφορα μερικά συνώνυμα: η ελευθερία, η ζωή, η πληρότητα, η άνεση, η αρμονία, αλλά κανένας όρος ή παράδειγμα δεν μεταβιβάζουν πλήρως τη σημασία των γραφών του Alexander στον αναγνώστη.

Έτσι μια άλλη κρίσιμη πτυχή του QWAN είναι η επίδραση που έχει επάνω στους χρήστες και τους σχεδιαστές της αρχιτεκτονικής που τους κάνει να αισθάνονται ζωντανοί, ολόκληροι, και άνετοι.
Φυσικά αυτός ο «τελικός σκοπός» της επίτευξης του QWAN είναι πολύ αόριστος. Αυτή η ιδιαίτερη πτυχή των προτύπων φαίνεται να δανείζεται τις έννοιες από το βουδισμό Zen, Taoism, και τα πλατωνικά ιδανικά. Υπάρχουν εκείνοι που αισθάνονται ότι ολόκληρη η ιδέα του QWAN είναι πολύ ιδιότροπη και μεταφυσική, ότι δεν είναι επιστημονικό ή αρκετά απτό να έχει μια θέση στις αληθινές ειδικότητες εφαρμοσμένης μηχανικής. Η έννοια της ομορφιάς ως κάτι που είναι παγκοσμίως αντικειμενικό μπορεί να είναι ένα πράγμα σκληρό για μερικούς να δεχτούν.

Αλλά από πολλές απόψεις, η αίσθηση ενός ατόμου του QWAN είναι επίσης για τη γνωστική κρίση. Κάθε κύριος σχεδιαστής αναπτύσσει την δικιά του διαίσθησή από την εκτενή εμπειρία. Αν και αυτή η «διαίσθηση» μπορεί να είναι υποκειμενική, μπορεί να είναι παράξενα ακριβής και να δώσει στο σχεδιαστή μια σχεδόν ενστικτώδη αίσθηση τι θα λειτουργήσει και τι όχι. Εάν ένα πρότυπο μπορεί να μεταδώσει στους αναγνώστες και τους χρήστες του, αυτό το ίδιο «συνδεμένο» συναίσθημα της σύνδεσης με το σχέδιο και της βαθιάς κατανόησης του, κατόπιν θεωρητικά θα μεταδώσει στον αναγνώστη το ίδιο γνωστικό συναίσθημα της ορθότητάς του που ο σχεδιαστής δοκίμασε. Εάν ένα πρότυπο πετύχει σε αυτήν την προσπάθεια, τότε όλοι που βλέπουν και το χρησιμοποιούν αυτό θα δοκιμάσουν υποθετικά το ηχηρό συναίσθημα της ομορφιάς και της αρμονίας, ότι το QWAN υποτίθεται ότι θα καλεστεί.

1.15 Γλώσσες προτύπων

Μια συλλογή των προτύπων διαμορφώνει ένα λεξιλόγιο για την κατανόηση και την επικοινωνία των ιδεών. Μια τέτοια συλλογή μπορεί να υφαθεί επιτυχώς σε ένα συνεκτικό «σύνολο» που αποκαλύπτει τις έμφυτες δομές και τις σχέσεις των μερών του προς την πραγματοποίηση ενός κοινού στόχου. Αυτό είναι αυτό που ο Alexander καλεί γλώσσα προτύπων. Εάν ένα πρότυπο είναι μια επαναλαμβανόμενη λύση σε ένα πρόβλημα σε ένα πλαίσιο που δίνεται από μερικές δυνάμεις, τότε μια γλώσσα προτύπων είναι ένα σύνολο τέτοιων λύσεων που, σε κάθε επίπεδο της κλίμακας, δουλεύουν για να επιλύσουν μαζί ένα σύνθετο πρόβλημα σε μια τακτική λύση σύμφωνα με έναν προκαθορισμένο στόχο. Ο Cope ορίζει μια γλώσσα προτύπου ως εξής:
Μια γλώσσα προτύπων ορίζει μια συλλογή των προτύπων και των κανόνων για να συνδυαστούν σε ένα αρχιτεκτονικό ύφος. Οι γλώσσες προτύπων περιγράφουν τα πλαίσια λογισμικού ή τις οικογένειες των σχετικών συστημάτων.

Ο Cope δίνει έναν ελαφρώς διαφορετικό ορισμό στο"Software Design Patterns: Common Questions and Answers “ [27].
Μια γλώσσα προτύπων είναι μια δομημένη συλλογή των προτύπων που στηρίζονται το ένα στο άλλο για να μετασχηματίσουν τις ανάγκες και τους περιορισμούς σε μια αρχιτεκτονική.
Επίσης ο Cope υποστηρίζει ότι:
Οι καλές γλώσσες προτύπων καθοδηγούν το σχεδιαστή προς τις χρήσιμες αρχιτεκτονικές και μακριά από τις αρχιτεκτονικές που είναι βασισμένες σε ασυναρτησίες. Οι καλές αρχιτεκτονικές είναι ανθεκτικές, λειτουργικές και αισθητικά ευχάριστες και ένας καλός συνδυασμός προτύπων μπορεί να ισορροπήσει τις δυνάμεις σε ένα σύστημα για να προσπαθήσει προς αυτούς τους τρεις στόχους. Μια καλή γλώσσα προτύπων δίνει την ελευθερία στους σχεδιαστές να εκφραστούν και να προσαρμόσουν τη λύση στις ιδιαίτερες ανάγκες του πλαισίου όπου τα πρότυπα εφαρμόζονται.

Μια γλώσσα προτύπων περιλαμβάνει τους κανόνες και τις οδηγίες που εξηγούν πώς και πότε να εφαρμόσουν τα πρότυπα της, για να λύσουν ένα πρόβλημα που είναι μεγαλύτερο από οποιοδήποτε μεμονωμένο πρότυπο που μπορεί να λύσει. Αυτοί οι κανόνες και οδηγίες προτείνουν την εφαρμογή κάθε προτύπου στη γλώσσα. Μια γλώσσα προτύπων μπορεί να θεωρηθεί ως λεξικό των προτύπων συν μια γραμματική που καθορίζει πώς να τα συνθέσει μαζί σε σωστές προτάσεις. Οι καλές γλώσσες προτύπων είναι παραγωγικές, ικανές να παράγουν όλες τις πιθανές προτάσεις από ένα πλούσιο και εκφραστικό λεξιλόγιο προτύπων.

Μια γλώσσα προτύπων διαμορφώνει μια τέλεια μορφή στην οποία κάθε ένα από τα πρότυπα του συνεργάζεται να λύσει ένα πιο θεμελιώδες πρόβλημα που δεν εξετάζεται ρητά από οποιοδήποτε μεμονωμένο πρότυπο. Αυτό βοηθά μια γλώσσα προτύπων να επιτύχει μια οργανική σειρά (organic order), την οποία ο Alexander περιγράφει στο Oregon ως «είδος διάταξης που επιτυγχάνεται όταν υπάρχει μια τέλεια ισορροπία μεταξύ των αναγκών των μερών και των αναγκών του συνόλου». Μια γλώσσα προτύπων είναι όπως ένα οικοσύστημα προτύπων, τα οποία συσχετίζονται σε κάποιο επίπεδο. Αυτή η «οικολογική ποιότητα» των γλωσσών προτύπων συμβάλλει «στην πληρότητά τους» και τη δυνατότητά τους να μας βοηθήσουν να παραγάγουμε τις «ζωντανές» αρχιτεκτονικές» κατέχοντας το QWAN.
Στο «The Timeless Way of Building» [10], ο Alexander παρατηρεί αυτό που μια γλώσσα προτύπων ενσωματώνει αληθινά:

- «Έτσι, όπως στην περίπτωση των φυσικών γλωσσών, η γλώσσα προτύπων είναι παραγωγική. Όχι μόνο μας λέει τους κανόνες της ρύθμισης, αλλά μας παρουσιάζει πώς να κατασκευάσει τις ρυθμίσεις, τόσες πολλές όσες θέλουμε οι οποίες ικανοποιούν τους κανόνες».
- «Μια γλώσσα προτύπων δίνει σε κάθε πρόσωπο που την χρησιμοποιεί, τη δύναμη να δημιουργήσει μια άπειρη ποικιλία νέων και μοναδικών κτισμάτων, ακριβώς όπως η συνηθισμένη γλώσσα του δίνει τη δύναμη να δημιουργήσει μια άπειρη ποικιλία προτάσεων».
- «Η δομή της γλώσσας δημιουργείται από το δίκτυο των συνδέσεων μεταξύ των μεμονωμένων προτύπων και οι γλώσσες υπάρχουν, στο βαθμό που αυτά τα πρότυπα διαμορφώνουν ένα σύνολο».
- «Κάθε πρότυπο, εξαρτάται και από τα μικρότερα πρότυπα που περιέχει και στα μεγαλύτερα πρότυπα μέσα στα οποία περιλαμβάνεται.»
- «Η γλώσσα είναι καλή, ικανή να κάνει κάτι ολόκληρο, όταν είναι μορφολογικά και λειτουργικά πλήρης».

Ο Michael Beedle, συντάκτης του «Reengineering the Application Development Process» [26], παρομοιάζει τα αποτελέσματα της χρησιμοποίησης των γλωσσών προτύπων με την παραγωγή των προκυπτουσών συμπεριφορών. Οι γλώσσες προτύπων παρέχουν μια δυναμική διαδικασία για την τακτική επίλυση των προβλημάτων μέσα στην περιοχή τους, που οδηγεί έμμεσα στην επίλυση ενός πολύ ευρύτερου προβλήματος. Τα πρότυπα και οι κανόνες σε μια γλώσσα προτύπων συνδυάζονται και διαμορφώνουν ένα αρχιτεκτονικό ύφος. Κατά αυτόν τον τρόπο, οι γλώσσες προτύπων καθοδηγούν τους αναλυτές συστημάτων, τους αρχιτέκτονες, τους σχεδιαστές, και τους εφαρμοστές για να παράγουν εφαρμόσιμα συστήματα που λύνουν τα κοινά οργανωτικά και προβλήματα ανάπτυξης σε όλα τα επίπεδα κλίμακας και ποικιλομορφίας.

1.16 Αποσπασματική αύξηση

Στο Oregon [5] ο Alexander περιγράφει πώς η γλώσσα προτύπων του και η αντίστοιχη μέθοδος «άχρονων τρόπων» εφαρμόστηκαν σε ένα πρόγραμμα προγραμματισμού για το πανεπιστήμιο του Όρεγκον. Εξηγεί την αρχή της οργανικής σειράς στην οποία «ο προγραμματισμός και η κατασκευή θα καθοδηγηθούν με μια διαδικασία που επιτρέπει στο σύνολο να προκύψει βαθμιαία από τις τοπικές πράξεις.» Μετά εξηγεί τη διαδικασία της αποσπασματικής αύξησης ως μια που είναι βασισμένη στην ιδέα της επισκευής σε αντιδιαστολή με την αντικατάσταση. Αντί να σχεδιάσει μια αρχιτεκτονική για να αντικαταστήσει μια υπάρχουσα, και που τελικά θα αντικατασταθεί, ο Alexander ορίζει μια πιο εξελικτική προσέγγιση που ξετυλίγει βαθμιαία μια πλήρη δομή από ένα αρχικό ίδρυμα από τη συνεχή τροποποίηση, τη βελτίωση και την επισκευή. Αναφέρεται στην προσέγγιση «προτύπου για αντικατάσταση» ως μεγάλη ανάπτυξη κομματιών και την συγκρίνει με την αποσπασματική αύξηση ως εξής:
Σύμφωνα με τη μεγάλη άποψη κομματιών, κάθε πράξη του προτύπου ή η οικοδόμηση είναι ένα απομονωμένο γεγονός που δημιουργεί ένα απομονωμένο κτίσμα. Σύμφωνα με την αποσπασματική άποψη, κάθε περιβάλλον αλλάζει και αυξάνεται όλη την ώρα, προκειμένου να κρατηθεί η χρήση του στην ισορροπία . Η ποιότητα του περιβάλλοντος είναι ένα είδος ημι-σταθερής ισορροπίας στη ροή του χρόνου. Η μεγάλη ανάπτυξη κομματιών είναι βασισμένη στην ιδέα της αντικατάστασης. Η αποσπασματική αύξηση είναι βασισμένη στην ιδέα της επισκευής.

Η σχέση μεταξύ των προτύπων και της αποσπασματικής αύξησης είναι ότι οι γλώσσες προτύπων προορίζονται να αυξήσουν και να εξελίξουν ολόκληρες αρχιτεκτονικές μέσω αυτής της διαδικασίας της αποσπασματικής αύξησης. Οι διάφορες ακολουθίες στις οποίες μια γλώσσα προτύπων καθοδηγεί το χρήστη στην εφαρμογή των προτύπων πρέπει εκεί μέσα να ξετυλίξουν μια πλήρη αρχιτεκτονική που προκύπτει βαθμιαία από τη διαδοχική εφαρμογή των μεμονωμένων προτύπων στην κατάλληλη σειρά.

1.17 Κατάλογοι και συστήματα προτύπων

Οι συντάκτες του POSA έχουν ταξινομήσει τα διαφορετικά είδη συλλογών προτύπων που κατέχουν ποικιλίες βαθμών της δομής και της αλληλεπίδρασης σε καταλόγους προτύπων, συστήματα και γλώσσες:

Κατάλογοι προτύπων
Ένας κατάλογος προτύπων είναι μια συλλογή των σχετικών προτύπων. Υποδιαιρεί χαρακτηριστικά τα πρότυπα τουλάχιστον σε έναν μικρό αριθμό ευρειών κατηγοριών και μπορεί να περιλάβει κάποιο ποσό παραπομπής-αναφοράς μεταξύ των προτύπων.

Συστήματα προτύπων
Ένα σύστημα προτύπων είναι ένα συνεκτικό σύνολο σχετικών προτύπων που λειτουργούν μαζί για να υποστηρίξουν την κατασκευή και την εξέλιξη ολόκληρων των αρχιτεκτονικών. Όχι μόνο οργανώνεται σε σχετικές ομάδες και σε υποομάδες σε πολλαπλά επίπεδα, αλλά περιγράφει τις πολλές αλληλεξαρτήσεις μεταξύ των προτύπων, των σχηματισμών ομάδας και πώς μπορούν να συνδυαστούν και να συντεθούν για να λύσουν τα πιο σύνθετα προβλήματα. Τα πρότυπα σε ένα σύστημα προτύπων πρέπει να περιγραφούν σε ένα συνεπές και ομοιόμορφο ύφος και πρέπει να καλύπτουν μια αρκετά ευρεία βάση των προβλημάτων και των λύσεων για να επιτρέψουν στα σημαντικά μερίδια των πλήρων αρχιτεκτονικών για να χτιστούν.

Ένας κατάλογος προτύπων προσθέτει ένα μικρό κομμάτι της δομής και της οργάνωσης σε μια συλλογή προτύπων, αλλά δεν πηγαίνει πολύ μακριά από την παρουσίαση μόνο της πιο φαινομενικώς ορατής δομής και σχέσεων. Ένα σύστημα προτύπων προσθέτει τη βαθιά δομή, την πλούσια αλληλεπίδραση προτύπων και την ομοιομορφία σε έναν κατάλογο προτύπων. Και τα συστήματα προτύπων και οι γλώσσες προτύπων διαμορφώνουν συνεπή σύνολα στενά αναμειγμένων προτύπων για την περιγραφή και την επίλυση των προβλημάτων σε μια ιδιαίτερη περιοχή. Αλλά μια γλώσσα προτύπων προσθέτει την ευρωστία, την περιεκτικότητα και την πληρότητα σε ένα σύστημα προτύπων. Η αρχική διαφορά είναι ότι οι γλώσσες προτύπων είναι υπολογιστικά πλήρεις, παρουσιάζοντας όλους τους πιθανούς συνδυασμούς προτύπων και παραλλαγές τους για να παραγάγουν τις πλήρεις αρχιτεκτονικές. Στην πράξη η διαφορά μεταξύ των συστημάτων προτύπων και των γλωσσών προτύπων μπορεί να είναι εξαιρετικά δύσκολο να εξακριβωθεί.

Ενώ ένα σύστημα προτύπων μπορεί να είναι μια συνεκτική συλλογή των προτύπων για ένα πολύ ευρύ θέμα, μια γλώσσα προτύπων πρέπει να είναι περισσότερο από ένα «ευρύ θέμα». Ένα σύστημα προτύπων δεν έχει απαραίτητα όλα αυτά τα στοιχεία προτύπων. Μπορεί να εστιάσει σε ένα εξίσου ευρύ ή στενό θέμα, αλλά μπορεί να μην

έχει απαραίτητα μια σαφή αποστολή, ή αφήνει διάφορα σημαντικά χάσματα ασυμπλήρωτα στο διάστημα προβλήματος .

Αλλά ίσως η σημαντικότερη διαφορά μεταξύ των γλωσσών προτύπων και των συστημάτων προτύπων είναι ότι οι γλώσσες προτύπων δεν δημιουργούνται εντελώς ξαφνικά. Εξελίσσονται από τα συστήματα προτύπων μέσω της διαδικασίας της αποσπασματικής αύξησης (και ένα σύστημα προτύπων μπορεί στη συνέχεια να εξελιχθεί από έναν κατάλογο προτύπων κατά τρόπο παρόμοιο). Όπως οι γλώσσες προτύπων βοηθούν να αυξηθούν αυξητικά ολόκληρες αρχιτεκτονικές, τα συστήματα προτύπων μπορεί να χρησιμεύσουν να αυξηθούν σε ολόκληρες γλώσσες προτύπων.

1.18 Γράφοντας πρότυπα

Το γράψιμο των καλών προτύπων είναι πολύ δύσκολο. Πρέπει όχι μόνο να παρέχουν τα γεγονότα, αλλά πρέπει επίσης να πουν μια ιστορία που συλλαμβάνει την εμπειρία που προσπαθούν να μεταβιβάσουν. Ένα πρότυπο πρέπει να βοηθήσει τους χρήστες του, να κατανοήσουν τα υπάρχοντα συστήματα, να προσαρμόσει τα συστήματα για να ανταποκριθούν στις ανάγκες των χρηστών και να κατασκευάσει νέα συστήματα. Η διαδικασία εύρεσης προτύπων καλείται pattern mining.

Ένα πρότυπο όταν συναντηθεί τυχαία δεν αναγνωρίζεται από την αρχή. Μπορεί να σημειωθούν οι αρχές μερικών πραγμάτων ότι είναι πρότυπα, αλλά μπορεί να αποδειχτεί ότι μερικά δεν είναι πρότυπα, ή είναι μόνο κομμάτια των προτύπων, ή μέθοδοι που μπορούν να αποτελέσουν μέρος της λογικής ενός συγκεκριμένου προτύπου. Μια λύση στην οποία καμία δύναμη δεν είναι παρούσα δεν είναι ένα πρότυπο.

Ο καλύτερος τρόπος να αναγνωριστούν και να τεκμηριωθούν τα χρήσιμα πρότυπα είναι με την εκμάθηση από άλλους που το έχουν κάνει καλά! Υπάρχουν διάφορα βιβλία και άρθρα που περιγράφουν τα πρότυπα και προσπαθούν να δουν εάν είναι δυνατό να αναγνωριστούν όλα τα απαραίτητα στοιχεία προτύπων και οι επιθυμητές ιδιότητες που αναφέρθηκαν νωρίτερα.

Η περαιτέρω βοήθεια για εκείνους που είναι αρκετά θαρραλέοι να αναλάβουν το γράψιμο ενός προτύπου, ή μια γλώσσα προτύπων, μπορεί να βρεθεί στο βιβλίο του Ward Cunningham «Tips for Writing Pattern Languages» [28], και των Gerard Meszaros' και Jim Doble's «A Pattern Language for Pattern Writing» [29]. Και τα δύο έγγραφα είναι απαραίτητοι πόροι για το γράψιμο προτύπων και των γλωσσών προτύπων.

Πώς οι εμπειρογνώμονες αποφασίζουν τι κάνει ένα καλό πρότυπο; Οι διασκέψεις PLoP έχουν διάφορα κριτήρια που υποστηρίζουν ότι τα έγγραφα προτύπων πρέπει να ακολουθούν:
1). Εστίαση στη δυνατότητα πραγματοποίησης: Τα πρότυπα πρέπει να περιγράψουν τις αποδεδειγμένες λύσεις στα επαναλαμβανόμενα προβλήματα παρά τα πιο πρόσφατα επιστημονικά αποτελέσματα.
2). Επιθετική αμέλεια της πρωτοτυπίας: Οι συγγραφείς προτύπων δεν πρέπει να είναι οι αρχικοί εφευρέτες των λύσεων που τεκμηριώνουν.
3). Μη-ανώνυμη αναθεώρηση: Ο συγγραφέας έρχεται σε επαφή με το συντάκτη προτύπων και συζητά πώς τα πρότυπα μπορούν να διευκρινιστούν ή να βελτιωθούν.

4). Εργαστήρια συγγραφέα αντί παρουσιάσεων: Αντί για την παρουσίαση από τους μεμονωμένους συντάκτες, τα πρότυπα συζητούνται στα εργαστήρια του συγγραφέα. Ένα ανοικτό φόρουμ όπου όλοι που παρευρίσκονται επιδιώκουν να βελτιώσουν τα πρότυπα που παρουσιάζονται με τη συζήτηση, τι συμπαθούν για τα πρότυπά τους καθώς επίσης και άλλες περιοχές στα οποία λείπουν.
5). Προσεκτική έκδοση: Οι συντάκτες προτύπων έχουν την ευκαιρία να ενσωματώσουν όλα τα σχόλια και τις ιδέες κατά τη διάρκεια των εργαστηρίων του συγγραφέα πριν παρουσιάζουν τα πρότυπα με την ολοκληρωμένη τους μορφή.

1.19 Το μέλλον των προτύπων

Η τρέχουσα δημοτικότητα των προτύπων λογισμικού έχει κεντρίσει τις πολυάριθμες δραστηριότητες για να διευρύνει τη χρήση και την υποστήριξή τους εντός της κοινότητας ανάπτυξης λογισμικού. Υπάρχουν ομάδες ανθρώπων που χρησιμοποιούν τα πρότυπα για να τεκμηριώσουν τις διαδικασίες ανάπτυξης λογισμικού υπό μορφή γλώσσας προτύπων. Αρκετές από τις κύριες αντικειμενοστρεφείς μεθόδους σχεδιασμού λογισμικού έχουν προσθέσει την υποστήριξη για τη διαμόρφωση και την αναπαράσταση των σχεδιαστικών προτύπων. Πολλά εργαλεία ανάπτυξης λογισμικού έχουν προσθέσει την παρόμοια υποστήριξη. Μερικά ερευνητικά προγράμματα προσπαθούν να κωδικοποιήσουν τυπικά τα σχεδιαστικά πρότυπα για τους σκοπούς της παραγωγής του κώδικα πηγής. Οι εμπορικές βιβλιοθήκες λογισμικού αναπτύσσονται, που παρέχουν τις επαναχρησιμοποιήσιμες εφαρμογές γνωστών σχεδιαστικών προτύπων (η JAVA παρέχει μερικούς από αυτές στην βιβλιοθήκη της). Μπορεί να μην είναι πολύ πριν να εισαγάγουν μερικές γλώσσες προγραμματισμού τη πρόσθετη σύνταξη για την αναπαράσταση των σχεδιαστικών προτύπων ως ρητά κατασκευάσματα προγραμματισμού.

Υπάρχει υποψία ότι τα πρότυπα μια ημέρα θα αντικαταστήσουν τους προγραμματιστές υπολογιστών. Αυτή η άποψη έχει εμφανιστεί πολλές φορές στο παρελθόν σχετικά με τις διάφορες άλλες «νέες» τεχνολογίες. Οι γλώσσες και τα εργαλεία που θα χρησιμοποιηθούν για να λύσουν τα προβλήματα λογισμικού, θα μπορούν στο μέλλον να προωθήσουν αρκετά πέρα από αυτά που είναι προς το παρόν διαθέσιμα. Αλλά θα απαιτήσουν υπεύθυνους για την ανάπτυξη με εξελιγμένες δεξιότητες και τη σοφία για να τις χρησιμοποιήσουν αποτελεσματικά. Όσο συμβαδίζουν οι προγραμματιστές λογισμικού με τις νέες έννοιες και τις τεχνολογίες λογισμικού, θα υπάρχει πάντα μια ανάγκη για να κατασκευάσουν τις χρήσιμες λύσεις λογισμικού που χρησιμοποιούν τα νέα εργαλεία και τις καινούριες γλώσσες.

Ενώ η δυνατότητα να κωδικοποιηθούν τα πρότυπα ως γενικά τμήματα λογισμικού μπορεί να είναι σημαντική, πιο σημαντική είναι η γνώση για το πώς και πότε θα συνδυαστούν τα πρότυπα από κοινού, με τη δυνατότητα να χρησιμοποιηθεί ένα κοινό λεξιλόγιο. Επειδή τα πρότυπα συλλαμβάνουν τη γνώση που προορίζεται για τους ανθρώπους, είναι ο κοινωνικός αντίκτυπος των προτύπων που διαμορφώνει κατά ένα μεγάλο μέρος τον τεχνολογικό αντίκτυπό τους.

Οι εργασίες του Christopher Alexander στα πρότυπα και τις γλώσσες προτύπων είχαν ολοκληρωθεί περισσότερο από δύο δεκαετίες προτού να γίνουν δημοφιλείς εντός της κοινότητας λογισμικού. Κατά τη διάρκεια αυτών των λίγων δεκαετιών, ο Alexander είχε εργαστεί σε ένα βιβλίο με τίτλο «The Nature of Order» [30] που προωθεί

σημαντικά τις προηγούμενες ιδέες του για τα πρότυπα, την αρχιτεκτονική και την ομορφιά.

1.20 Τελικές παρατηρήσεις

Οι μεγάλοι μηχανικοί όχι μόνο σχεδιάζουν τα προϊόντα τους αυστηρά σύμφωνα με τις αρχές της επιστήμης. Πρέπει να προσαρμόσουν τις λύσεις τους για να κάνουν τις βέλτιστες ανταλλαγές και τους συμβιβασμούς μεταξύ των γνωστών λύσεων, των αρχών, και των περιορισμών για να ικανοποιήσουν τις συνεχώς αυξανόμενες και συνεχώς μεταβαλλόμενες απαιτήσεις του κόστους, του προγράμματος, της ποιότητας, και των αναγκών των πελατών. Η βοήθεια προτύπων είναι πολύ σημαντική.

Τα πρότυπα αναπαριστούν την εμπειρία που, μέσω της αφομοίωσής τους, μεταβιβάζουν οι ειδικοί στους μη ειδικούς υπεύθυνους για την ανάπτυξη. Βοηθούν να δημιουργηθεί ένα κτίσμα σύμφωνα με ένα κοινό αρχιτεκτονικό όραμα. Εάν πρέπει να υπάρχει εξέλιξη της ανάπτυξης λογισμικού, πρέπει αυτές οι αποδεδειγμένες «καλύτερες πρακτικές» και «τα παθήματα που γίνονται μαθήματα» να τεκμηριωθούν, να συνταχθούν, να διερευνηθούν και να διαδοθούν ευρέως ως πρότυπα (και αντι-πρότυπα). Μόλις εκφραστεί μια λύση με μορφή προτύπων, μπορεί έπειτα να εφαρμοστεί και σε άλλα πλαίσια και να διευκολύνει με την επαναχρησιμοποίηση της.

Τα πρότυπα είναι εξαιρετικά πολύτιμα εργαλεία για την επίκτητη γνώση και εμπειρία, για να βελτιώσουν την ποιότητα και την παραγωγικότητα λογισμικού με το να αντιμετωπίσουν τα θεμελιώδη ζητήματα στην ανάπτυξη του λογισμικού. Με τη χρησιμοποίηση αυτών των εργαλείων μπορούν να αντιμετωπιστούν καλύτερα οι προκλήσεις.

Τα πρότυπα εκθέτουν τη γνώση για την κατασκευή λογισμικού που έχουν κερδίσει πολλοί εμπειρογνώμονες κατά τη διάρκεια πολλών ετών. Όλη η εργασία για τα πρότυπα πρέπει επομένως να εστιάσει, στο να γίνει αυτός ο πολύτιμος πόρος ευρέως διαθέσιμος. Κάθε προγραμματιστής λογισμικού πρέπει να είναι σε θέση να χρησιμοποιήσει τα πρότυπα αποτελεσματικά όταν φτιάχνει συστήματα λογισμικού. Όταν αυτό επιτευχθεί, θα είμαστε σε θέση να γιορτάσουμε την ανθρώπινη νοημοσύνη που απεικονίζουν τα πρότυπα και σε κάθε μεμονωμένο πρότυπο και σε όλα τα πρότυπα μαζί.

Αναφορές :

[1]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, (1994), *Design Patterns: Elements of Reusable Object-Oriented Software*.
[2]: Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, (1996), *Pattern-Oriented Software Architecture: A System of Patterns*, Wiley.
[3]: John M. Vlissides, James O. Coplien, Norman L. Kerth , (1996), *Pattern Languages of Program Design and Pattern Languages of Program Design 2*. Addison Wesley Professional.
[4]: Christopher Alexander , (1964), *Notes on the Synthesis of Form*, Harvard University Press.
[5]: Christopher Alexander, (1975), *The Oregon Experiment*, Oxford University Press.
[6]: Christopher Alexander, (1977), *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press.
[7]: Christopher Alexander, (1979), *The Timeless Way of Building*, Oxford University Press.
[8]: http://c2.com/doc/oopsla87.html

[9]: http://www.iterasi.net/openviewer.aspx?sqrlitid=utu4f_7qgum2ts0hd5bbvq
[10]: Christopher Alexander, (1979), *The Timeless Way of Building* , Oxford University Press.
[11]: James Coplien, (1991), *Advanced C++ Programming Styles and Idioms*, Addison-Wesley Professional.
[12]: http://www.laputan.org/reflection/addendum.html
[13]: http://en.wikipedia.org/wiki/Pattern_Languages_of_Programs
[14]: Dirk Riehle and Heinz Zullighoven, (1996), *Understanding and Using Patterns in Software Development*, John Wiley & Sons.
[15]: Richard Gabriel, (1996), *Patterns of Software: Tales From the Software Community*, Oxford University Press.
[16]: Jim Coplien, (1995), *Pattern Languages of Program Design (Software Patterns Series)*, Addison-Wesley Professional, pp 576.
[17]: http://www.cs.wustl.edu/~schmidt/POSA/
[18]: F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad, M. Stal, (1996), *Pattern-Oriented Software Architecture A System of Patterns*, John Wiley and Sons Ltd.
[19]: Douglas C. Schmidt Michael Stal, Hans Rohnert and Frank Buschmann, (2000), *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Published by Wiley & Sons in 2000.
[20]: Michael Kircher and Prashant Jain, (2004), *Pattern-Oriented Software Architecture, Volume 3: Patterns for Resource Management*, Wiley.
[21]: Frank Buschmann, Kevlin Henney Douglas C. Schmidt, (2007), *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing* , Wiley.
[22]: Frank Buschmann, Kevlin Henney Douglas C. Schmidt, (2007), *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*, Wiley.
[23]: http://g.oswego.edu/dl/ca/ca/ca.html
[24]: Doug Lea, (1994), *Christopher Alexander: an introduction for object-oriented designers*, Published ACM SIGSOFT Software Engineering Notes,Volume 19, Issue 1 (January 1994), pp. 39 – 46.
[25]: Nikos Salingaros, (1995), *The Laws of Architecture from a Physicist's Perspective*, Physics Essays **8** (1995), pp. 638-643.
[26]: Michael A. Beedle, (1997), *Reengineering the Application Development Process*, Business Object Design and Implementation III: Patterns, Workflow, Components, and the Web Workshop Agenda, 6 October 1997, Atlanta,Georgia.
[27]: Linda Rising, (1998), *Software Design Patterns: Common Questions and Answers*, Cambridge University Press 1998.
[28]: Ward Cunningham's "Tips for Writing Pattern Languages" on the WikiWiki Web http://c2.com/cgi/wiki?TipsForWritingPatternLanguages
[29]: Gerard Meszaros and Jim Doble, (1997), *A pattern language for pattern writing*, Publisher Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
[30]: Christopher Alexander, (2004), *The Phenomenon of Life: Nature of Order, Book 1: An Essay on the Art of Building and the Nature of the Universe*, Center for Environmental Structure.

2. ΘΕΜΕΛΙΩΔΗ ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ
Fundamental Design Patterns

2.1 Σύντομη αναφορά των θεμελιωδών σχεδιαστικών προτύπων.

Τρεις από τις ιδιότητες της αντικειμενοστρεφούς τεχνολογίας λογισμικού: η κληρονομικότητα (inheritance), ο πολυμορφισμός (polymorphism) και η ενθυλάκωση (encapsulation) θεωρούνται ότι συνιστούν την βάση επί της οποίας μπορεί να

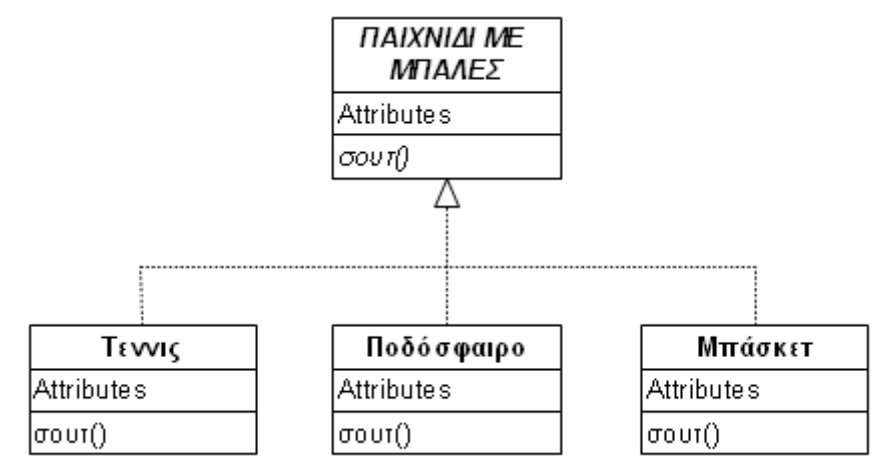
θεμελιωθεί οποιαδήποτε αντικειμενοστρεφής σχεδίαση (design) και ανάπτυξη (implementation) λογισμικού.
Σχεδιαστικά πρότυπα που φέρουν τις ονομασίες: interface (διεπαφή), abstract class (αφηρημένη κλάση), abstract super-class (αφηρημένη υπερ-κλάση ή πατρική-κλάση), εφαρμόζουν την ιδιότητα της κληρονομικότητας και παρέχουν τρόπους με τους οποίους μπορούν να οργανώνονται, κατά κληρονομικό τρόπο, οι σχέσεις μεταξύ των κλάσεων αντικειμένων (classes of objects) στους εκάστοτε σχεδιασμούς αντικειμενοστρεφούς λογισμικού (object oriented software).
Σε περιπτώσεις που η κατά κληρονομικό τρόπο οργάνωση των αντικειμένων του υπό σχεδίαση λογισμικού δεν είναι η καταλληλότερη, εξετάζεται η δυνατότητα εφαρμογής ενός διαφορετικού προτύπου που ονομάζεται delegation (αντιπροσώπευση). Πολύ συχνά, εκεί που δεν είναι δυνατόν να εφαρμοστούν τα σχεδιαστικά πρότυπα κληρονομικότητας, εφαρμόζεται η αντιπροσώπευση. Πολλοί σχεδιασμοί αντικειμενοστρεφούς λογισμικού (object oriented software designs) χρησιμοποιούν τουλάχιστον ένα από τα παραπάνω πρότυπα. Η χρήση τους είναι τόσο συχνή που πολλές φορές παραλείπεται η μνεία τους στις σχετικές τεκμηριώσεις των περισσότερων σχεδιασμών.
Το πρότυπο immutable (μη μεταβλητότητας - αμεταβλησίας) βρίσκει εφαρμογή σε σχεδιασμούς λογισμικού στους οποίους το ζητούμενο είναι η διατήρηση της κατάστασης ενός αντικειμένου (μη αλλαγή των τιμών των μεταβλητών του). Τέτοια περίπτωση είναι ο σχεδιασμός συστήματος στο οποίο πολλά αντικείμενα απεικονίζονται να επιχειρούν ταυτόχρονη πρόσβαση σε ένα μόνο αντικείμενο δια μέσου πολλαπλών διεργασιών (processes) ή νημάτων (threads). Η εφαρμογή του προτύπου αυτού παρέχει την δυνατότητα εξασφάλισης την αμεταβλησίας της κατάστασης του εν λόγω αντικειμένου σε περίπτωση αποτυχίας ή αποφυγής της εφαρμογής της αρχής του αμοιβαίου αποκλεισμού (mutual exclusion).
Το πρότυπο marker interface (διεπαφή σήμανσης – αναγνώρισης) παρέχει έναν τρόπο να απλοποιηθεί ο σχεδιασμός κλάσεων που έχουν κάποια σταθερή ιδιότητα και είναι επιθυμητό να αναγνωρίζεται άμεσα αυτή. Πρόκειται για εφαρμογή της ιδιότητας της κληρονομικότητας με σκοπό την ικανότητα αναγνώρισης όλων των μελών μιας ολόκληρης ομάδας κλάσεων που υλοποιούν (implement) και επομένως κληρονομούν μία διεπαφή - σήμανση, που χαρακτηρίζει για αυτόν τον λόγο όλη την ιεραρχία.
Το πρότυπο proxy (εξουσιοδότησης) εφαρμόζεται σε διάφορους σχεδιασμούς λογισμικού, στους οποίους απαιτείται ένα αντικείμενο να ενεργεί ως ενδιάμεσο προκειμένου να επιτυγχάνεται η πρόσβαση ενός άλλου αντικείμενου σε ένα τρίτο.
Τα Θεμελιώδη Σχεδιαστικά πρότυπα χρησιμοποιούνται για την ανάπτυξη λογισμικού διαφόρων πεδίων των επιστημών, όπως Ιατρική και Βιολογία. Χρησιμοποιούνται σε πολλές εφαρμογές και έχουν διάφορες χρήσεις.
Όπως αναφέρει μια εργασία σχετικά με τη Γενετική ‘‘Η ανάλυση των στοιχείων έκφρασης γονιδίων DNA έχει αποκαλύψει βαθύτερα πρότυπα. Αυτά τα πρότυπα συμβάλλουν ανομοιόμορφα στη δομή των σχεδιαγραμμάτων έκφρασης. Επιπλέον, τα χαρακτηριστικά γνωρίσματα ενός δεδομένου συνόλου σχεδιαγραμμάτων έκφρασης μπορούν να κατανοηθούν χρησιμοποιώντας έναν μικρό αριθμό χαρακτηριστικών τρόπων. Αυτό οδηγεί στο εντυπωσιακό συμπέρασμα ότι η μεταγραφή ενός γονιδιώματος είναι ενορχηστρωμένη σε μερικά θεμελιώδη πρότυπα της αλλαγής έκφρασης γονιδίων. Αυτά τα πρότυπα είναι απλά και εξουσιάζοντας τις αλλαγές στην έκφραση των γονιδίων σε όλο το γονιδίωμα.’’[21].
Στο Πανεπιστήμιο Καλιφόρνιας στο τμήμα Κινησιολογίας παρουσιάστηκε μια διάλεξη για να αναπτύξει τη λογική μορφή στα θεμελιώδη πρότυπα κίνησης. Οι σπουδαστές θα αναπτύξουν την ικανότητα στη γνώση και την κατανόηση των τρόπων

με τους οποίους οι δεξιότητες μηχανών μπορούν να οργανωθούν στις εξελικτικά κατάλληλες προόδους σε διάφορες τοποθετήσεις μετακίνησης συμπεριλαμβανομένων των εκπαιδευτικών παιχνιδιών, της γυμναστικής και του χορού [23].
Τα Θεμελιώδη Σχεδιαστικά πρότυπα έχουν χρησιμοποιηθεί για να δημιουργηθούν καινούρια πρότυπα. Ένα παράδειγμα είναι τα Θεμελιώδη πρότυπα Πολιτιστικών Διαφορών, στα οποία αναφέρεται ότι οι πρακτικές επικοινωνίας διαφέρουν μεταξύ τους και μέσα στον κάθε πολιτισμό. Αναγνωρίζοντας πρότυπα σχετικά με τον πολιτισμό και την επικοινωνία, τα άτομα μπορούν να γνωρίζουν τις αντιδράσεις των ατόμων από τους άλλους πολιτισμούς και να αλληλεπιδράσουν πιο αποτελεσματικά με αυτούς [22].
Σε συνέδριο που έγινε το 2004 στην Ελβετία στην παρουσίαση ‘‘Μερικές σκέψεις στη θεμελιώδη διανομή των εταιριών’’ αναφέρθηκαν τα εξής:
‘‘Υπάρχουν μερικά συγκριτικά θεμελιώδη πρότυπα συμπεριφοράς μεταξύ του επιχειρησιακού πληθυσμού, όπου τα πρότυπα των εταιριών που βασίστηκαν στο μέγεθός τους φαίνεται να διορθώθηκαν. Αυτές οι σχέσεις έχουν αγνοηθεί συχνά στην υπάρχουσα ερευνητική λογοτεχνία μικρών επιχειρήσεων, αλλά μπορούν να παίξουν έναν πολύτιμο ρόλο στην ανάπτυξη των αρχών που κυβερνούν την κανονική διανομή των εταιριών σε πολλές εθνικές οικονομίες.’’ [26].
Σε άλλες εργασίες , όχι μόνο παρουσιάζονται νέα πρότυπα, αλλά γίνονται αντικείμενο επεξεργασίας και αναπτύσσονται. Τα θεμελιώδη πρότυπα συχνότητας (F0) απεικονίζουν τις αλλαγές ανάπτυξης της προφορικής γλώσσας, η F0 η ταξινόμηση σχεδίων είναι μια σημαντική τεχνική για τη μελέτη από τη λεκτική ανάλυση [24].
Σε μια άλλη εργασία προτείνονται τρία θεμελιώδη πρότυπα που δομούν την οντολογία REA και χρησιμεύουν ως η ελάχιστη απαραίτητη σημασιολογία για την άλγεβρα που θα αναπτυχθεί. [25] .

2.2 Διεπαφή (Interface)

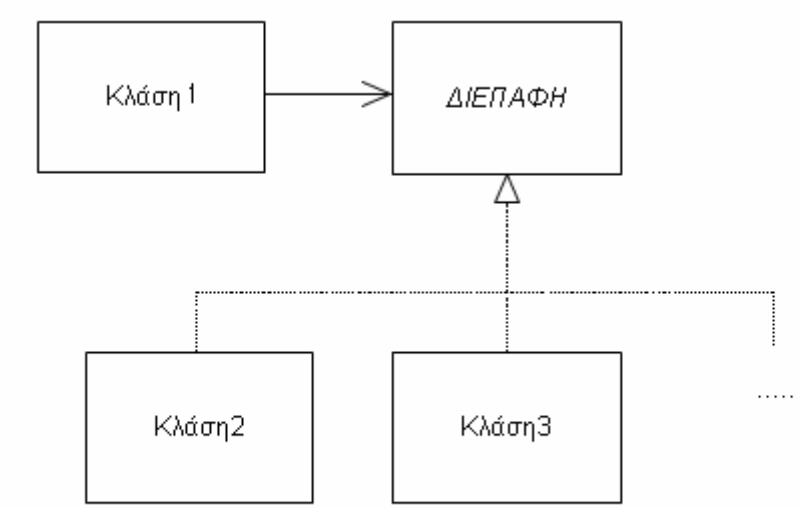
Η οντότητα ορίζεται ως κάτι το οποίο έχει ξεκάθαρη και χωριστή ύπαρξη, αν και δεν είναι αναγκαίο η ύπαρξη αυτή να είναι υλική. Οι οντότητες χρησιμοποιούνται σε συστήματα ανάπτυξης μοντέλων που δείχνουν την επικοινωνία και την εσωτερική λειτουργία [1].
Γενικά η Διεπαφή αναφέρεται σε μια αφαίρεση της οντότητας ,που προσφέρει η οντότητα στο χώρο έξω από αυτή, δηλαδή σε άλλες οντότητες. Με αυτόν τον τρόπο διαχωρίζεται η εσωτερική διεργασία από τις μεθόδους τις εξωτερικής επικοινωνίας. Μπορεί να παρέχει πολλές αφαιρέσεις της οντότητας. Έτσι επιτρέπεται να τροποποιηθεί εσωτερικά, χωρίς να επηρεάζονται οι οντότητες που επικοινωνούν με τη συγκεκριμένη οντότητα [2]. Επιπρόσθετα, μπορεί να προσφέρει τρόπο μετάφρασης μεταξύ οντοτήτων, οι οποίες δεν μιλούν την ίδια γλώσσα. Ένα παράδειγμα είναι η Διεπαφή Χρήστη (User Interface), η οποία βοηθά στην επικοινωνία μεταξύ του υπολογιστή και του ανθρώπου που χειρίζεται τον υπολογιστή [3]. Η διεπαφή περιγράφει το σύνολο των αιτημάτων στο οποίο ένα αντικείμενο μπορεί να ανταποκριθεί [28].
Η Διεπαφη γενικά είναι μια κλάση της οποίας οι μέθοδοι δεν είναι υλοποιημένες. Η διεπαφή συνδέεται με κληρονομικότητα με μια ή περισσότερες κλάσεις. Οι κλάσεις αυτές περιέχουν τις μεθόδους που περιέχει και η διεπαφή. Όμως, κάθε κλάση έχει έναν στόχο διαφορετικό από τις άλλες κλάσεις. Επομένως, καλώντας μια μέθοδο μιας κλάσης, επιστρέφει διαφορετικό αποτέλεσμα από την ίδια μέθοδο μιας άλλης κλάσης.
Για να γίνει κατανοητή η διεπαφή θα παρουσιαστεί ένα παράδειγμα. Έστω ότι υπάρχει ένα βίντεο-παιχνίδι, που ονομάζεται παιχνίδι με μπάλες, το οποίο χωρίζεται

σε τρεις υποκατηγορίες. Αυτές είναι τέννις, ποδόσφαιρο και μπάσκετ. Η διεπαφή παιχνίδι με μπάλες περιέχει μια μέθοδο που ονομάζεται σουτ. Η μέθοδος σουτ θα περιέχεται και στις κλάσεις Τέννις, Πδόσφαιρο και Μπάσκετ. Αυτή η οργάνωση απεικονίζεται στο σχήμα 2.2.1
Ανάλογα σε ποιο παιχνίδι βρίσκεται ο χρήστης, η μέθοδος σουτ επιστρέφει διαφορετικό αποτέλεσμα γραφικών. Αν ο χρήστης παίζει το παιχνίδι τέννις, τότε με την μέθοδο σουτ θα φαίνεται στην οθόνη ένας αθλητής που θα κρατάει μια ρακέτα και θα χτυπάει ένα μπαλάκι. Αν ο χρήστης παίζει ποδόσφαιρο, τότε η μέθοδος σουτ πρέπει να επιστρέφει ένα αποτέλεσμα το οποίο θα δείχνει έναν αθλητή που να κλοτσάει μια μπάλα ποδοσφαίρου. Ομοίως, όταν ο χρήστης παίζει μπάσκετ, η μέθοδος σουτ επιστρέφει ένα αποτέλεσμα το οποίο θα δείχνει έναν αθλητή που να διώχνει την μπάλα μπάσκετ με τα χέρια του.



Σχήμα 2.2.1 Απεικονίζεται το διάγραμμα κλάσεων της διεπαφής για το παιχνίδι με μπάλες.

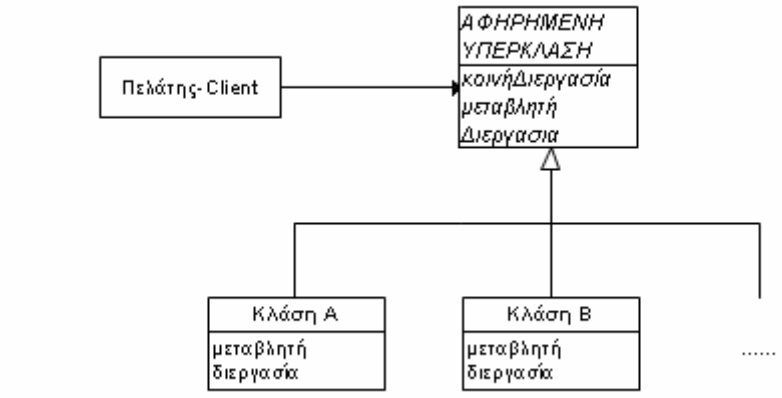
Το Πρότυπο Διεπαφής χρησιμοποιείται σε περιπτώσεις όπου υπάρχει μια κλάση, η οποία επικοινωνεί με άλλες κλάσεις. Όμως η αρχική κλάση πρέπει να είναι ανεξάρτητη από τις υπόλοιπες κλάσεις. Οι υπόλοιπες κλάσεις παρέχουν δεδομένα ή και υπηρεσίες. Η ανεξαρτησία αυτή είναι πολύ σημαντική, διότι αν αλλαχτούν οι υπόλοιπες κλάσεις , πρέπει ο αντίκτυπος στην αρχική κλάση να είναι όσο το δυνατόν μικρότερος. Επομένως , ο στόχος μπορεί να επιτευχθεί, επιτρέποντας άλλες κλάσεις να έχουν πρόσβαση στα δεδομένα ή και τις υπηρεσίες, μέσω μιας διεπαφής [4]. Διεπαφή είναι μια κλάση της οποίας οι μέθοδοι δεν είναι υλοποιημένες.
Η κλάση 1 παίρνει τις πληροφορίες που χρειάζεται έμμεσα από τις κλάσεις 2 και 3 μέσω της διεπαφής. Αν δεν χρησιμοποιούταν η διεπαφή και επικοινωνούσαν άμεσα οι δυο κλάσεις, τότε οι αλλαγές που θα γινόταν στην δεύτερη κλάση θα επηρέαζαν την πρώτη κλάση, πράγμα το οποίο δεν είναι επιθυμητό.
Στο σχήμα 2.2.2 απεικονίζεται το διάγραμμα κλάσεων του πρότυπου διεπαφής.



Σχήμα 2.2.2 : Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου διεπαφής.

2.3 Αφηρημένη Υπερ-κλάση (Abstract Superclass)

Το πρότυπο της Αφηρημένης υπέρ-κλάσης χρησιμοποιείται όταν ορισμένες κλάσεις έχουν κοινή διεπαφή ή οι διεπαφές τους συμπίπτουν σε μια κοινή διεπαφή. Επίσης πρέπει η κοινή διεπαφή να καθοριστεί σε μια θέση έτσι ώστε όλες οι κλάσεις να ξέρουν ότι πρέπει να παραμείνουν σε αυτήν την διεπαφή .Με αυτόν τον τρόπο ο πελάτης γνωρίζει ποια διεπαφή να αναμείνει [5].
Το πρότυπο αυτό ορίζει μια αφηρημένη υπερ-κλάση (AbstractSuperclass), η οποία καθορίζει μια ή περισσότερες λειτουργίες που θα γίνουν σε μια ή περισσότερες κλάσεις. Αυτές τώρα οι κλάσεις κληρονομούν και ορισμένα χαρακτηριστικά από την αφηρημένη υπέρ-κλάση. Δημιουργείται η αφηρημένη υπέρ-κλάση και ορισμένες κλάσεις, που συνδέονται με κληρονομικότητα, όπως φαίνεται στο παρακάτω σχήμα.
Στο σχήμα 2.3.1 απεικονίζεται το διάγραμμα κλάσεων του πρότυπου αφηρημένης υπέρ-κλάσης.

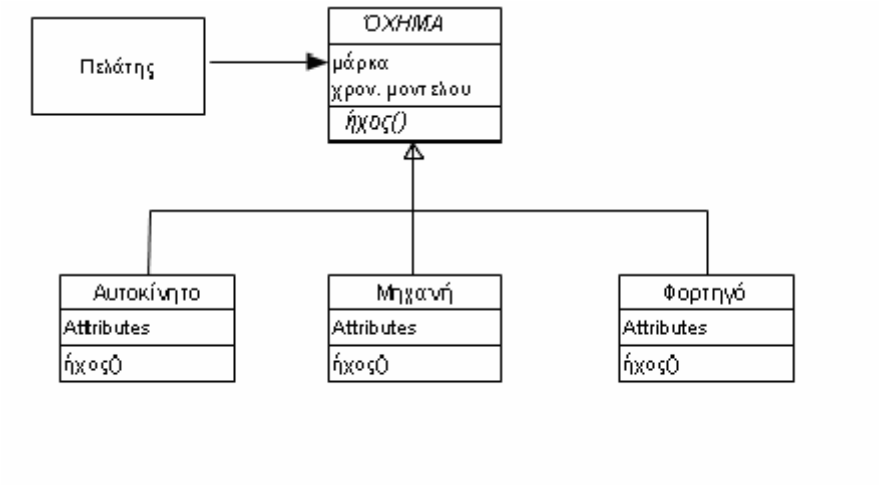


Σχήμα 2.3.1 : Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου αφηρημένης υπέρ-κλάσης.

Η Αφηρημένη υπέρ-κλάση καθορίζει τη διεπαφή που μοιράζονται οι Κλάση Α, Κλάση Β κτλ. Επίσης οι Κλάση Α, Κλάση Β κτλ, εφαρμόζουν την διεπαφή που κληρονομείται από την Αφηρημένη υπερκλάση [5].
Ορισμένες συναρτήσεις- μέθοδοι είναι κοινές και για την αφηρημένη υπέρ-κλάση και για τις υπόλοιπες κλάσεις, και καταγράφονται στην αφηρημένη υπέρ-κλάση. Κάποιες άλλες συναρτήσεις είναι διαφορετικές για τις υπόλοιπες κλάσεις. Οι συναρτήσεις αυτές θα υλοποιηθούν σε αυτές τις κλάσεις, αλλά η κάθε μια θα δώσει το δικό της αποτέλεσμα. Στην αφηρημένη υπέρ-κλάση καταγράφονται ως αφηρημένες συναρτήσεις.

Σε αντίθεση με μια διεπαφή, μια αφηρημένη υπέρ-κλάση, είναι μια εφαρμογή που καθορίζει μερικώς κοινή συμπεριφορά των υποκλάσεών της. Συνεπώς, η διεπαφή και η αφηρημένη Superclass μπορεί να εμφανίζονται μαζί αλλά είναι διαφορετικά πρότυπα [2].

Πιο κάτω παρουσιάζεται ένα παράδειγμα του προτύπου αφηρημένης υπέρ-κλάσης. Στο παράδειγμα αυτό, υπάρχει ο Πελάτης, η κλάση με όνομα Όχημα και τρεις κλάσεις με ονόματα Αυτοκίνητο, Μηχανή και Φορτηγό. Η κλάση Όχημα είναι αφηρημένη υπέρ-κλάση για το Αυτοκίνητο, τη Μηχανή και το Φορτηγό. Το Όχημα έχει ορισμένα χαρακτηριστικά γνωρίσματα, την μάρκα και την χρονολογία μοντέλου του Οχήματος. Το Όχημα έχει επίσης και μια λειτουργία με όνομα ήχος. Δηλαδή, είναι ο ήχος που παράγει κάθε όχημα. Ο πελάτης ανάλογα με το τι ζητήσει, γνωρίζει ποια διεπαφή να αναμείνει. Αν ζητήσει το αυτοκίνητο, τότε θα έχει τη μάρκα του οχήματος, τη χρονολογία του μοντέλου και τον ήχο ενός αυτοκινήτου.



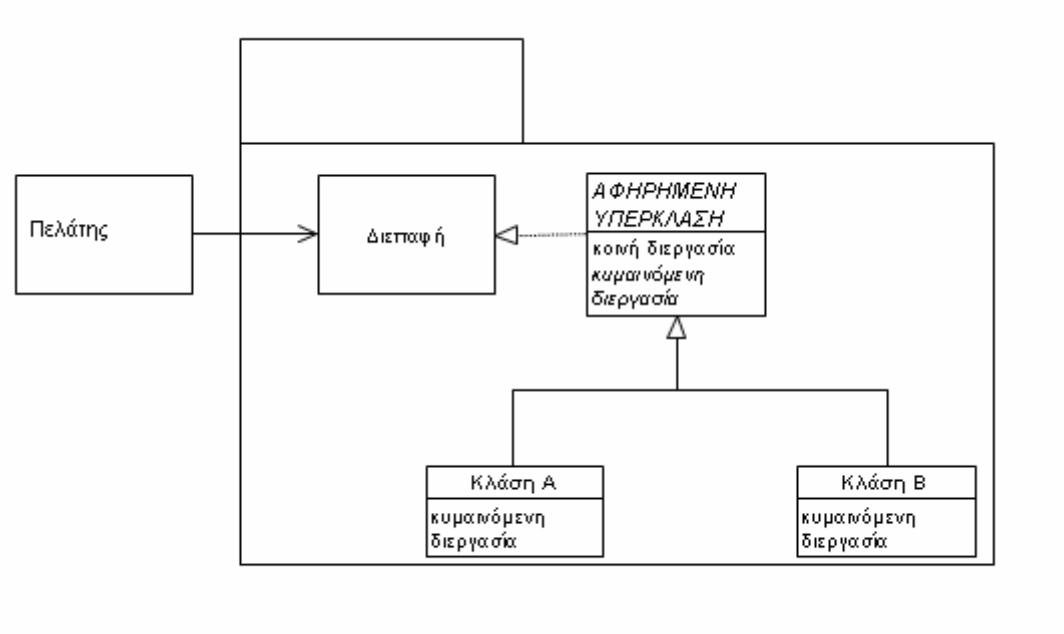
Σχήμα 2.3.2 : Απεικονίζεται ένα παράδειγμα του προτύπου αφηρημένης υπέρ-κλάσης.

Στην εργασία του Walter L. Hursch [38] τίθεται το ερώτημα εάν η υπέρ-κλάση πρέπει να είναι αφηρημένη. Επίσης εξετάζεται η σχέση μεταξύ κληρονομικότητας και αφαίρεσης.
Το πρότυπο αυτό πολλές φορές επεκτείνεται, όπως φαίνεται στο πρότυπο αφηρημένης συσκευής. Όπως φαίνεται και στην εργασία με τίτλο ‘‘ Abstract Device Pattern and Tango ’’ το αφηρημένο πρότυπο μπορεί να εφαρμοστεί σε Συσκευές για να εφαρμόσει τις τυποποιημένες διεπαφές για τις οικογένειες των συσκευών, να

εξετάσει την αφαίρεση και να βελτιώσει γενικά τη διαλειτουργικότητα των τμημάτων πελατών και κεντρικών υπολογιστών ανάμεσα στα συστήματα ελέγχου [27].

2.4 Interface and Abstract Class

Το πρότυπο αυτό είναι ένας συνδυασμός του Interface pattern και του Abstract Superclass pattern. Χρησιμοποιείται για να συνδυάζονται τα οφέλη των δύο αυτών προτύπων. Δηλαδή σε μια περίπτωση να χρησιμοποιούνται τα οφέλη της διεπαφής και σε άλλη αυτά της αφηρημένης κλάσης. Η διαφορά που έχουν μεταξύ τους είναι ότι μια αφηρημένη κλάση μπορεί να μην εφαρμόσει όλες τις συναρτήσεις της, κάτι το οποίο δεν μπορεί να κάνει η διεπαφή.
1). Όταν δεν πρέπει ο πελάτης να δει την κλάση από την οποία δημιουργείται το αντικείμενο, το οποίο παρέχει μια υπηρεσία, τότε χρησιμοποιείται το πρότυπο διεπαφής. Τα αντικείμενα του πελάτη μπορούν μέσω διεπαφής, έμμεσα δηλαδή, να έχουν πρόσβαση στα αντικείμενα που παρέχουν υπηρεσίες.
2). Αν πρέπει να σχεδιαστεί ένα σύνολο κλάσεων με παρόμοιες ιδιότητες, τότε χρησιμοποιείται το πρότυπο της αφηρημένης υπέρ-κλάσης.
Σε περίπτωση που πρέπει στον ίδιο σχεδιασμό να υλοποιηθούν και οι δυο ανάγκες, δηλαδή το ένα και δυο (1και 2), τότε χρησιμοποιείται το πρότυπο της διεπαφής και αφηρημένης κλάσης, όπως φαίνεται στο σχήμα 2.4.1 [4].
Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του πρότυπου αφηρημένης κλάσης και διεπαφής.



Σχήμα 2.4.1 : Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου αφηρημένης κλάσης και διεπαφής.

2.5 Εξουσιοδότηση-Εκπροσώπευση (Delegation)

Η κληρονομικότητα επεκτείνει (extends) τη λειτουργία μιας κλάσεως και μέσω αυτής δίνει την δυνατότητα στις κληρονομούσες κλάσεις να χρησιμοποιήσουν τις μεθόδους που κληρονόμησαν.
Ωστόσο, σε πολλές περιπτώσεις, η χρησιμοποίηση της ιδιότητας της κληρονομικότητας της αντικειμενοστρεφούς τεχνολογίας, για την οργάνωση των αντικειμένων ενός λογισμικού συστήματος είναι ακατάλληλη ως λύση. Η εκπροσώπηση μπορεί να εφαρμοστεί σε τέτοιες περιπτώσεις, προσφέροντας έναν διαφορετικό τρόπο ο οποίος, ενώ δεν παρέχει δυνατότητες επέκτασης των κλάσεων, μπορεί να επιτρέψει τη χρήση των μελών τους από μια άλλη. Τα μέλη των κλάσεων που χρησιμοποιούνται είναι κυρίως οι μέθοδοι.

Η κληρονομικότητα, χαρακτηρίζεται από την λέξη-κλειδί “is”. Δηλαδή, η κληρονομούσα κλάση “είναι” (is) η πατρική κλάση. Η έννοια “είναι” μεταξύ κλάσεων έχει την σημασία ότι εννοιολογικά η οντότητα που παρίσταται από την πατρική κλάση ενσωματώνεται στην οντότητα που παρίσταται από την κληρονομούσα κλάση. Εάν η πατρική κλάση εκπροσωπεί την οντότητα Vehicle (όχημα), μια κλάση που εκπροσωπεί την οντότητα Bus (λεωφορείο) μπορεί να κληρονομήσει την πατρική γιατί ένα λεωφορείο “είναι” όχημα. Επιπλέον η κληρονομούσα μπορεί να έχει κάποια επιπλέον στοιχεία που εξειδικεύουν την κληρονομηθείσα οντότητα και την κάνουν πιο συγκεκριμένη. Με την έννοια αυτή η κληρονομούσα “επεκτείνει” την πατρική κλάση. Συνεπώς η οντότητα λεωφορείο θεωρείται εξειδίκευση της οντότητας όχημα που επεκτείνει την αρχική οντότητα, αφού η εξειδίκευση προέρχεται αναγκαστικά από ιδιότητες και συμπεριφορές που χαρακτηρίζουν μοναδικά το λεωφορείο και το διακρίνουν από την γενικότερη έννοια του οχήματος και από άλλα είδη οχημάτων. Τέτοιο χαρακτηριστικό είναι ότι ένα λεωφορείο μπορεί να μεταφέρει μεγάλο πλήθος επιβατών (εκφραζόμενο ως μια μεταβλητή NumberOfPassengers) ενώ έχει την ιδιότυπη συμπεριφορά να σταματά συχνά και σε όλες τις στάσεις (εκφραζόμενο ως μέθοδος busStopping).
Όσον αφορά την δυνατότητα χρήσης των κληρονομημένων μελών από την κληρονομούσα, αυτό θεωρείται αυτονόητο. Δηλαδή, ιδιότητες και συμπεριφορές (ισοδύναμα: μεταβλητές και μέθοδοι) της πατρικής κλάσης, μπορούν να χρησιμοποιηθούν από την κληρονομούσα κλάση. Ωστόσο, η δυνατότητα χρήσης αυτών των στοιχείων δεν είναι ποτέ αυτοσκοπός. Απαγορεύεται η υιοθέτηση οργάνωσης των αντικειμένων λογισμικού συστήματος με σχήματα κληρονομικότητας, μόνο και μόνο για να αξιοποιηθεί η δυνατότητα χρήσης των κληρονομημένων στοιχείων. Απαραίτητη προϋπόθεση για την εφαρμογή της ιδιότητας της κληρονομικότητας στο σχεδιασμό αντικειμενοστρεφών λογισμικών συστημάτων, είναι η εννοιολογική ικανοποίηση του κριτηρίου “is” μεταξύ των οντοτήτων που εκπροσωπούνται από τις κλάσεις που πρόκειται να συνδεθούν με κληρονομικότητα

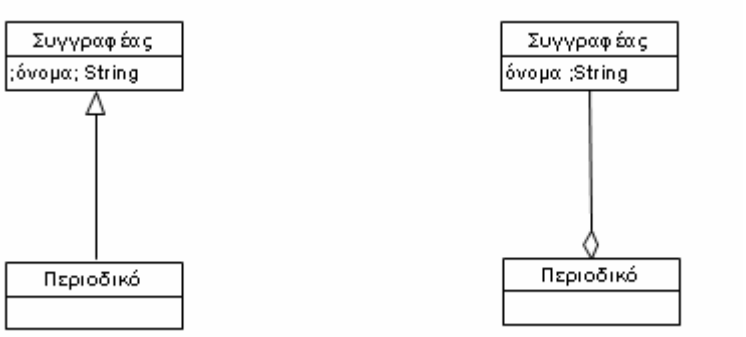
Σε όσες περιπτώσεις οι σχέσεις μεταξύ των οντοτήτων που παρίστανται από τις αντίστοιχες κλάσεις δεν μπορούν να συνδεθούν με μια σχέση του τύπου “είναι” είναι ενδεχόμενο να μπορεί να χρησιμοποιηθεί το πρότυπο εκπροσώπησης.
Λέξεις κλειδιά για την εφαρμογή αυτού του προτύπου είναι οι λέξεις “έχει” (“has”-“has a”). Η εξουσιοδότηση- αντιπροσώπευση (Delegation) χρησιμοποιείται όταν δεν πρέπει να είναι φανερές μεταβλητές και μέθοδοι μιας κλάσης από την υπέρ-κλάση. Έχει ένα τρόπο να επεκτείνει τις κλάσεις πιο γενικού σκοπού από ότι η

κληρονομικότητα (Inheritance). Δηλαδή για να επεκταθεί η συμπεριφορά μιας κλάσης, η κλάση αυτή καλεί τις μεθόδους μιας δεύτερης κλάσης.
Η εξουσιοδότηση είναι μια τεχνική, σύμφωνα με την οποία ένα αντικείμενο εκφράζει μια συμπεριφορά, όμως η ευθύνη για τη συμπεριφορά αυτή ανήκει σε ένα άλλο αντικείμενο [7][9]. Μπορεί να αντιμετωπισθεί ως σχέση μεταξύ των αντικειμένων, όπου ένα αντικείμενο διαβιβάζει ορισμένες κλήσεις μεθόδου σε ένα άλλο αντικείμενο, αποκαλούμενο εκπρόσωπό του [8].

Ένα παράδειγμα για να κατανοηθεί η διαφορά μεταξύ κληρονομικότητας και εξουσιοδότησης είναι το εξής:

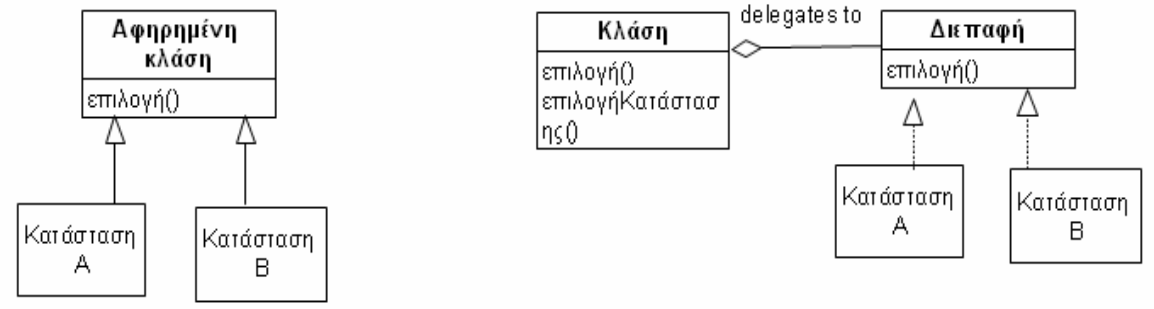
Έστω ότι οι υπάλληλοι μπορούν να ταξινομηθούν βάσει του τρόπου με τον οποίο πληρώνονται π.χ., ανά μία ώρα ή με μισθό.
Χρησιμοποιώντας την κληρονομικότητα, μπορούν να σχεδιαστούν τρεις κλάσεις: κλάση Υπαλλήλων που έχει κοινή λειτουργία για όλους τους υπαλλήλους, και δύο υποκλάσεις: ΥπάλληλοςΩρας και ΥπάλληλοςΜισθού που έχουν λεπτομέρειες σχετικά με την αμοιβή των υπαλλήλων. Ενώ αυτό το πρότυπο είναι κατάλληλο για μερικές εφαρμογές, θα δημιουργηθούν προβλήματα σε ένα σενάριο, όπου ένα πρόσωπο αλλάζει, για παράδειγμα από ανά μία ώρα σε μισθοδοτούμενο τον τρόπο πληρωμής. Η κλάση ενός αντικειμένου και η σχέση κληρονομικότητας είναι και οι δυο στατικές, και αυτό σημαίνει ότι τα αντικείμενα δεν μπορούν να αλλάξουν την κλάση τους εύκολα.
Ένα πιο ευέλικτο πρότυπο θα περιλάμβανε εξουσιοδότηση. Ένα αντικείμενο υπαλλήλων θα μπορούσε να εξουσιοδοτήσει τις κλήσεις μεθόδου σχετικά με τις πληρωμές σε ένα αντικείμενο, του οποίου ευθύνη θα είναι μόνο ο τρόπος με τον οποίο πληρώνεται ο υπάλληλος. Στην πραγματικότητα, χρησιμοποιείται η κληρονομικότητα και εδώ, αλλά με διαφορετικό τρόπο. Δημιουργείται μια αφηρημένη κλάση (ή διεπαφή) αποκαλούμενη ΚατηγοριοποίησηΠληρωμής με δύο υποκαλάσεις ΜεΤηνΩραΚατηγοριοποίηση και ΜεΜισθοΚατηγοριοποίηση που εφαρμόζουν συγκεκριμένους υπολογισμούς σύμφωνα με την κατηγοριοποίηση. Χρησιμοποιώντας την εξουσιοδότηση όπως παρουσιάζεται, θα ήταν πολύ πιο εύκολο να αλλαχτεί η ταξινόμηση αμοιβής ενός υπάρχοντος αντικειμένου υπαλλήλων.
Σε άλλες αναφορές παρουσιάζεται ως η ιδέα ,ότι όταν πρέπει να αλλάξει ο τρόπος που μια κλάση παρέχεται, η αντιπροσώπευση παρέχει έναν γενικότερο τρόπο από την κληρονομικότητα [30].

Παρακάτω παρουσιάζονται δυο τρόποι για να σχεδιαστεί το μοντέλο ‘Magazines have Authors’ είναι με κληρονομικότητα και με εξουσιοδότηση. Η κληρονομικότητα δουλεύει σωστά, επειδή το Περιοδικό (Magazine) κληρονομεί το όνομα του Συγγραφέα (Author), όμως είναι κακός σχεδιασμός. Το πρόβλημα βρίσκεται στο γεγονός ότι το Magazine – is – NOT – a – kind – of – Author. Πιο σωστό είναι ότι το Magazine – has – an– Author.



Σχήμα 2.5.1 : Αριστερά απεικονίζεται η κληρονομικότητα , ενώ δεξιά η εξουσιοδότηση.

Ένα γενικότερο σχήμα που δείχνει τη διαφορά μεταξύ κληρονομικότητας και εξουσιοδότησης είναι:



Σχήμα 2.5.2 : Αριστερά απεικονίζεται η κληρονομικότητα , ενώ δεξιά η εξουσιοδότηση.

Προστίθεται μια νέα κλάση, η οποία συγχωνεύει τη λειτουργία της αρχικής κλάσης και χρησιμοποιεί ένα στιγμιότυπό της , καλώντας τις μεθόδους της [6] .
Η εξουσιοδότηση είναι γενικότερου σκοπού από την κληρονομικότητα. Αυτό φαίνεται από το γεγονός, ότι με την εξουσιοδότηση ένα αντικείμενο μπορεί να αλλάζει συμπεριφορά με την πάροδο του χρόνου, κάτι το οποίο δεν μπορεί να γίνει με την κληρονομικότητα. Αυτό συμβαίνει επειδή η κληρονομικότητα είναι στατική. Ένα αρνητικό στοιχείο της εξουσιοδότησης είναι ότι απαιτεί μεγαλύτερο κομμάτι κώδικα να γραφτεί, από ό,τι απαιτεί η κληρονομικότητα. Οποιαδήποτε επέκταση σε μια κλάση που μπορεί να ολοκληρωθεί από την κληρονομικότητα μπορεί επίσης να ολοκληρωθεί από την εξουσιοδότηση.
Σε άλλες αναφορές το πρότυπο της εξουσιοδότησης αναφέρεται ως πρότυπο ιδρύματος. Δηλαδή πρότυπα που είτε χρησιμοποιούνται είτε ειδικεύονται από άλλα πρότυπα [29].

2.6 Αμετάβλητο πρότυπο (Immutable pattern)

Τα αντικείμενα είναι στιγμιότυπα (instances) της κλάσης τους και από μια κλάση μπορούν να κατασκευαστούν πολλά στιγμιότυπα (αντικείμενα). Η κλάση ορίζει τι πεδία έχει ένα αντικείμενο, αλλά κάθε αντικείμενο αποθηκεύει το δικό του σύνολο τιμών. Το σύνολο των τιμών των πεδίων ενός αντικειμένου ονομάζεται κατάσταση

του αντικειμένου. Ακόμα και όταν δύο αντικείμενα είναι πανομοιότυπα (ίδια κατάσταση), ξεχωρίζουν γιατί έχουν διαφορετικό όνομα και οντότητα [10].

Στον αντικειμενοστρεφή προγραμματισμό ένα αμετάβλητο αντικείμενο είναι ένα αντικείμενο του οποίου η κατάσταση δεν μπορεί να αλλαχτεί, αφού δημιουργηθεί και μετά. Αυτό είναι αντίθετο με το μεταβλητό αντικείμενο, στο οποίο η κατάστασή του μπορεί να αλλαχτεί αφού έχει δημιουργηθεί [11] [14]. Σε περιπτώσεις όπου πολλά αντικείμενα έχουν πρόσβαση στο ίδιο αντικείμενο, πολλές φορές δημιουργείται πρόβλημα στο εν λόγω αντικείμενο.

Για κάθε αντικείμενο πρέπει να υπάρχει μια αναφορά- χειριστήριο με την οποία να αποκτάται πρόσβαση στο ίδιο το αντικείμενο (τα δεδομένα και τις μεθόδους του) [12]. Το αμετάβλητο πρότυπο αυξάνει την ευρωστία των αντικειμένων που μοιράζονται αναφορές στο ίδιο αντικείμενο και μειώνει τα γενικά έξοδα (overhead) της ταυτόχρονης πρόσβασης σε ένα αντικείμενο. Αυτό επιτυγχάνεται με τον εξής τρόπο. Δεν επιτρέπει να αλλάξει η κατάσταση ενός αντικειμένου αφότου κατασκευαστεί [31]. Το αμετάβλητο πρότυπο αποφεύγει επίσης την ανάγκη να συγχρονιστούν τα πολλά νήματα της εκτέλεσης που χρησιμοποιούν ένα αντικείμενο [13]. Πιο αυστηρά ορίζεται το πρότυπο αυτό ,ως η εγγύηση ότι ένα αντικείμενο δεν θα αλλαχτεί, όταν νήματα δεν πρέπει να τροποποιήσουν ταυτόχρονα ένα αντικείμενο [32].

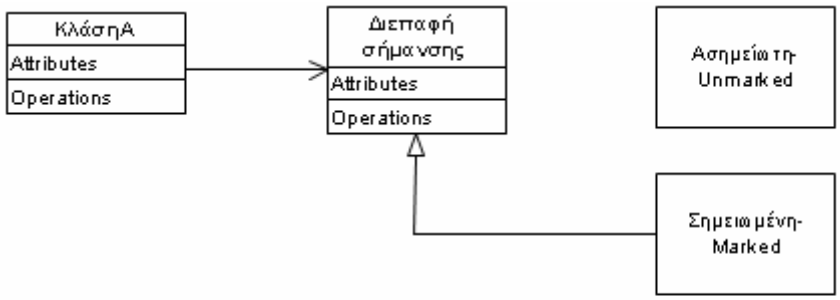
Αν οι αλλαγές στο αντικείμενο που το χρησιμοποιούν δεν γίνουν συγχρονισμένα, τότε υπάρχει σοβαρό πρόβλημα με το αντικείμενο. Με τον όρο συγχρονισμένα εννοείται ότι μόλις τελειώσουν οι αλλαγές στο αντικείμενο, τότε επιτρέπεται να αρχίσουν νέες και όχι νωρίτερα. Τα προβλήματα που θα προκύψουν είναι η δημιουργία bugs και ότι το υπόλοιπο τμήμα που δουλεύει σωστά, θα έχει επιπλέον γενικά έξοδα (overhead) συγχρονισμού των προσβάσεων στην κατάσταση του αντικειμένου [4]. Επομένως, σχεδιάζοντας ένα αντικείμενο για πρώτη φορά, από τα πρώτα πράγματα που πρέπει να ληφθούν υπόψη είναι , εάν θα είναι μεταβλητό ή όχι.

Μια διεργασία είναι ένα αυτοδύναμο περιβάλλον εκτέλεσης (self-contained execution environment), που διαθέτει τους δικούς της αποκλειστικούς πόρους κατά το χρόνο εκτέλεσης της (run-time resources) με πιο σημαντικό αυτό της μνήμης. Μια εφαρμογή είναι δυνατό να αποτελείται από διάφορες συνεργαζόμενες διεργασίες.
Ένα νήμα είναι μια ελαφριά διεργασία , επομένως είναι και αυτό ένα αυτοδύναμο περιβάλλον εκτέλεσης. Κάθε διεργασία περιέχει ένα ή περισσότερα νήματα εκτέλεσης και όλα τα νήματα μιας διεργασίας μοιράζονται από κοινού τους πόρους της διεργασίας, μοιράζονται δηλαδή ένα κοινό χώρο στη μνήμη.
Τα νήματα χρησιμεύουν όταν πρέπει να υλοποιηθεί μια εφαρμογή, η οποία να μπορεί να εκτελεί ταυτόχρονα περισσότερες από μια εργασίες. Σε αντίθεση με τις διεργασίες, τα νήματα μπορούν να δημιουργηθούν με μικρότερο επίβαρο (overhead) και συνεπώς μπορούν να υποστηρίξουν με πιο αποδοτικό τρόπο την ταυτόχρονη διεκπεραίωση αυτών των εργασιών.
Τα προβλήματα των πολυνηματικών εφαρμογών προκύπτουν λόγω του ότι τα νήματα μοιράζονται μεταξύ τους κάποια πεδία και κάποια αντικείμενα. Τα προβλήματα αυτά είναι παρεμβολές νημάτων (thread interference) και απώλεια συνοχής μνήμης (memory consistency errors). Η λύση είναι ο συγχρονισμός των νημάτων, μια λύση αρκετά ‘‘ακριβή’’ [15] .

Το αμετάβλητο πρότυπο είναι το πιο χρήσιμο πρότυπο σε πολυνηματικές διεργασίες. Αυτό συμβαίνει, επειδή, όταν μια διεργασία είναι να εκτελεστεί σε δεδομένα που αναπαριστά ένα αμετάβλητο αντικείμενο, τότε ένα άλλο νήμα μπορεί να δράσει στα ίδια δεδομένα, χωρίς το φόβο της αλλαγής του αντικειμένου, κατά τη διάρκεια της λειτουργίας [11].
Αφού τα αμετάβλητα αντικείμενα δεν μπορούν να μεταβληθούν, τότε δεν χρειάζεται να δημιουργείται αντίγραφο ασφαλείας του αντικειμένου, για την προστασία από τυχόν επιθέσεις κακόβουλων. Επιπλέον αφού δεν χρειάζεται να δημιουργούνται αντίγραφα συνεχώς, μειώνεται η μνήμη που θα χρησιμοποιούταν διαφορετικά.
Η ιδέα του αμετάβλητου αντικειμένου δεν έμεινε στάσιμη. Πολλές αναφορές έχουν γίνει για την έννοια της αμεταβλητότητας. Δηλαδή είναι πολύ χρήσιμη η ιδιότητα που έχουν ορισμένα αντικείμενα να μένουν αμετάβλητα. Τέτοιες αναφορές γίνονται στα άρθρα ‘‘Permissions to Specify the Composite Design Pattern’’ [33] και ΄΄Immutable List Structure΄΄ [34] .

2.7 Πρότυπο σήμανσης διεπαφής (Marker Interface)

Το πρότυπο σήμανσης αποτελεί μια διεπαφή χωρίς δηλωμένες μεθόδους, την οποία μπορεί να εφαρμόσει μια κλάση για να δείξει ότι έχει συγκεκριμένες ιδιότητες. Πολλές φορές χρησιμοποιείται με γλώσσες που εξασφαλίζουν (run time) πληροφορίες για τα αντικείμενα. [16]. Χρησιμοποιείται όταν είναι ανάγκη να υποδειχθούν οι ιδιότητες μιας κλάσης. Επιτρέπει τον προσδιορισμό των ιδιοτήτων των αντικειμένων χωρίς να υποθέτει ότι είναι στιγμιότυπα μιας κλάσης [35].
Για να χρησιμοποιηθεί το πρότυπο αυτό , μια κλάση δημιουργεί μια διεπαφή σήμανσης και μεθόδους, οι οποίες αλληλεπιδρούν με τα στιγμιότυπα της κλάσης και ελέγχουν για την ύπαρξη της διεπαφής. Όλες οι κλάσεις που δημιουργούν την διεπαφή αυτή, δεν χρειάζεται να καλέσουν καμία μέθοδο για την διεπαφή (αφού δεν έχει διεπαφή). Όμως οι κλάσεις που εφάρμοσαν τις διεπαφές είναι σημειωμένες (Marked) . Το πρότυπο σήμανσης επιτρέπει σε μια κλάση να εφαρμόσει μια διεπαφή σήμανσης (marker interface), η οποία εκθέτει πληροφορίες της κλάσης που δεν μπορούν να καθοριστούν αποκλειστικά από τις μεθόδους της κλάσης [16].
Το πρότυπο λειτουργεί με τον ακόλουθο τρόπο: εάν το αντικείμενο είναι στιγμιότυπο της κλάσης θα παραγάγει σημειωμένη-Marked , εάν δηλαδή το αντικείμενο έχει μια ιδιότητα της κλάσης. Συνήθως δείχνει αν μια κλάση πρέπει ή δεν πρέπει να θεωρηθεί αποδεκτή κλάση για μια ορισμένη λειτουργία.
Η ουσία του προτύπου αυτού είναι ότι ένα αντικείμενο που είτε εφαρμόζει είτε δεν εφαρμόζει μια διεπαφή σήμανσης, περνιέται σε μια μέθοδο μιας κλάσης. Η παράμετρος που αντιστοιχεί στο αντικείμενο δηλώνεται χαρακτηριστικά ως αντικείμενο. Η δήλωση ότι μια κλάση εφαρμόζει μια διεπαφή σήμανσης υπονοεί ότι η κλάση συμπεριλαμβάνεται στην ταξινόμηση που υπονοείται από τη διεπαφή. Επίσης υπονοεί ότι όλες οι υποκλάσεις εκείνης της κλάσης συμπεριλαμβάνονται στην ταξινόμηση.
Στο σχήμα 2.7.1 απεικονίζεται το διάγραμμα κλάσεων του πρότυπου σήμανσης.



Σχήμα 2.7.1 : Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου σήμανσης.

Το σχήμα 2.6 παρουσιάζει μια διεπαφή σήμανσης . Επίσης παρουσιάζει μια κλάση που καλείται Σημειωμένη- (Marked) που εφαρμόζει την διεπαφή αυτή, μια κλάση Unmarked που δεν την εφαρμόζει και μια ΚλάσηΑ που γνωρίζει τη διεπαφή .
Τα στιγμιότυπα της ΚλάσηςΑ λαμβάνουν τις κλήσεις σε μια μέθοδο τους .Η παράμετρος που περνούν σε εκείνη την μέθοδο μπορεί να είναι ένα αντικείμενο που εφαρμόζει ή δεν εφαρμόζει την διεπαφή.
Το πρότυπο αυτό βρίσκει διάφορες εφαρμογές, όπως δημιουργία σχημάτων αφηρημένης αναπαράστασης περιεχομένου [36]. Επίσης σε διάφορες εργασίες εφαρμόζεται η κεντρική ιδέα του προτύπου αυτού. Πιο συγκεκριμένα μια από αυτές αναφέρεται ότι στην Java υπάρχει ένα κοινό τέχνασμα, αποκαλούμενο marker interface [37].

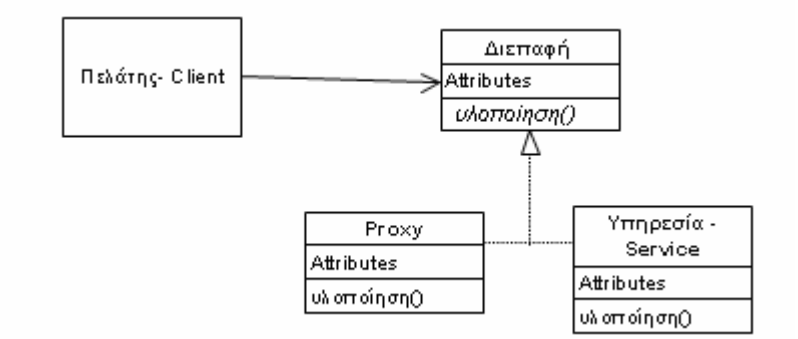
2.8 Πρότυπο Πληρεξουσίου (Proxy Pattern)

Το πρότυπο πληρεξουσίου (Proxy Pattern) χρησιμοποιείται όταν πρέπει να αναπαρασταθεί κάτι που είναι πολύπλοκο σε κάτι πιο απλό. Εάν η δημιουργία ενός αντικειμένου είναι δύσκολη και ακριβή ,τότε χρησιμοποιείται ένα απλό αντικείμενο. Το αντικείμενο αυτό είναι το Proxy. Επίσης χρησιμοποιείται σε περιπτώσεις όπου μια κλάση δεν μπορεί ή δεν είναι επιθυμητό να επικοινωνεί άμεσα με ένα αντικείμενο, όμως πρέπει με κάποιο τρόπο να επικοινωνεί με αυτό.
Όταν χρειάζεται να χρησιμοποιηθούν λίγες μέθοδοι από ορισμένα ‘‘ακριβά-βαριά’’ αντικείμενα, δεν συμφέρει να χρησιμοποιηθούν τα αντικείμενα. Θα χρησιμοποιηθούν σε περίπτωση που είναι ολόκληρα απαραίτητα. Μια λύση είναι να χρησιμοποιηθούν ‘‘ελαφριά’’ αντικείμενα, τα οποία θα έχουν την ίδια διεπαφή με τα ‘‘ακριβά’’ αντικείμενα. Τα ‘‘ελαφριά’’ αντικείμενα λέγονται proxies και χρησιμοποιούν τα ‘‘ακριβά ‘’ αντικείμενα όταν είναι αναγκαίο [19].
Ένα απλό παράδειγμα για να γίνει κατανοητή η χρήση του προτύπου αυτού είναι το εξής: Έστω ότι υπάρχει ένας εργαζόμενος σε ηλεκτρονικό υπολογιστή, ο οποίος βλέπει διάφορες εικόνες στην οθόνη του και επιλέγει κάποιες από αυτές. Κάποιες φορές χρειάζεται να ανοίγει το συγκεκριμένο φάκελο με τις φωτογραφίες και να βλέπει όλες τις φωτογραφίες. Άλλες φορές όμως θέλει απλά να βρει μια φωτογραφία από το όνομά της. Επομένως δεν είναι απαραίτητο να φορτωθούν στην μνήμη του υπολογιστή πάλι όλες οι φωτογραφίες.
Ένα πρότυπο πληρεξούσιου χρησιμοποιεί αντικείμενα πληρεξούσιου κατά τη διάρκεια της αλληλεπίδρασης του αντικειμένου. Ένα αντικείμενο πληρεξούσιου

ενεργεί ως υποκατάστατο του πραγματικού αντικειμένου. Για παράδειγμα, όταν πρέπει ένα αντικείμενο να αλληλεπιδράσει με ένα μακρινό αντικείμενο, για παράδειγμα μέσω ενός δικτύου, ο τρόπος που προτιμάται για την αλληλεπίδραση αυτή και για να μην είναι φανερός ο μηχανισμός αλληλεπίδρασης, είναι με τη χρησιμοποίηση ενός αντικειμένου πληρεξούσιου. Το αντικείμενο πληρεξουσίου είναι αυτό που μεσολαβεί για την επικοινωνία μεταξύ του αντικειμένου αίτησης και του μακρινού αντικειμένου [17].

Μερικοί τύποι πληρεξούσιων είναι οι ακόλουθοι:
1). Απομακρυσμένο πληρεξούσιο (Remote Proxy) - παρέχει μια αναφορά σε ένα αντικείμενο που βρίσκεται σε ένα διαφορετικό διάστημα διευθύνσεων στην ίδια ή διαφορετική μηχανή.
2). Εικονικό πληρεξούσιο (Virtual Proxy) - επιτρέπει τη δημιουργία ενός εντατικού αντικειμένου μνήμης όταν αυτό ζητηθεί. Το αντικείμενο δεν θα δημιουργηθεί έως ότου απαιτείται πραγματικά.
3). Copy-On-Write-Proxy - αναβάλλει την αντιγραφή (κλωνοποίηση) ενός αντικειμένου στόχου μέχρι να το απαιτήσουν οι ενέργειες των πελατών. Πραγματικά μια μορφή εικονικού πληρεξούσιου.
4). Πληρεξούσιο προστασίας (πρόσβαση) (Protection access Proxy) - παρέχει σε διαφορετικούς πελάτες , διαφορετικά επίπεδα πρόσβασης σε ένα αντικείμενο στόχου.
5). Πληρεξούσιο κρύπτης (Cache Proxy) - παρέχει την προσωρινή αποθήκευση των αποτελεσμάτων των ακριβών διαδικασιών στόχων, έτσι ώστε πολλοί πελάτες να μπορούν να μοιραστούν τα αποτελέσματα
6). Firewall Proxy - προστατεύει τους στόχους από κακόβουλους πελάτες.
7). Πληρεξούσιο συγχρονισμού (Synchronization Proxy) - παρέχει πολλές προσβάσεις σε ένα αντικείμενο στόχος.
8). Έξυπνο πληρεξούσιο αναφοράς (Smart Reference Proxy) - παρέχει τις πρόσθετες ενέργειες όποτε ένα αντικείμενο στόχου έχει αναφερθεί, όπως ο υπολογισμός του αριθμού αναφορών στο αντικείμενο [20].

Γενικά το Proxy Pattern προτείνει μια λύση στο πρόβλημα της ανεξαρτητοποίησης του Client – πελάτη και των κλάσεων που χρησιμοποιεί, από τη διαδικασία πρόσβασης τα αντικείμενα αυτών των κλάσεων. Ο πελάτης δεν έχει αναφορά στο πραγματικό αντικείμενο. Έχει όμως αναφορά σε ένα αντικείμενο (το Proxy) που προσφέρει η ίδια η διεπαφή . Ο πελάτης χειρίζεται το Proxy με το ίδιο τρόπο που θα χειριζόταν την Υπηρεσία και επικαλείται την μέθοδο υλοποίησης(). Ο πελάτης τώρα μπορεί να δημιουργήσει ένα αντικείμενο της Υπηρεσίας και να καλέσει την μέθοδο υλοποίησης() στο αντικείμενο.
‘‘Ένα πρότυπο πληρεξούσιου ενθυλακώνει ένα χαρακτηριστικό σχεδιασμού όπου ένας αναπληρωτής χρησιμοποιείται για να αντιπροσωπεύσει ή να ρυθμίσει την πρόσβαση σε έναν προϊστάμενο. Στην καθημερινή ζωή, τα παραδείγματα ενός σχεδίου πληρεξούσιου περιλαμβάνουν ένα εικονίδιο ιστοσελίδας, ενώ μια εικόνα μεταφορτώνεται’’, όπως αναφέρεται στο βιβλίο Pro .NET 2.0 Code and Design Standards in C# [39].
Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του πρότυπου πληρεξουσίου.



Σχήμα 2.8.1 : Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου πληρεξουσίου.

Όπως φαίνεται και στο σχήμα ,η Proxy έχει μια αναφορά με την οποία έχει πρόσβαση στην Υπηρεσία. Χρησιμοποιεί την ίδια διεπαφή με την Υπηρεσία, έτσι ώστε ένα proxy να μπορεί να αλλαχτεί για την υπηρεσία. Έχει πρόσβαση στην υπηρεσία και μπορεί να τη δημιουργεί και να τη διαγράφει. Η Διεπαφή είναι μια κοινή διεπαφή για την Υπηρεσία και την Proxy. Επομένως η Proxy μπορεί να χρησιμοποιηθεί οπουδήποτε η Υπηρεσία χρειαστεί[18].

Αναφορές:

[1]: http://en.wikipedia.org/wiki/Entity
[2]: Dirk Riehle, (2000), *Fundamental Class Patterns in Java,* http://dirkriehle.com/computer-science/research/2000/plop-2000-class-patterns.html
[3]: http://en.wikipedia.org/wiki/Interface_Pattern
[4]: Mark Grand, (2002*), Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML, Second Edition* – Team FLY, Wiley, pp.562.
[5]: Bobby Woolf , (1997), *The Abstract Class Pattern*, Knowledge Systems Corp.
[6]: Design David Leberknight & Ron LeMaster, (1996), *Object - Oriented Programming & Design Part V: Object- Oriented.*
[7]: http://en.wikipedia.org/wiki/Delegation_pattern
[8]: http://www.jguru.com/faq/view.jsp?EID=27916
[9]: http://www.knowledgerush.com/kr/encyclopedia/Delegation_pattern/
[10]: Ιωάννης Γαβιώτης, (2007), http://www.syros.aegean.gr/users/gaviotis/tmp2
[11]: http://www.nationmaster.com/encyclopedia/Immutable-pattern
[12]: *Αντικειμενοστραφής Προγραμματισμός 1 Ανακεφαλαίωση 12-1-2006*, Τμήμα Πολιτισμικής Τεχνολογίας και Επικοινωνίας
[13]: http://www.mindspring.com/~mgrand/pattern_synopses.htm#Immutable
[14]: http://www.123exp-computing.com/t/03974210977/
[15]: Σταύρος Πολυβίου, *Πολυνηματικές εφαρμογές σε Java*
[16]: http://www.nationmaster.com/encyclopedia/Marker-interface
[17]: http://www.developer.com/design/article.php/10925_3309461_2
[18]: http://www.dofactory.com/Patterns/Patterns.aspx#list
[19]: http://www.oodesign.com/proxy-pattern.html
[20]: Bob Tarr, (2000), *The Proxy Pattern Design Patterns In Java*
[21]: Neal S. Holter, Madhusmita Mitra, Amos Maritan, Marek Cieplak, Jayanth R. Banavar, and Nina V, (2000), *Fundamental patterns underlying gene expression profiles: Simplicity from complexity*, Proc Natl Acad Sci U S A. 2000 July 18; 97(15): 8409–8414. Published online 2000 July 11.
[22]: Elaine Winters, (2003), *Fundamental Patterns of Cultural Differences*, International Technical Communication SIG STC Online article April, 2003
[23]: Belinda Stillwell, (2008), California State University, Northridge Department of Kinesiology KIN 271/Lab - Fundamental Movement, Gymnastics and Rhythms Fall 2008
[24]: Discrimination and clustering for fundamental frequency pattern
analysis to investigate infant language acquisition by Hiroko KATO, Masanobu TANIGUCHI ,Tomohiro NAKATANI and Shigeaki AMANO NTT

[25]: Wim Laurier and Geert Poels, (2007), *Three Fundamental REA Business Patterns* , PROGRAM REA-25, June 13-15, 2007.
[26]: Michael Schaper, (2004), *The "One in Twenty" Rule: Some Thoughts On The Fundamental Distribution Of Firms*, Paper for Rencontres conference, 20-23 September 2004, Appenzell, Switzerland.
[27]: A.Gotz, P.Verdier, J.Coquet , A.Buteau, C.Scafuri, (2005), *ABSTRACT DEVICE PATTERN AND TANGO***,** 10th ICALEPCS Int. Conf. on Accelerator & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, PO1.037-6 (2005).
[28]: Cabral LIMA, Henrique GANDRA, (2003), *Learning of design patterns: the LeSoop system*, Seventh Workshop on Pedagogies and Tools for Learning Object-Oriented Concepts Germany 2003.
[29]: Dwight Deugo, (2003), *Foundation Patterns*, USENIX 2003 Annual Technical Conference, FREENIX pp29-38.
[30]: Ian Holyer, *Design Patterns* , COMS30205 Advanced Software Development,
[31]: MARIEKE HUISMAN AND CLEMENT HURLIN, (2006), *PERMISSION SPECIFICATIONS FOR COMMON MULTITHREADED PROGRAMMING PATTERNS*, *In* Reflections on Type Theory, Lambda Calculus, and the Mind. Essays Dedicated to Henk Barendregt on the Occasion of his 60th Birthday, *2007.*
[32]: Ulrich Schaefer, (2008), *Design Pattern* Algorithms, Tools and Techniques in JAVA for Natural Language Processing - Winter 2008/2009.
[33]: Kevin Bierhoff & Jonathan Aldrich, (2008), *Permissions to Specify the Composite Design Pattern*, 7th International Workshop on Specification and Verification of Component-Based Systems (SAVCBS '08) at FSE-16, Atlanta, GA, USA, November 9-10, 2008.
[34]: Dung X. Nguyen, (2000), *IMMUTABLE LIST STRUCTURE Intermediate Programming*, Comp 212 - Intermediate Programming Rice University - Fall 2000.
[35]: Bruegge B, Dutoit A H (2004), *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Upper Saddle River: Pearson Prentice Hall, NJ, 2004.
[36]: Stephen Cranefield, Mariusz Nowostawski and Martin Purvis, (2001), *A proposal for a strongly-typed ACR framework*, International Conference on Autonomous Agents, Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2, Bologna, Italy, SESSION: Session 5A: agent communication languages 2002, pp.553 – 554.
[37]: James Newkirk and Alexei A. Vorontsov, (2002), *How .NET's Custom Attributes Affect Design* , IEEE Software, vol. 19, no. 5, Sep./Oct. 2002, pp. 18-20.
[38]: Walter L. Hursch, (2001), *Shoyld Superclasses be abstract?* , International Conference on Software Engineering, Proceedings of the 23rd International Conference on Software Engineering, 2001,Toronto, Ontario, Canada, pp. 253 – 262.
[39]: Horner, Mark, (2005), *Pro .NET 2.0 Code and Design Standards in C#* , PublisherApress.

3. ΔΗΜΙΟΥΡΓΙΚΑ ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ
Creational Design Patterns

3.1 Σύντομη αναφορά των δημιουργικών σχεδιαστικών προτύπων

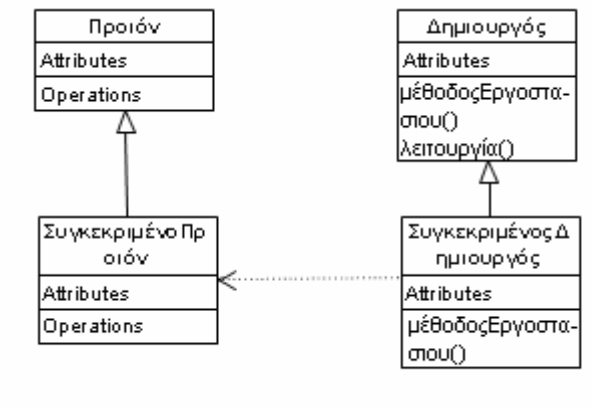
Τα δημιουργικά σχεδιαστικά πρότυπα είναι σχεδιαστικά πρότυπα , τα οποία έχουν να κάνουν με μηχανισμούς δημιουργίας αντικειμένων, προσπαθώντας να δημιουργήσουν αντικείμενα με τρόπο που θα ταιριάζει σε κάθε διαφορετική κατάσταση. Η δημιουργία του αντικειμένου χωρίς τη χρήση των δημιουργικών σχεδιαστικών

προτύπων, μπορεί να οδηγήσει σε προβλήματα σχεδίασης ή πολυπλοκότητα στο σχεδιασμό. Τα δημιουργικά σχεδιαστικά πρότυπα λύνουν το πρόβλημα αυτό, ελέγχοντας τη δημιουργία του αντικειμένου [1].
Αυτά τα πρότυπα παρέχουν τις οδηγίες σχετικά με τη δημιουργία των αντικειμένων. Βοηθούν να μην είναι φανερές λεπτομέρειες αρχικοποίησης αντικειμένου από τον κώδικα που χρησιμοποιεί εκείνα τα αντικείμενα. Με αυτόν τον τρόπο κάνουν ένα σύστημα ανεξάρτητο, για το πώς τα αντικείμενά του δημιουργούνται, συντίθενται και αναπαρίστανται. Αυτό οδηγεί στην υψηλή συνοχή μεταξύ των αντικειμένων και των χρηστών τους, αλλά σε μια χαμηλή σύζευξη μεταξύ των χρηστών και του τρόπου που τα αντικείμενα δημιουργούνται
Όλα τα δημιουργικά σχεδιαστικά πρότυπα προσπαθούν να βρουν τον καλύτερο τρόπο να δημιουργήσουν αντικείμενα. Αυτό είναι σημαντικό, επειδή, αν χρειαστεί να δημιουργηθεί ένα πρόγραμμα, αυτό δεν πρέπει να εξαρτάται από τον τρόπο με τον οποίο τα αντικείμενα δημιουργούνται και τακτοποιούνται.
Ανάλογα με το πώς δημιουργείται το αντικείμενο στα πλαίσια του προγράμματος, αυτό δημιουργεί δύσκολη δημιουργία κώδικα. Σε πολλές περιπτώσεις, η φύση του αντικείμενου που δημιουργείται θα μπορούσε να ποικίλει ανάλογα με τις ανάγκες του προγράμματος και το πρόγραμμα να γίνει πιο ευέλικτο και γενικό.
Τα δημιουργικά σχεδιαστικά πρότυπα παρέχουν τις οδηγίες σχετικά με το ποια κλάση θα αρχικοποιηθεί ή σε ποια αντικείμενα θα εξουσιοδοτήσει ένα αντικείμενο μια ευθύνη. Η αξία των προτύπων αυτών είναι να καθορίσει με ποιο τρόπο θα συγχωνευτούν όλες αυτές οι οδηγίες. Συχνά, υπάρχουν περισσότερα από ένα δημιουργικά σχεδιαστικά πρότυπα που μπορούν να εφαρμοστούν σε μια κατάσταση. Μερικές φορές μπορούν να συνδυαστούν αυτά τα πρότυπα.
Ένα από τα δημιουργικά σχεδιαστικά πρότυπα που θα εξεταστούν είναι το πρότυπο μεθόδου εργοστασίου (Factory Method Pattern), που ορίζει μια διεπαφή για τη δημιουργία ενός αντικειμένου, αλλά αφήνει τις υποκλάσεις να αποφασίσουν ποια κλάση θα αρχικοποιηθεί.
Ένα παρόμοιο πρότυπο είναι το αφηρημένο πρότυπο εργοστασίου (Abstract Factory Pattern), που παρέχει μια διεπαφή για τη δημιουργία οικογενειών των σχετικών ή εξαρτώμενων αντικειμένων χωρίς να διευκρινίσει τις κλάσεις τους.
Το πρότυπο οικοδόμησης-οικοδόμου (Builder Pattern) είναι ένα δημιουργικό σχεδιαστικό πρότυπο που διαχωρίζει την κατασκευή ενός σύνθετου αντικειμένου από την αναπαράστασή του, έτσι ώστε με την ίδια διαδικασία κατασκευής να μπορεί να δημιουργήσει διαφορετικές αναπαραστάσεις.
Η απόδοση μπορεί να είναι μερικές φορές το πιο σημαντικό θέμα κατά τη διάρκεια της ανάπτυξης λογισμικού και η δημιουργία αντικειμένου είναι ένα δαπανηρό βήμα. Ενώ το πρότυπο πρωτοτύπου (Prototype Pattern) βοηθά στη βελτίωση της απόδοσης με την κλωνοποίηση των αντικειμένων, το πρότυπο αποθέματος αντικειμένου (Object Pool Pattern) προσφέρει έναν μηχανισμό για να επαναχρησιμοποιήσει τα αντικείμενα που είναι ακριβά να δημιουργηθούν [12]. Επίσης το πρότυπο αντικειμένου αποθέματος (Object Pool Pattern) μπορεί να προσφέρει σημαντική αύξηση της απόδοσης για τον ακόλουθο λόγο. Είναι πολύ αποτελεσματικό σε περιπτώσεις όπου το κόστος για την αρχικοποίηση ενός στιγμιότυπου μιας κλάσης είναι πολύ μεγάλο, όμως ο αριθμός των στιγμιότυπων που χρησιμοποιούνται εκείνη τη στιγμή είναι μικρός [11] .
Το πρότυπο πρωτοτύπου (Prototype Pattern) χρησιμοποιείται όταν η δημιουργία ενός στιγμιότυπου μιας κλάσης είναι πολύ χρονοβόρα ή περίπλοκη-σύνθετη. Επιπρόσθετα, το πρότυπο Singleton επιτρέπει πολλά μέρη ενός προγράμματος να μοιράζονται ένα ενιαίο πόρο. Αυτό είναι χρήσιμο όταν απαιτείται ακριβώς ένα αντικείμενο να

συντονίσει τις ενέργειες ενός συστήματος. Μερικές φορές γενικεύεται σε συστήματα που λειτουργούν αποτελεσματικότερα όταν υπάρχουν μόνο ένα ή μερικά αντικείμενα [19].
Πολλά από τα δημιουργικά σχεδιαστικά πρότυπα αυτά αναφέρονται πολύ συχνά σε διάφορες εργασίες και εφαρμογές. Χαρακτηριστικό παράδειγμα είναι οι εφαρμογές AspectJ [22], που παρουσιάζονται βελτιώσεις του κώδικα των προτύπων. Επίσης στην εργασία με τίτλο ‘‘An Empirical Study on the Evolution of Design Patterns ’’ [23] παρουσιάζονται αποτελέσματα από μια εμπειρική μελέτη, η οποία έχει σκοπό να γίνει κατανοητή η εξέλιξη των προτύπων στα συστήματα JHotDraw, argoUML και Eclipse-JDT.

3.2 Πρότυπο Μεθόδου Εργοστασίου (Factory Method Pattern)

Το πρότυπο μεθόδου εργοστασίου (Factory Method Pattern) επιστρέφει ένα στιγμιότυπο από διάφορες κλάσεις ανάλογα με τα στοιχεία που παρέχονται σε αυτό. Συνήθως όλες οι κλάσεις εδώ έχουν μια κοινή πατρική κλάση και κοινές μεθόδους, αλλά κάθε μια από αυτές εκτελεί έναν στόχο διαφορετικά και βελτιστοποιείται για διαφορετικά είδη στοιχείων.
Συχνά μια εφαρμογή (application) ή ένα πλαίσιο (framework) πρέπει να διεκπεραιώσει ένα σύνολο ενεργειών, ανάλογα με το σκοπό που έχει να εκπληρώσει. Αφηρημένες κλάσεις χρησιμοποιούνται από την εφαρμογή, για να καθορίσουν και να διατηρήσουν τις σχέσεις μεταξύ των αντικειμένων. Η εφαρμογή πρέπει αρχικοποιήσει κλάσεις, αλλά γνωρίζει μόνο για τις αφηρημένες κλάσεις που δεν μπορούν να αρχικοποιηθούν. Δεν γνωρίζει την κλάση του αντικειμένου που πρέπει να δημιουργήσει. Οι εφαρμογή μπορεί να γνωρίζει πότε να αρχικοποιήσει ένα νέο αντικείμενο μιας κλάσης, αλλά όχι το είδος της υποκλάσης. Έτσι χρησιμοποιείται το πρότυπο μεθόδου εργοστασίου, το οποίο γνωρίζει ποια υποκλάση να δημιουργήσει.
Με άλλα λόγια το πρότυπο αυτό χρησιμοποιείται όταν μια κλάση δεν μπορεί να προβλέψει την κλάση των αντικειμένων , την οποία πρέπει να δημιουργήσει [3]. Επίσης μπορεί να εφαρμοστεί όταν μια κλάση θέλει οι υποκλάσεις της να διευκρινίζουν τα αντικείμενα που δημιουργεί. Μια ακόμα περίπτωση στην οποία μπορεί να εμφανιστεί το εν λόγω πρότυπο, είναι όταν κλάσεις εξουσιοδοτούν κάποια ευθύνη σε μια από τις υποκλάσεις της και πρέπει να εντοπιστεί η πληροφορία (της ευθύνης) στην συγκεκριμένη υποκλάση [2].
Στο επόμενο σχήμα φαίνεται το διάγραμμα κλάσεων του προτύπου μεθόδου εργοστασίου.



Σχήμα 3.2.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου μεθόδου εργοστασίου.

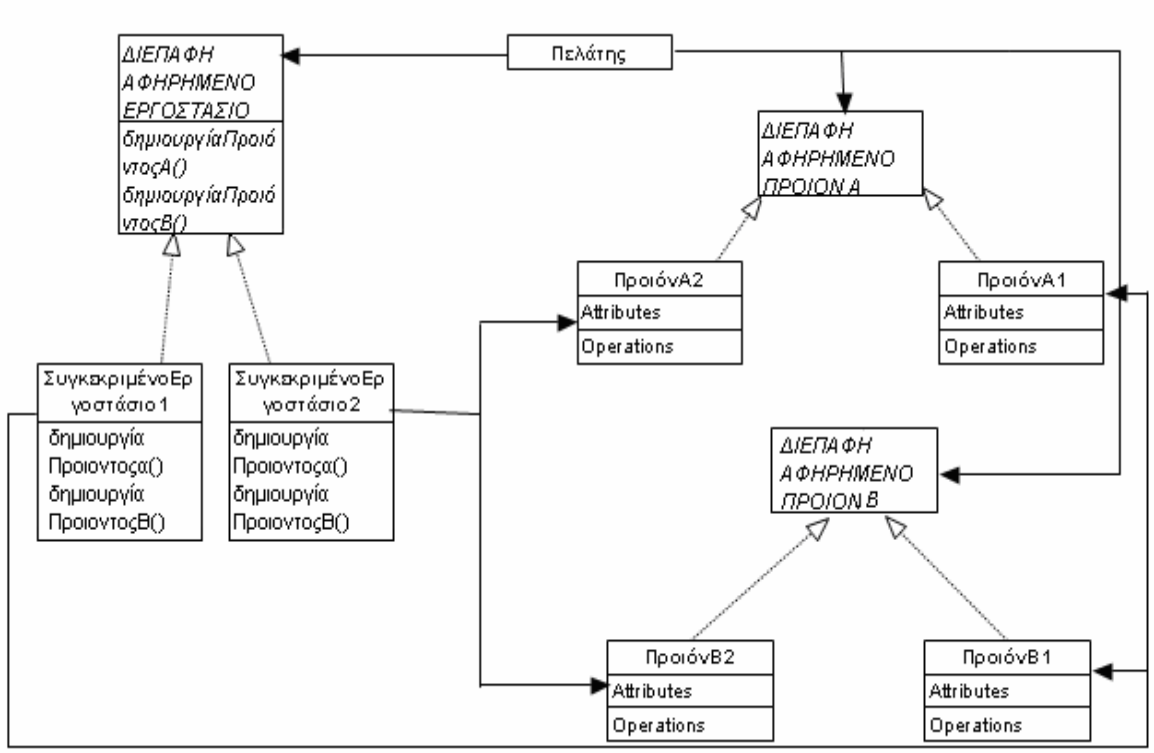
Τα συστατικά του σχήματος 3.2.1 είναι οι κλάσεις Προϊόν, ΣυγκεκριμένοΠροιόν, Δημιουργός και η ΣυγκεκριμένοςΔημιουργός.
1). Η κλάση Προϊόν ορίζει την διεπαφή των αντικειμένων, που δημιουργεί η μέθοδος εργοστασίου. Η κλάση ΣυγκεκριμένοΠροιόν εφαρμόζει την διεπαφή της κλάσης Προϊόν .
2). Η κλάση Δημιουργός δηλώνει τη μέθοδο εργοστασίου , η οποία επιστρέφει ένα αντικείμενο τύπου της κλάσης Προϊόν. Μπορεί να καλέσει επίσης και την μέθοδο εργοστασίου για να δημιουργήσει ένα αντικείμενο της κλάσης Προϊόν.
3). Η κλάση Δημιουργός βασίζεται στις υποκλάσεις του για να εφαρμόσει την μέθοδο εργοστασίου, η οποία επιστρέφει ένα στιγμιότυπο της κλάσης ΣυγκεκριμένοΠροιόν.
4}. Η κλάση ΣυγκεκριμένοςΔημιουργός αγνοεί την μέθοδο εργοστασίου και επιστρέφει ένα στιγμιότυπο της κλάσης ΣυγκεκριμένοΠροιόν.

Μερικά πλεονεκτήματα που έχει το πρότυπο είναι ότι δεν δεσμεύονται οι κλάσεις της εφαρμογής στον κώδικά . Επιπρόσθετα επιτρέπει στις υποκλάσεις να παρέχουν μια εκτεταμένη έκδοση ενός αντικειμένου, επειδή η δημιουργία ενός αντικειμένου μέσα σε μια κλάση είναι πιο ευέλικτη από τη δημιουργία ενός αντικείμενου άμεσα στον πελάτη [4].
Ένα μειονέκτημα όμως που έχει είναι ότι για να αρχικοποιηθεί έστω ένα ΣυγκεκριμένοΠροιόν, πρέπει να δημιουργηθεί υποκλάση της κλάσης Δημιουργός.
Στην εργασία με τίτλο ‘‘Implementing GoF Design Patterns in BETA’’ [24] εξετάστηκε πως η BETA υποστηρίζει ορισμένα σχεδιαστικά πρότυπα που αναφέρονται στο βιβλίο με τίτλο ‘‘Elements of Reusable Object-Oriented Software’’ [25]. Μερικά από αυτά τα πρότυπα είναι το πρότυπο μεθόδου εργοστασίου, το πρότυπο αφηρημένου εργοστασίου, το πρότυπο πρωτοτύπου και το πρότυπο singleton. Επίσης στην εργασία με τίτλο ‘‘The Difficulties of Using Design Patterns among Novices: An Exploratory Study ’’ [26] γίνονται κατανοητά μερικά σημεία όπου οι αρχάριοι κάνουν λάθη με τη χρήση των προτύπων. Εκτός από το πρότυπο μεθόδου εργοστασίου χρησιμοποιούνται και άλλα πρότυπα.
‘‘Γενικά οι μαθητές είναι πιο εξοικειωμένοι με τις αλγεβρικές δομές , ενώ κατανοούν δύσκολα τα σχεδιαστικά πρότυπα’’. Αυτό αναφέρεται στο έγγραφο ‘‘Teaching about Creational Design Patterns – General Implementations of an Algebraic Structure’’, όπου τα σχεδιαστικά πρότυπα υλοποιούνται με χρήση της άλγεβρας [27] .

3.3 Πρότυπο αφηρημένου εργοστασίου (Abstract Factory Pattern)

Το πρότυπο αφηρημένου εργοστασίου (Abstract Factory Pattern) είναι παρόμοιο με το πρότυπο μεθόδου εργοστασίου, αλλά στο αφηρημένο πρότυπο εργοστασίου ο στόχος της δημιουργίας αντικειμένου μεταβιβάζεται-εξουσιοδοτείται σε ένα διαφορετικό αντικείμενο, αντί να εξουσιοδοτείται σε μια υποκλάση. Το αντικείμενο, το οποίο εξουσιοδοτείται (το Αφηρημένο Εργοστάσιο), χρησιμοποιεί το πρότυπο μεθόδου εργοστασίου [7].
Το Πρότυπο αφηρημένου εργοστασίου (Abstract Factory Pattern) χρησιμοποιείται όταν ένα σύστημα πρέπει να είναι ανεξάρτητο από το πώς δημιουργούνται, συντίθενται και αναπαρίστανται τα προϊόντα τους. Επίσης χρησιμοποιείται όταν ένα σύστημα πρέπει να διαμορφώσει από ένα προϊόν από πολλές οικογένειες. Μια ακόμα

περίπτωση όπου το πρότυπο αυτό μπορεί να εφαρμοστεί, είναι όταν σχεδιάζεται μια οικογένεια από αντικείμενα τα οποία συσχετίζονται. Ακόμα, μπορεί να εφαρμοστεί όταν πρέπει να δημιουργηθεί μια κλάση που θα περιέχει ένα κατάλογο με προϊόντα, από τα οποία πρέπει να αποκαλύπτονται μόνοι οι διεπαφές τους και όχι οι εφαρμογές τους [5].
Ο πελάτης μπορεί να δημιουργεί μια συγκεκριμένη εφαρμογή από το αφηρημένο εργοστάσιο και μετά να χρησιμοποιεί τις διεπαφές για να δημιουργήσει τα συγκεκριμένα αντικείμενα που επιθυμεί. Δεν γνωρίζει ή δεν τον νοιάζει ποια συγκεκριμένα αντικείμενα παίρνει από κάθε ένα από τα εργοστάσια, αφού χρησιμοποιεί μόνο τις διεπαφές των προϊόντων τους [6] .
Στο επόμενο σχήμα παρουσιάζεται το διάγραμμα κλάσεων του προτύπου.



Σχήμα 3.3.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου αφηρημένου εργοστασίου.

Οι συμμετέχοντες στο σχήμα 3.3.1 είναι ο Πελάτης, η Διεπαφή ΑφηρημένοΕργοστάσιο, οι κλάσεις ΣυγκεκριμένοΕργοστάσιο1 και ΣυγκεκριμένοΕργοστάσιο2, οι Διεπαφές ΑφηρημένοΠροιόνΑ και ΑφηρημένοΠροιόνΒ και οι κλάσεις ΠροιόνΑ1, ΠροιόνΑ2, ΠροιόνΒ1 και ΠροιόνΒ2.
1). Ο Πελάτης χρησιμοποιεί τις διάφορες κλάσεις για να ζητήσει και να λάβει υπηρεσίες από το προϊόν με το οποίο ο πελάτης εργάζεται. Οι κλάσεις του πελάτη ξέρουν μόνο για τις αφηρημένες κλάσεις. Δεν πρέπει να γνωρίζουν τίποτα από τις συγκεκριμένες κλάσεις.
2). Η Διεπαφή ΑφηρημένοΕργοστάσιο καθορίζει τις αφηρημένες μεθόδους , οι οποίες θα δημιουργήσουν στιγμιότυπα των συγκεκριμένων κλάσεων προϊόντων. Οι αφηρημένες μέθοδοι αποτελούνται από ένα σύνολο μεθόδων για την παραγωγή των προϊόντων.

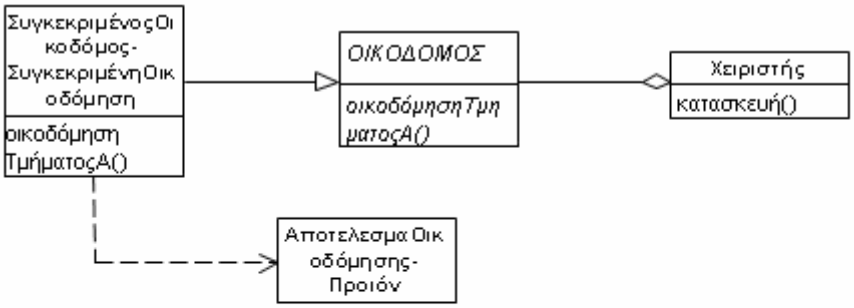
3). Οι κλάσεις ΣυγκεκριμένοΕργοστάσιο1 και ΣυγκεκριμένοΕργοστάσιο2 εφαρμόζουν τις μεθόδους που καθορίζονται από τις υπέρ-κλάσεις αφηρημένου εργοστασίου για να δημιουργήσουν στιγμιότυπα των συγκεκριμένων κλάσεων προϊόντων. Εφαρμόζουν τις διαφορετικές οικογένειες προϊόντων. Για να δημιουργηθεί ένα Προϊόν , ο Πελάτης χρησιμοποιεί ένα από αυτά τα συγκεκριμένα εργοστάσια , άρα δεν θα χρειαστεί ποτέ να αρχικοποιηθεί ένα αντικείμενο προϊόντος.
4). Οι ΔιεπαφέςΑφηρημένοΠροιονΑ και ΑφηρημένοΠροιονΒ είναι η οικογένεια των προϊόντων. Κάθε συγκεκριμένο εργοστάσιο μπορεί να παράγει ένα ολόκληρο σύνολο από προϊόντα. Οι κλάσεις που χρησιμοποιούν τις διεπαφές αυτές, εργάζονται με τα προϊόντα των διεπαφών. Δηλαδή στα προϊόντα που ανταποκρίνονται οι διεπαφές.

5). Οι κλάσεις ΠροιόνΑ1, ΠροιόνΑ2, ΠροιόνΒ1 και ΠροιόνΒ2 είναι συγκεκριμένες κλάσεις. Κάθε μια από αυτές ανταποκρίνεται σε ένα χαρακτηριστικό γνώρισμα του προϊόντος .

Το πρότυπο αφηρημένου εργοστασίου έχει ορισμένες επιπτώσεις – συνέπειες.
1). Απομονώνει τις συγκεκριμένες κλάσεις. Ο κώδικας του πελάτη χειρίζεται τα αντικείμενα μέσω της αφηρημένης διεπαφής τους. Δεν ξέρει πότε ο κώδικάς του εξετάζει τις συγκεκριμένες κλάσεις των αντικειμένων . Αυτό χαλαρώνει τη σύνδεση μεταξύ του κώδικα του πελάτη και των προϊόντων που χρησιμοποιεί.
2). Καθιστά την ανταλλαγή των προϊόντων των οικογενειών πιο εύκολη. Η κλάση ενός συγκεκριμένου εργοστασίου διευκρινίζεται μόνο σε μια θέση και έτσι μπορεί εύκολα να αλλάξει.
3). Μπορούν να αναμιχθούν τα αντικείμενα από τις διαφορετικές οικογένειες προϊόντων.
4). Η υποστήριξη νέων ειδών προϊόντων είναι δύσκολη. Για να προστεθεί ένα νέο προϊόν, μια νέα μέθοδος πρέπει να προστεθεί στην αφηρημένη κλάση εργοστασίων και σε όλες τις συγκεκριμένες υποκλάσεις της [7].

3.4 Πρότυπο οικοδόμησης- οικοδόμων (Builder Pattern)

Το Πρότυπο οικοδόμησης- οικοδόμων (Builder Pattern) διαχωρίζει την κατασκευή ενός σύνθετου αντικειμένου από την αναπαράσταση - απεικόνισή του. Έτσι, διαφορετικές αναπαραστάσεις μπορούν να δημιουργηθούν ανάλογα με τις ανάγκες του προγράμματος [8]. Το Πρότυπο οικοδόμησης - οικοδόμων χρησιμοποιείται, όταν η δημιουργία ενός σύνθετου προϊόντος πρέπει να είναι ανεξάρτητη από τα μέρη που αποτελούν το προϊόν. Το αποτέλεσμα μπορεί να έχει διαφορετικές αναπαραστάσεις [10]. Η διαδικασία δημιουργίας δεν πρέπει να γνωρίζει για τη διαδικασία συγκέντρωσης. Η διαδικασία συγκέντρωσης είναι ο τρόπος με τον οποίο τα μέρη τοποθετούνται μαζί για να αποτελέσουν το προϊόν. Επίσης χρησιμοποιείται σε περιπτώσεις όπου η διαδικασία δημιουργίας πρέπει να επιτρέψει διαφορετικές αναπαραστάσεις για το αντικείμενο που κατασκευάζεται.
Στο παρακάτω σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου οικοδόμων.



Σχήμα 3.4.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου οικοδόμων.

Τα μέλη του διαγράμματος είναι τα εξής: Η διεπαφή Οικοδόμος-Builder, η κλάση Χειριστής, το ΑποτέλεσμαΟικοδόμησης-Προϊόν και η κλάση ΣυγκεκριμένοςΟικοδόμος-ΣυγκεκριμένηΟικοδόμηση.
1). Η διεπαφή Οικοδόμος-Builder δημιουργεί τμήματα του αντικειμένου ΑποτέλεσμαΟικοδόμησης-Προϊόν.
2). Η κλάση ΣυγκεκριμένοςΟικοδόμος-ΣυγκεκριμένηΟικοδόμηση κατασκευάζει και συναρμολογεί τα τμήματα του προϊόντος, χρησιμοποιώντας τη Διεπαφή Οικοδόμος-Builder. Καθορίζει και παρακολουθεί τις αναπαραστάσεις που δημιουργεί. Επίσης παρέχει διεπαφή, μέσω της οποίας μπορεί να ξαναβρεί το προϊόν.
3). Ο Χειριστής κατασκευάζει ένα αντικείμενο, χρησιμοποιώντας τη διεπαφή Οικοδόμος-Builder.
4). Το ΑποτέλεσμαΟικοδόμησης-Προιόν αναπαριστά το σύνθετο υπό κατασκευή αντικείμενο. Η αναπαράσταση του αντικειμένου χτίζεται από την κλάση ΣυγκεκριμένοςΟικοδόμος-ΣυγκεκριμένηΟικοδόμηση. Η ίδια κλάση (ΣυγκεκριμένοςΟικοδόμος-ΣυγκεκριμένηΟικοδόμηση) καθορίζει επίσης την διαδικασία με την οποία θα συναρμολογηθεί. Περιλαμβάνει κλάσεις που καθορίζουν τα συστατικά μέρη, συμπεριλαμβάνοντας διεπαφές για τη συναρμολόγηση των συστατικών μερών- τμημάτων στο τελικό αποτέλεσμα.

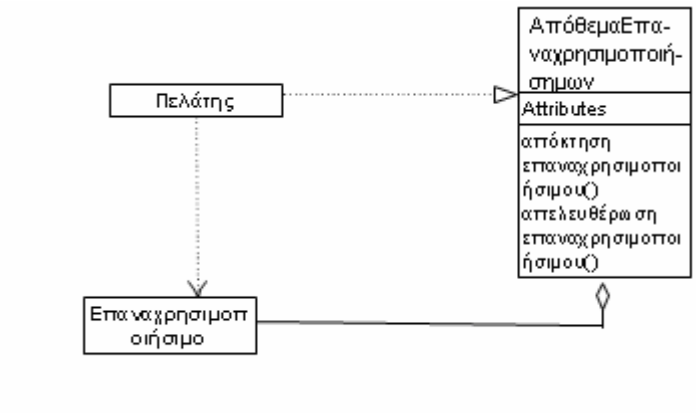
Για να γίνει πιο κατανοητό το προηγούμενο σχήμα θα εξηγηθεί πως συνεργάζονται τα συστατικά του σχήματος 4.3.1. Αρχικά ο Πελάτης δημιουργεί το αντικείμενο του Χειριστή και το διαμορφώνει σύμφωνα με το επιθυμητό αντικείμενο Οικοδόμου. Ο Χειριστής ειδοποιεί τον οικοδόμο πότε πρέπει να χτιστεί κάθε τμήμα του προϊόντος. Ο Οικοδόμος χειρίζεται αιτήσεις του χειριστή και προσθέτει τμήματα στο προϊόν. Τελικά ο Πελάτης ανακτά το προϊόν από τον οικοδόμο.

Το πρότυπο οικοδόμων έχει, ορισμένες συνέπειες. Μια σημαντική επίπτωση είναι ότι ένας οικοδόμος επιτρέπει να ποικίλει η εσωτερική αναπαράσταση του προϊόντος που χτίζει. Κρύβει επίσης τις λεπτομέρειες για το πώς το προϊόν συναρμολογείται. Άλλη μια συνέπεια είναι ότι κάθε συγκεκριμένος οικοδόμος είναι ανεξάρτητος από άλλους και από το υπόλοιπο του προγράμματος. Αυτό βελτιώνει τη διαμόρφωσή του και καθιστά την προσθήκη άλλων οικοδόμων σχετικά απλή. Υπάρχει και μια ακόμη συνέπεια, εξίσου σημαντική με τις δυο προηγούμενες. Σύμφωνα με αυτή επειδή κάθε

οικοδόμος κατασκευάζει το τελικό προϊόν σιγά-σιγά με βήματα ανάλογα με τα δεδομένα, ο χρήστης έχει περισσότερο έλεγχο κάθε τελικού προϊόντος που ένας οικοδόμος κατασκευάζει.
Ένα σχέδιο οικοδόμων μοιάζει κάπως με ένα αφηρημένο σχέδιο εργοστασίων, δεδομένου ότι και τα δυο επιστρέφουν κλάσεις φτιαγμένες από διάφορες μεθόδους και αντικείμενα. Η κύρια διαφορά είναι ότι ενώ το αφηρημένο εργοστάσιο επιστρέφει μια οικογένεια σχετικών κλάσεων, ο οικοδόμος κατασκευάζει ένα σύνθετο αντικείμενο με βήματα (βαθμιαία), ανάλογα με τα δεδομένα που παρουσιάζονται σε αυτόν.

3.5 Πρότυπο Αποθέματος Αντικειμένων (Object Pool Pattern)

Τα αντικείμενα αποθέματος, τα οποία είναι γνωστά και ως πόροι αποθέματος, χρησιμοποιούνται για να διαχειριστούν την αποθήκευση αντικειμένων. Ένας πελάτης ο οποίος έχει πρόσβαση σε ένα απόθεμα αντικειμένου, μπορεί να αποφύγει τη δημιουργία νέων αντικειμένων, απλά ρωτώντας το απόθεμα αν έχει ήδη αρχικοποιηθεί αυτό που ζητάει. Γενικά το απόθεμα θα είναι αυξανόμενο απόθεμα, δηλαδή το ίδιο το απόθεμα θα δημιουργήσει νέα αντικείμενα εάν το απόθεμα είναι κενό, ή μπορεί να υπάρχει ένα απόθεμα, το οποίο θα περιορίζει τον αριθμό αντικειμένων που δημιουργούνται.
Είναι επιθυμητό να κρατηθούν όλα τα επαναχρησιμοποιήσιμα αντικείμενα που δεν είναι αυτήν την περίοδο σε λειτουργία στο ίδιο απόθεμα αντικειμένου, έτσι ώστε να μπορούν να ρυθμιστούν. Για να επιτευχθεί αυτό, η επαναχρησιμοποιήσιμη κλάση αποθέματος σχεδιάζεται για να είναι μια κλάση Singleton [11]. (Βλέπε πρότυπο singleton 3.7)
Η γενική ιδέα του προτύπου είναι πως αν τα στιγμιότυπα μιας κλάσεως μπορούν να χρησιμοποιηθούν ξανά, γίνεται να αποφευχθεί η δημιουργία νέων στιγμιότυπων της κλάσης.
Στο σχήμα 3.5.1 απεικονίζεται το διάγραμμα κλάσεων του προτύπου αποθέματος αντικειμένου.



Σχήμα 3.5.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου αποθέματος αντικειμένου.

Οι συμμετέχοντες στο διάγραμμα κλάσεων είναι οι εξής: [13]

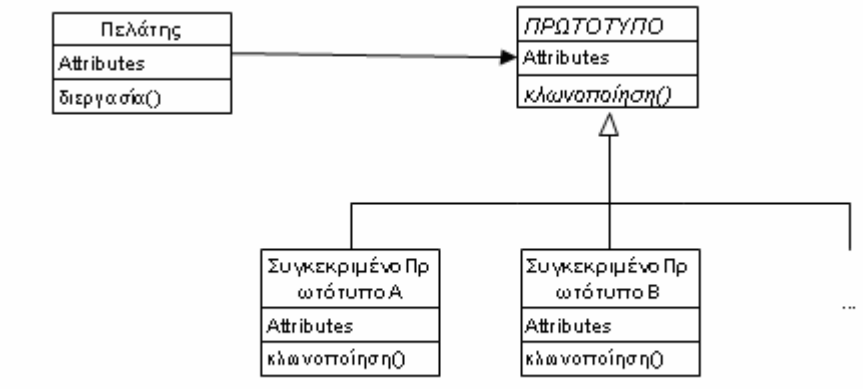
1). Ο Πελάτης, ο οποίος αποκτά και χρησιμοποιεί τους πόρους.

2). Η κλάση Επαναχρησιμοποιήσιμο είναι μια οντότητα, όπως μνήμη ή ένα νήμα. Τα αντικείμενα της κλάσης αυτής συνεργάζονται με άλλα αντικείμενα και μετά δεν είναι άλλο χρήσιμα για την συνεργασία αυτή.
3). Η κλάση ΑπόθεμαΕπαναχρησιμοποιήσιμων. Δημιουργεί στιγμιότυπα, τα οποία χειρίζονται Επαναχρησιμοποιήσιμα αντικείμενα που θα χειριστούν από τον Πελάτη (πιο συγκεκριμένα από αντικείμενα του Πελάτη).
Ο τρόπος με τον οποίο λειτουργεί το σχήμα 3.5.1 εξηγείται στις επόμενες τέσσερεις παραγράφους.
Συνήθως, είναι επιθυμητό να κρατηθούν όλα τα επαναχρησιμοποιήσιμα αντικείμενα που δεν είναι σε λειτουργία τώρα στο ίδιο απόθεμα αντικειμένου, έτσι ώστε να μπορούν να ρυθμιστούν από μια συνεπή πολιτική. Για να επιτευχθεί αυτό, η κλάση ΑπόθεμαΕπαναχρησιμοποιήσιμων σχεδιάζεται να είναι μια κλάση Singleton. Δηλαδή ο κατασκευαστής της είναι ιδιωτικός, το οποίο αναγκάζει άλλες κλασεις να καλέσουν τη μέθοδο λήψηΔεδομένωνSingleton του για να πάρουν ένα στιγμιότυπο της κλάσης ΑπόθεμαΕπαναχρησιμοποιήσιμου.
Ένα αντικείμενο του Πελάτη καλεί τη μέθοδο απόκτησηεπαναχρησιμοποιήσιμου ενός αντικειμένου της κλάσης ΑπόθεμαΕπαναχρησιμοποιήσιμου, όταν χρειάζεται ένα επαναχρησιμοποιήσιμο αντικείμενο. Ένα αντικείμενο της κλάσης ΑπόθεμαΕπαναχρησιμοποιήσιμο διατηρεί μια συλλογή των επαναχρησιμοποιήσιμων αντικειμένων. Χρησιμοποιεί τη συλλογή των επαναχρησιμοποιήσιμων αντικειμένων για να περιέχει μια ομάδα των επαναχρησιμοποιήσιμων αντικειμένων που δεν είναι αυτήν την περίοδο σε λειτουργία-χρήση [14].
Εάν υπάρχουν οποιαδήποτε επαναχρησιμοποιήσιμα αντικείμενα στο απόθεμα, όταν καλείται η μέθοδος απόκτησηεπαναχρησιμοποιήσιμου, αφαιρεί ένα επαναχρησιμοποιήσιμο αντικείμενο από το απόθεμα και το επιστρέφει. Εάν το απόθεμα είναι κενό, τότε η μέθοδος απόκτησηεπαναχρησιμοποιήσιμου δημιουργεί ένα επαναχρησιμοποιήσιμο αντικείμενο, εάν μπορεί. Εάν δεν μπορεί να δημιουργήσει ένα νέο επαναχρησιμοποιήσιμο αντικείμενο, τότε περιμένει έως ότου επιστραφεί ένα επαναχρησιμοποιήσιμο αντικείμενο στη συλλογή.
Τα αντικείμενα του Πελάτη περνούν ένα επαναχρησιμοποιήσιμο αντικείμενο στη μέθοδο απελευθέρωσηεπαναχρησιμοποιήσιμου ενός αντικειμένου της ΑπόθεμαΕπαναχρησιμοποιήσιμου όταν τελειώσουν με το αντικείμενο. Η μέθοδος απελευθέρωσηεπαναχρησιμοποιήσιμου επιστρέφει ένα επαναχρησιμοποιήσιμο αντικείμενο στην ομάδα των επαναχρησιμοποιήσιμων αντικειμένων που δεν είναι σε λειτουργία.

Σε πολλές εφαρμογές του προτύπου αποθέματος αντικειμένου, υπάρχουν λόγοι για το συνολικό αριθμό των επαναχρησιμοποιήσιμων αντικειμένων που μπορούν να υπάρξουν. Σε τέτοιες περιπτώσεις, το αντικείμενο της ΑπόθεμαΕπαναχρησιμοποιήσιμου που δημιουργεί τα επαναχρησιμοποιήσιμα αντικείμενα είναι αρμόδιο για την μη δημιουργία περισσότερων από έναν διευκρινισμένο μέγιστο αριθμό επαναχρησιμοποιήσιμων αντικειμένων. Εάν τα αντικείμενα της ΑπόθεμαΕπαναχρησιμοποιήσιμου είναι αρμόδια για τον περιορισμό του αριθμού αντικειμένων που θα δημιουργήσουν, κατόπιν η κλάση ΑπόθεμαΕπαναχρησιμοποιήσιμου θα έχει μια μέθοδο για να διευκρινίζει το μέγιστο αριθμό αντικειμένων που δημιουργούνται.

3.6 Πρωτότυπο πρότυπο (Prototype Pattern)

Το πρωτότυπο σημαίνει τη δημιουργία ενός κλώνου. Αυτό υπονοεί την κλωνοποίηση ενός αντικειμένου για να αποφύγει τη δημιουργία του αντικειμένου. Εάν το κόστος δημιουργίας ενός νέου αντικειμένου είναι μεγάλο και η δημιουργία απαιτεί πολλούς πόρους, τότε κλωνοποιείται το αντικείμενο [15].
Το πρότυπο πρωτότυπου (Prototype Pattern) χρησιμοποιείται σε καταστάσεις, όπου η δημιουργία ενός στιγμιότυπου μιας κλάσης είναι πολύ χρονοβόρα ή σύνθετη. Κατόπιν, αντί να δημιουργηθούν περισσότερα στιγμιότυπα, δημιουργούνται αντίγραφα του αρχικού στιγμιότυπου, που τροποποιείται ανάλογα με την περίπτωση. Τα πρωτότυπα μπορούν επίσης να χρησιμοποιηθούν όποτε χρειάζονται κλάσεις που διαφέρουν μόνο στον τύπο επεξεργασίας που προσφέρουν.
Η κλωνοποίηση (cloning) είναι η λειτουργία της αντιγραφής ενός αντικειμένου. Το κλωνοποιημένο αντικείμενο, το αντίγραφο, αρχικοποιείται με την τρέχουσα κατάσταση του αντικειμένου στο οποίο ο κλώνος επικαλέστηκε. Η κλωνοποίηση ενός αντικειμένου είναι βασισμένη στις έννοιες της ρηχής και βαθιάς αντιγραφής (shallow and deep cloning).
Η κλωνοποίηση έχει οριστεί ως: «Η κλωνοποίηση παράγει ένα ρηχό αντίγραφο ενός αντικειμένου , ένα αντίγραφο της πιο ακραίας δομής αντικειμένου , παίρνοντας όλες τις τιμές των χαρακτηριστικών ιδιοτήτων του αρχικού, αλλά με ανεξάρτητη κατάσταση. Η κλωνοποίηση είναι λίγο καινούρια, αλλά λειτουργεί στα αντικείμενα αντί στις κλάσεις. Οποιοδήποτε αντικείμενο ενεργεί και ως στιγμιότυπο και ως πρωτότυπο για την περαιτέρω κλωνοποίηση…» (Abadi & Cardelli, 1996) .
Η ρηχή αντιγραφή έχει την ακόλουθη επίδραση. Όταν το αρχικό αντικείμενο τροποποιείται, το νέο αντικείμενο αλλάζει επίσης. Αυτό οφείλεται στο γεγονός ότι το ρηχό αντίγραφο κάνει αντίγραφα των αναφορών μόνο, και όχι των αντικειμένων στα οποία αναφέρονται.
Η βαθιά αντιγραφή έχει την ακόλουθη επίδραση. Όταν το αρχικό αντικείμενο τροποποιείται, το νέο αντικείμενο δεν επηρεάζεται, δεδομένου ότι το ολόκληρο σύνολο αντικειμένων στο οποίο το αρχικό αντικείμενο αναφέρεται αντιγράφηκε επίσης [16].
Ο σχεδιαστής/ο υπεύθυνος για την ανάπτυξη μιας εφαρμογής έχει την ευθύνη να καθορίσει τον τύπο κλωνοποίησης που θα χρησιμοποιηθεί στην εφαρμογή. Ένα πολύ σοβαρό πρόβλημα μνήμης μπορεί να προκύψει εάν αυτό το θέμα αγνοηθεί.
Το διάγραμμα κλάσεων του προτύπου αυτού παρουσιάζεται στο επόμενο σχήμα.



Σχήμα 3.6.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου πρωτότυπου.

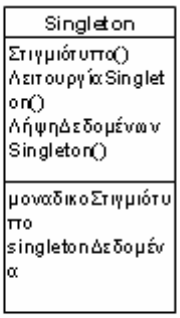
Τα συστατικά μέρη του σχήματος 3.6.1 είναι ο Πελάτης , η Διεπαφή ή αφηρημένη κλάση Πρωτότυπο και τα ΣυγκεκριμένοΠρωτότυποΑ και ΣυγκεκριμένοΠρωτότυποΒ.
1). Ο Πελάτης δημιουργεί ένα νέο αντικείμενο αναγκάζοντας τη διεπαφή Πρωτότυπο να κλωνοποιηθεί. Αυτό γίνεται με τη μέθοδο διεργασία. Μετά αλλάζει και διαμορφώνει τις λεπτομέρειες στο κλωνοποιημένο αντικείμενο, έτσι ώστε να συμπεριφέρεται όπως αυτός επιθυμεί [18].
2). Η διεπαφή Πρωτότυπο κλωνοποιεί τον εαυτό της χρησιμοποιώντας τη μέθοδο κλωνοποίηση.
3). Οι κλάσεις ΣυγκεκριμένοΠρωτότυποΑ και ΣυγκεκριμένοΠρωτότυποΒ εφαρμόζουν μια μέθοδο για να κλωνοποιήσουν τον εαυτό τους [17]. Η κλωνοποίηση γίνεται από την μέθοδο κλωνοποίηση.

Οι συνέπειες που έχει το πρότυπο πρωτότυπου είναι ότι μπορούν να προστεθούν και να αφαιρεθούν οι κλάσεις κατά τη διάρκεια εκτέλεσης, κλωνοποιώντας τις. Μπορεί να επανεξεταστεί και να αναθεωρηθεί η εσωτερική αναπαράσταση των δεδομένων μιας κλάσης κατά τη διάρκεια εκτέλεσης. Επίσης είναι δυνατόν να καθοριστούν νέα αντικείμενα κατά τη διάρκεια εκτέλεσης χωρίς να χρειάζεται να δημιουργηθούν δομές κληρονομικότητας ή άλλες δομές [16].

3.7 Πρότυπο Singleton (Singleton Pattern)

Μερικές κλάσεις πρέπει να έχουν ακριβώς ένα στιγμιότυπο. Αυτές οι κλάσεις περιλαμβάνουν συνήθως την κεντρική διαχείριση ενός πόρου. Ο πόρος μπορεί να είναι εξωτερικός, όπως συμβαίνει με ένα αντικείμενο που διαχειρίζεται την επαναχρησιμοποίηση των συνδέσεων βάσεων δεδομένων. Ο πόρος μπορεί να είναι εσωτερικός, όπως ένα αντικείμενο που κρατά μια αρίθμηση λάθους και στατιστικά στοιχεία για έναν μεταγλωττιστή [20]. Το πρότυπο Singleton διαβεβαιώνει ότι θα δημιουργηθεί ένα στιγμιότυπο μιας κλάσης [9]. Όλα τα αντικείμενα που χρησιμοποιούν ένα στιγμιότυπο της κλάσης, θα χρησιμοποιήσουν το ίδιο στιγμιότυπο.
Γενικά το πρότυπο Singleton χρησιμοποιείται όταν πρέπει να υπάρξει ακριβώς ένα στιγμιότυπο μιας κλάσης, και πρέπει να είναι προσιτό στους πελάτες από ένα γνωστό σημείο πρόσβασης. Επίσης, όταν το μοναδικό στιγμιότυπο πρέπει να επεκταθεί και οι πελάτες πρέπει να είναι σε θέση να χρησιμοποιήσουν ένα στιγμιότυπο που έχει επεκταθεί χωρίς να τροποποιήσουν τον κώδικά τους [21].

Το πρότυπο Singleton είναι σχετικά απλό, δεδομένου ότι περιλαμβάνει μόνο μια κλάση. Στο σχήμα 3.7.1 απεικονίζεται η κλάση Singleton του προτύπου.



Σχήμα 3.7.1 Απεικόνιση του προτύπου Singleton.

Ερμηνεία του σχήματος 3.7.1.
1). Ο πελάτης έχει πρόσβαση σε ένα στιγμιότυπο μέσω της μεθόδου ΛειτουργίαSingleton. Η κλάση Singleton είναι υπεύθυνη για να δημιουργεί το δικό της μοναδικό στιγμιότυπο. Η μέθοδος Στιγμιότυπο επιστρέφει ένα μοναδικόΣτιγμιότυπο, όταν το ζητήσει ο πελάτης. Για να είναι βέβαιο ότι θα δημιουργηθεί μόνο ένα στιγμιότυπο η μέθοδος Στιγμιότυπο είναι στατική.

Για να γίνει κατανοητή η σημασία του προτύπου Singleton θα παρουσιαστεί ένα παράδειγμα. Έστω ότι πρέπει να δημιουργηθεί μια κλάση, η οποία θα διαβεβαιώνει ότι μόνο ένα τραγούδι θα παίζεται από μια εφαρμογή. Υπάρχει περίπτωση η εφαρμογή να παίξει δύο τραγούδια ταυτόχρονα, κάτι το οποίο δεν είναι επιθυμητό. Επομένως, για να αποφευχθεί η κατάσταση, να γίνεται αναπαραγωγή δύο τραγουδιών ταυτόχρονα , πρέπει να βρεθεί μια λύση. Η λύση αυτή είναι η εξής. Η κλάση που θα δημιουργηθεί πρέπει να σταματάει την αναπαραγωγή του πρώτου τραγουδιού, όταν αρχίζει η αναπαραγωγή του δεύτερου τραγουδιού. Ένας τρόπος είναι να σχεδιαστεί μια κλάση να εφαρμόζει αυτήν την προϋπόθεση, είναι να εξασφαλιστεί ότι υπάρχει μόνο ένα στιγμιότυπο της κλάσης κοινό σε όλα τα αντικείμενα που χρησιμοποιούν εκείνη την κλάση. Εάν όλα τα αιτήματα που ζητούν να αρχίσουν ένα τραγούδι περνούν από το ίδιο αντικείμενο, τότε είναι απλό για το αντικείμενο να σταματήσει το τελευταίο τραγούδι που άρχισε, πριν αρχίζει το επόμενο.

Αναφορές :

[1]: http://en.wikipedia.org/wiki/Creational_pattern
[2]: Dhananjay Indurkar,(2004*), Factory Method*, Winter 2004: CS590L Distributed Component Architecture, Workshop II Design Patterns.
[3]: Raphael Enns, (1999), *Design Patterns and CodePro 1, 2* , Canadian Institute of Health Research Design Grant Program Social Sciences and Humanities Research Council of Canada, Canada Health Services Research Foundation.
[4]: http://www.apwebco.com/gofpatterns/creational/FactoryMethod.html
[5]: Raman Ramsin, (2007), *Patterns in Software Engineering, Design Patterns,* Creational Department of Computer Engineering Sharif University of Technology.
[6]: Sang Shin, (2008), *Java Patterns* , Java Technology Architect Sun Microsystems, Inc.Sun Tech Days 2008-2009.
[7]: Mel Ó Cinnéide, (1999), *The Abstract Factory Pattern,* Proceedings, IEEE International Conference on A methodology for the automated introduction of design patterns 1999 pp.463 – 472.
[8]: Kayun Chantarasathaporn, and Chonawat Srisa-an, *Energy Conscious Builder Design Pattern with C# and Intermediate Language*, PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY VOLUME 13 MAY 2006 ISSN 1307-6884.
[9]: Eivind J. Nordby Karlstad, (2007), *Singleton creational design Pattern,* Proceedings of the 9th IEEE International Conference on Engineering of Computer-Based Systems pp. 118.
[10]: Mark J.Sebern, (4/2/00), *Builder Design Pattern,* SE281-S00-09.
[11]: http://sourcemaking.com/design_patterns/object_pool
[12]: http://www.oodesign.com/object-pool-pattern.html
[13]: Michael Kircher,(2002), *Pooling,* Prashant Jain Corporate Technology, Siemens AG, Munich, Germany 2002.
[14]: http://www.developer.com/java/ent/article.php/626171
[15]: http://www.allapplabs.com/java_design_patterns/prototype_pattern.htm
[16]: Grogono, P. and Sakkinen, M. (2000). Copying and Comparing: Problems and Solutions. European Conference on Object Oriented Programming and Applications (SEA 2000).

[17]: Woei-Kae Chen, (2003), *Object- Oriented Programming Prototype Pattern* , 27th Annual International Computer Software and Applications Conference, COMPSAC 2003, Dallas, Texas.
[18]: Gabriel Becerra, (2004), *A Discussion on the Prototype*, Softawre Engineering Research Network SENG 609.04 Design Patterns.
[19]:http://en.wikipedia.org/wiki/Singleton_pattern
[20]: Mark Grand, (2002), *Patterns in Java, Volume 1—A Catalog of Reusable Design Patterns Illustrated with UML, Second Edition* –TEAM FLY Wiley.
[21]: Raman Ramsin , (2007), *Patterns in Software Engineering, Design Patterns,* Creational Department of Computer Engineering Sharif University of Technology.
[22]: Jan Hannemann and Gregor Kiczales, (2002), *Design Pattern Implementation in Java and AspectJ*, OOPSLA '02 ACM SIGPLAN Notices Volume 37 , Issue 11 (November 2002)
SESSION: Aspects, pp.161 – 173.
[23]: Lerina Aversano, Gerardo Canfora, Luigi Cerulo, Concettina Del Grosso, Massimiliano Di Penta, (2007), *An Empirical Study on the Evolution of Design Patterns*, Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2007, Dubrovnik, Croatia, September 3-7, 2007. ACM 2007.
[24]: Implementing GoF Design Patterns in BETA Ellen Agerbo and Aino Cornils Dept of Computer Science, Aarhus University, Denmark
[25]: E. Gamma, R. Helm, R. Johnson, J. Vlissides (1995), *Elements of Reusable Object- Oriented Software (GoF 95*), Addison-Wesley Publishing Company.
[26]: Masita Abdul Jalil and Shahrul Azman Mohd Noah, (2007), *The Difficulties of Using Design Patterns among Novices: An Exploratory Study*, Fifth International Conference on Computational Science and Applications 2007.
[27]: Virginia Niculescu, (2003), *Teaching about Creational Design Patterns – General Implementations of an Algebraic Structure*, Symbolic and Numeric Algorithms for Scientific Computing, 2005. SYNASC 2005. Seventh International Symposium on 25-29 Sept. 2005 pp.8.

4. ΔΟΜΙΚΑ ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ
Structural Design Patterns

4.1 Σύντομη αναφορά των δομικών σχεδιαστικών προτύπων.

Τα δομικά σχεδιαστικά πρότυπα περιγράφουν, τον τρόπο με τον οποίο οι κλάσεις και τα αντικείμενα συντίθενται μεταξύ τους για να δημιουργήσουν μεγαλύτερες δομές [2]. Οι κλάσεις των δομικών προτύπων χρησιμοποιούν την κληρονομικότητα για να συνθέσουν διεπαφές. Τα αντικείμενα των δομικών προτύπων, όμως περιγράφουν τρόπους, με τους οποίους θα συντεθούν τα αντικείμενα, με στόχο να δημιουργηθεί μια νέα λειτουργία.
Στο σχεδιασμό λογισμικού, τα δομικά πρότυπα δίνουν τη δυνατότητα στο σχεδιαστή λογισμικού να ενισχύσει τις κλάσεις. Δηλαδή η συμπεριφορά ή η λειτουργία μπορεί να ποικίλει με το χειρισμό της δομής των κλάσεων που συμμετέχουν.

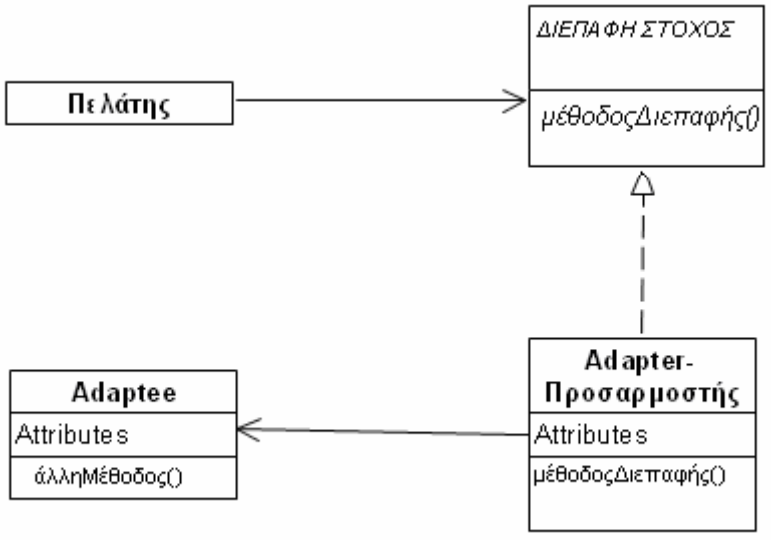
Για παράδειγμα, ένα μεγάλο εμπορικό κέντρο, εξωτερικά μοιάζει σαν ένα ενιαίο κτίριο, ενώ στην πραγματικότητα αποτελείται από διαφορετικά ανεξάρτητα καταστήματα. Αν κάποιος πελάτης δει το κτίριο από έξω , θα το δει σαν ένα κτίριο, ενώ, αν το δει από την εσωτερική πλευρά του κτιρίου, θα δει τα επιμέρους καταστήματα [6].
Τα πρότυπα που ανήκουν σε αυτήν την κατηγορία είναι τα Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Dynamic Linkage και Virtual Proxy.
Τα πρότυπα αυτά βοηθούν να γίνουν κατανοητά ορισμένα ερωτήματα, όπως το πώς επεκτείνεται μια ιεραρχία που υλοποιεί ένα στοιχείο αφαίρεσης. Αυτό επιτυγχάνεται με τα πρότυπα Decorator, Adapter και Proxy. Επίσης απαντά στην ερώτηση, πώς συσχετίζεται μια κλάση ή μια ιεραρχία με μια ή περισσότερες άλλες που χρησιμοποιούνται για την υλοποίηση των πρώτων. Εδώ συμμετέχουν τα πρότυπα γέφυρας (Bridge Pattern) και πρόσοψης (Façade Pattern). Σύμφωνα με τη συμμορία τεσσάρων (GoF), ο σκοπός του προτύπου πρόσοψης (Façade Pattern) είναι να παρέχει μια ενοποιημένη διεπαφή σε ένα σύνολο διεπαφών σε ένα υποσύστημα [18]. Η πρόσοψη καθορίζει μια υψηλότερου επιπέδου διεπαφή που καθιστά το υποσύστημα ευκολότερο να χρησιμοποιηθεί. Μπορεί με τη χρήση του προτύπου Flyweight να βοηθήσει αποτελεσματικά στη βελτιστοποίηση της σχεδίασης των κλάσεων μιας ιεραρχίας [1].
Το πρότυπο προσαρμοστή (Adapter Pattern) λύνει το πρόβλημα της επέκτασης μιας υπάρχουσας ιεραρχία κλάσεων που υλοποιούν ένα συγκεκριμένο στοιχείο αφαίρεσης με την προσθήκη μιας νέας κλάσης. Η διεπαφή της νέας κλάσης δεν είναι συμβατή με το υπάρχον στοιχείο αφαίρεσης που υποθέτει η υπάρχουσα ιεραρχία.
Το αντικείμενο Flyweight είναι ένα αντικείμενο που ελαχιστοποιεί την απαιτούμενη μνήμη με τη διανομή όσο το δυνατόν περισσότερων δεδομένων με άλλα παρόμοια αντικείμενα. Είναι ένας τρόπος να χρησιμοποιηθούν τα αντικείμενα σε μεγάλους αριθμούς με μικρότερο ποσό μνήμης, από όσο θα χρειαζόταν χωρίς το πρότυπο αυτό. Συχνά μερικά τμήματα της κατάστασης αντικειμένου μπορούν να μοιραστούν και να τεθούν σε εξωτερικές δομές δεδομένων και να περαστούν στα αντικείμενα Flyweight προσωρινά όταν χρησιμοποιούνται [32].

4.2 Πρότυπο Προσαρμοστή-Μετασχηματιστή (Adapter Pattern)

Γενικά ο προσαρμοστής-μετασχηματιστής χρησιμοποιείται, όταν πρέπει οι ανεξάρτητες κλάσεις να λειτουργούν μαζί σε ένα ενιαίο πρόγραμμα. Αυτές οι ανεξάρτητες κλάσεις δεν θα μπορούσαν να λειτουργήσουν μαζί, επειδή οι διεπαφές είναι ασυμβίβαστες μεταξύ τους. Ο μετασχηματιστής είναι γνωστός και ως περιτύλιγμα- κάλυμμα (Wraper) [3] .
Το πρότυπο του μετασχηματιστή επιτρέπει να μετατραπούν/προσαρμοστούν οι υπάρχουσες διεπαφές σε κλάσεις. Αν δεν γίνει αυτό οι διεπαφές δεν θα μπορούν να συνεργαστούν. Τα αντικείμενα που χρησιμοποιούν αυτό το πρότυπο λέγονται περιτυλίγματα (wrappers), αφού τυλίγουν ένα αντικείμενο σε μια νέα διεπαφή. Υπάρχουν πολλές περιπτώσεις όπου η δημιουργία μετασχηματιστών μπορεί να βοηθήσει και τους σχεδιαστές και τους προγραμματιστές [6].
Για παράδειγμα, ένα κινητό τηλέφωνο στις Ηνωμένες Πολιτείες Αμερικής λειτουργεί με φορτιστή των 110Volt. Στην Ελλάδα λειτουργεί με φορτιστή των 220Volt. Άρα , για να χρησιμοποιηθεί ο φορτιστής των 110Volt στην Ελλάδα πρέπει να χρησιμοποιηθεί κάποιος προσαρμοστής-μετασχηματιστής . Ο μετασχηματιστής αυτός θα μετατρέπει τα 110Volt σε 220Volt [4].

Όπως αναφέρθηκε και στην παράγραφο 4.1, το πρότυπο μετασχηματιστή (Adapter Pattern) λύνει το πρόβλημα της επέκτασης μιας υπάρχουσας ιεραρχία κλάσεων που υλοποιούν ένα συγκεκριμένο στοιχείο αφαίρεσης με την προσθήκη μιας νέας κλάσης. Η διεπαφή της νέας κλάσης δεν είναι συμβατή με το υπάρχον στοιχείο αφαίρεσης που υποθέτει η υπάρχουσα ιεραρχία. Η λύση που ακολουθεί το πρότυπο είναι η εξής. Για κάθε κλάση μη συμβατή με το υπάρχον στοιχείο αφαίρεσης, υλοποιείται μια κλάση μετασχηματιστή. Η κλάση αυτή είναι συμβατή με το υπάρχον στοιχείο αφαίρεσης.
Το πρότυπο μετασχηματιστών περιγράφει πώς ένα αντικείμενο μπορεί να έχει έναν πελάτη, που περιμένει το αντικείμενο αυτό, για να εφαρμόσει μια διεπαφή, ακόμα κι αν δεν εφαρμόζει τη διεπαφή αυτή.
Μια κλάση μετασχηματιστή μπορεί να υλοποιηθεί χρησιμοποιώντας δυο διαφορετικές προσεγγίσεις [7]. Η πρώτη είναι με τη χρήση της κλάσης μετασχηματιστή. Εδώ η κλάση κατασκευάζεται ως υποκλάση της κλάσης, της μη συμβατής κλάσης. Κλήσεις στις μεθόδους της κλάσης μεταφράζονται σε κλήσεις σε μεθόδους που κληρονομούνται από την μη συμβατή κλάση. Η δεύτερη είναι με τη χρήση του αντικειμένου μετασχηματιστή. Σε αυτήν την περίπτωση, στην κλάση δηλώνονται αναφορές σε (περιτυλιγμένο) αντικείμενο της μη συμβατής κλάσης. Κλήσεις στις μεθόδους της κλάσης μεταφράζονται σε κλήσεις σε μεθόδους του περιτυλιγμένου αντικείμενου. Χρησιμοποιείται συνήθως όταν η μη συμβατή κλάση δεν επιτρέπει τη δημιουργία υποκλάσεων.

Έστω ότι υπάρχει μια κλάση που καλεί μια μέθοδο μέσω μιας διεπαφής. Ένα στιγμιότυπο αυτής της κλάσης πρέπει να καλεί μια μέθοδο ενός αντικειμένου που δεν εφαρμόζει τη διεπαφή. Μπορεί το στιγμιότυπο να καλεί μέσω ενός αντικειμένου μετασχηματιστή, το οποίο εφαρμόζει τη διεπαφή , εξουσιοδοτώντας τις κλήσεις σε μια μέθοδο του αντικειμένου που δεν εφαρμόζει τη διεπαφή. Το επόμενο σχήμα δείχνει το διάγραμμα κλάσεων του προτύπου μετασχηματιστή [9] [10].



Σχήμα 4.2.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου προσαρμοστή-μετασχηματιστή.

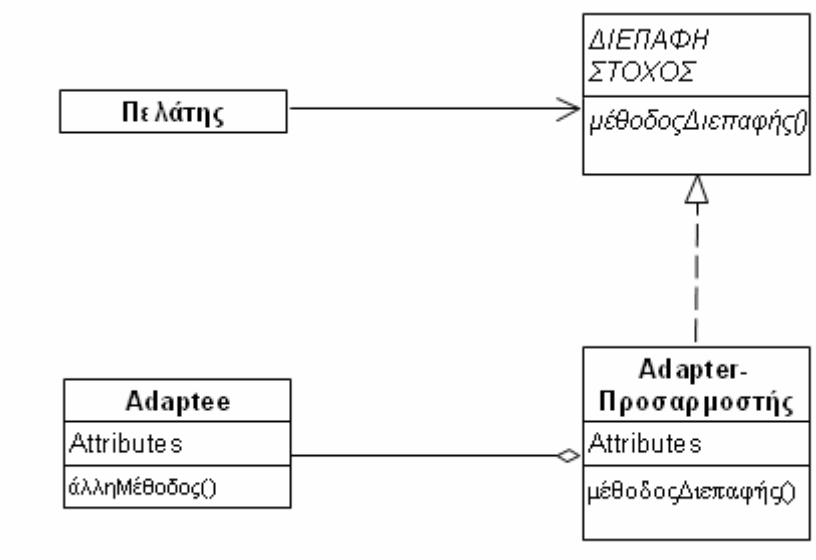
Εδώ είναι οι ρόλοι που παίζουν οι κλάσεις και η διεπαφή στο σχήμα 4.2.1

1). Πελάτης: Μια κλάση σε αυτόν τον ρόλο καλεί μια μέθοδο μιας άλλης κλάσης μέσω μιας διεπαφής για να αποφύγει την πραγματική κλάση που εφαρμόζει τη μέθοδο. Επίσης συνεργάζεται με αντικείμενα που προσαρμόζονται στην Διεπαφή-Στόχος.
2). Διεπαφή- Στόχος: Μια διεπαφή σε αυτόν τον ρόλο δηλώνει μια μέθοδο που καλεί η κλάση του πελάτη.
3). Adapter (προσαρμοστής): Μια κλάση σε αυτόν τον ρόλο εφαρμόζει τη διεπαφή Στόχος. Εφαρμόζει τη μέθοδο που ο Πελάτης καλεί με την εξουσιοδότηση της κλήσης σε μια μέθοδο της κλάσης Adaptee, η οποία δεν εφαρμόζει τη διεπαφή Στόχος.
4). Adaptee: Μια κλάση σε αυτόν τον ρόλο δεν εφαρμόζει τη μέθοδο της διεπαφής, αλλά έχει μια μέθοδο που η κλάση πελατών θέλει να καλέσει [9].

Είναι δυνατό για μια κλάση προσαρμοστή να κάνει περισσότερα, από απλώς τον εκπρόσωπο μιας κλήσης μεθόδου. Μπορεί να εκτελέσει κάποιο μετασχηματισμό. Μπορεί επίσης, να παρέχει την πρόσθετη λογική για να κρύψει τις διαφορές μεταξύ της προοριζόμενης σημασίας της μεθόδου της διεπαφής και της πραγματικής σημασίας της μεθόδου της κλάσης Adaptee. Δεν υπάρχει κανένα όριο στο πόσο σύνθετη μια κλάση προσαρμοστή μπορεί να είναι.

Σχετικά με τη δεύτερη προσέγγιση, ο προσαρμοστής αντικειμένου μετατρέπει τη διεπαφή μιας κλάσης Adaptee σε μια άλλη διεπαφή . Αντί της κληρονομικότητας, ο ίδιος ο προσαρμοστής δημιουργεί ένα στιγμιότυπο κλάσης Adaptee για να ολοκληρώσει την προσαρμογή.

Στη εικόνα 4.2.2 απεικονίζεται το αντικείμενο του προσαρμοστή.



Σχήμα 4.2.2 Απεικόνιση του αντικείμενου του προσαρμοστή- μετασχηματιστή.

Το αντικείμενο του προσαρμοστή - μετασχηματιστή χρησιμοποιείται γενικά όταν πρέπει να αποσυνδεθούν εντελώς ο πελάτης και η κλάση Adaptee. Με αυτήν την μορφή προσαρμοστή μόνο η κλάση Προσαρμοστή κατανοείται και από τον Πελάτη

και από την διεπαφή Στόχος. Η πολλαπλή κληρονομικότητα απαιτείται σε κάθε περίπτωση, για το πρότυπο προσαρμοστή αντικειμένου [8].
Όταν η κλάση Πελάτης καλεί οποιαδήποτε από τις μεθόδους της Διεπαφής – Στόχου, τότε η κλήση μεταφράζεται σε συγκεκριμένη κλήση σε κάθε στιγμιότυπο αντικειμένου της κλάσης
Adaptee.
Η συσχέτιση που υπάρχει μεταξύ του μετασχηματιστή και της κλάσης Adaptee είναι συνάθροιση. Το στιγμιότυπο της Adaptee γίνεται ένα μέρος του μετασχηματιστή. Δεδομένου ότι ο μετασχηματιστής κρατά το στιγμιότυπο της κλάσης Adaptee, ο πελάτης δεν εκτίθεται σε όλες τις μεθόδους της Adaptee. Με αυτόν τον τρόπο ο μετασχηματιστής αποσυνδέει τον πελάτη από την κλάση Adaptee.

Όσον αφορά και τις δυο προσεγγίσεις που αναφέρθηκαν, αυτή η εφαρμογή χρησιμοποιεί κληρονομικότητα μεταξύ του προσαρμοστή και της Adaptee, δεδομένου ότι τα μέλη κληρονομούνται από την πατρική κλάση και δεν επαναπροσδιορίζονται στην υποκλάση. Οι συνδέσεις των δεδομένων στις κλάσεις γονέων και παιδιών είναι σφιχτές.
Οι υπέρ-κλάσεις και υποκλάσεις που συμμετέχουν δεν μπορούν να επαναχρησιμοποιηθούν χωριστά. Επίσης υπάρχει μια σχέση μεταξύ του πελάτη και του μετασχηματιστή, δεδομένου ότι οποιεσδήποτε αλλαγές στον μετασχηματιστή έχουν επιπτώσεις στον πελάτη. Οποιαδήποτε αναθεώρηση- αλλαγή του σχεδιασμένου λογισμικού του μετασχηματιστή θα οδηγήσει σε αλλαγές και στον πελάτη.
Το πιο σημαντικό όφελος του προτύπου αυτού είναι ότι ο μετασχηματιστής είναι ένα εργαλείο που προσαρμόζεται εύκολα για τη σύνδεση μιας ήδη υπάρχουσας κλάσης με μια άλλη κλάση. Αυτό μπορεί να είναι πολύ χρήσιμο για τις κλάσεις που εφαρμόζουν κάποιο πολύπλοκο αλγόριθμο για την βέλτιστη συμπεριφορά στο χρόνου εκτέλεσης. Ο κώδικας μετασχηματιστή αποτελείται συνήθως από σύντομες και απλές μεθόδους [5].

4.3 Πρότυπο Γέφυρας (Bridge Pattern)

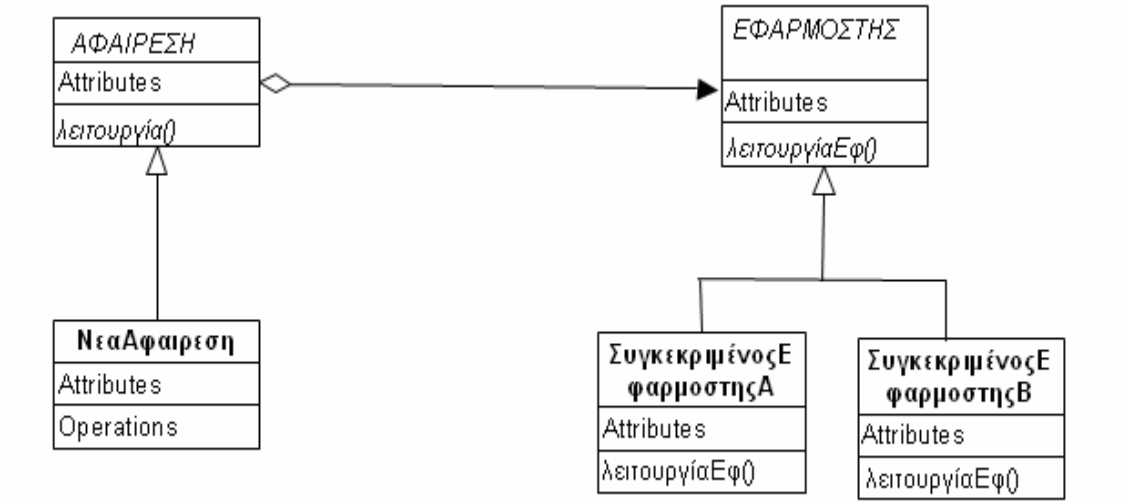
Σύμφωνα με τη συμμορία τεσσάρων (GoF) [18], «ο σκοπός του προτύπου γέφυρας είναι να αποσυνδέσει μια αφαίρεση από την εφαρμογή της, έτσι ώστε τα δύο να μπορούν να ποικίλουν ανεξάρτητα» .

Αποσύνδεση ή διαχωρισμός (decoupling or separating) σημαίνει να υπάρχουν αντικείμενα τα οποία συμπεριφέρονται ανεξάρτητα το ένα από το άλλο. Η αφαίρεση είναι πώς τα διαφορετικά αντικείμενα συσχετίζονται το ένα με το άλλο εννοιολογικά. Οι εφαρμογές είναι ο τρόπος με το οποίο χτίζονται οι αφαιρέσεις. Αλλά οι εφαρμογές εδώ σημαίνουν ότι τα αντικείμενα που παράγει η αφηρημένη κλάση, εφαρμόζουν τον εαυτό τους [17].
Διαχωρισμός μιας αφαίρεσης από την εφαρμογή της σημαίνει, διαχωρισμός μιας ποικίλης οντότητα από μια ποικίλη συμπεριφορά, έτσι ώστε να μπορούν να ποικίλουν ανεξάρτητα [19]. Είναι ένα πολύ ισχυρό πρότυπο που χρησιμοποιείται σε πολλές καταστάσεις. Το πρότυπο γέφυρας είναι χρήσιμο για παράδειγμα, όταν πρέπει να αποφευχθεί μια μόνιμη σύνδεση μεταξύ μιας αφαίρεσης και της εφαρμογής της [20].

Το πρότυπο γέφυρας είναι χρήσιμο όταν υπάρχει μια ιεραρχία αφαιρέσεων και μια αντίστοιχη ιεραρχία εφαρμογών. Αντί να συνδυάζονται οι αφαιρέσεις με τις

εφαρμογές σε πολλές κλάσεις, το πρότυπο γέφυρας εφαρμόζει τις αφαιρέσεις και τις εφαρμογές ως ανεξάρτητες κλάσεις που μπορούν να συνδυαστούν δυναμικά [9].
Το πρότυπο αυτό προτείνει επίσης μια λύση στην περίπτωση κατά την οποία το σύστημα περιλαμβάνει στοιχεία αφαίρεσης, από τα οποία μπορεί να προκύψουν διαφορετικές υλοποιήσεις, αλλά και νέα στοιχεία αφαίρεσης, τα οποία μπορεί επίσης να έχουν διαφορετικές υλοποιήσεις.
Η λύση του προβλήματος έγκειται στη δημιουργία δυο ιεραρχιών. Η πρώτη είναι ιεραρχία υψηλού επιπέδου και συσχετίζει στοιχεία αφαίρεσης, μέσω κληρονομικότητας. Η δεύτερη είναι ιεραρχία χαμηλού επιπέδου και σχετίζει μέσω κληρονομικότητας κλάσεις που χρησιμοποιούνται για την υλοποίηση των κλάσεων της πρώτης ιεραρχίας. Οι μέθοδοι των κλάσεων της ιεραρχίας υψηλού επιπέδου κάνουν χρήση αποκλειστικά των μεθόδων που ορίζονται στο βασικό στοιχείο της ιεραρχίας χαμηλού επιπέδου.
Η σύνδεση μεταξύ των δυο ιεραρχιών γίνεται κατασκευάζοντας μια ‘‘γέφυρα’’, δηλαδή το βασικό στοιχείο αφαίρεσης της ιεραρχίας υψηλού επιπέδου έχει μια αναφορά στο βασικό στοιχείο της ιεραρχίας χαμηλού επιπέδου. Άρα, οποιαδήποτε κλάση της ιεραρχίας υψηλού επιπέδου μπορεί να κάνει χρήση αντικειμένων οποιασδήποτε κλάσης της ιεραρχίας χαμηλού επιπέδου. Επίσης κατά τη διάρκεια εκτέλεσης μπορούν να γίνονται αλλαγές.

Στο σχήμα 4.3.1 απεικονίζεται το διάγραμμα κλάσεων του προτύπου γέφυρας.



Σχήμα 4.3.1 Απεικόνιση του διαγράμματος κλάσεων του προτύπου γέφυρας

Εδώ είναι οι περιγραφές των ρόλων που παίζουν οι κλάσεις στην οργάνωση που παρουσιάζεται στο σχήμα 4.3.1.
1). Αφαίρεση : Αυτή η κλάση αντιπροσωπεύει την κορυφαία αφαίρεση. Η Αφαίρεση καθορίζει την διεπαφή για τα αντικείμενα που εφαρμόζονται. Είναι υπεύθυνη για τη διατήρηση μιας αναφοράς σε ένα αντικείμενο που εφαρμόζει τη διεπαφή Εφαρμοστής-Implementor, έτσι ώστε μπορεί να εξουσιοδοτήσει τις διαδικασίες στην εφαρμογή του.
2). Η Εφαρμοστής-Implementor καθορίζει την διεπαφή για συγκεκριμένες εφαρμογές κλάσεων. Οι κλάσεις προήλθαν από Αφαίρεση, χρησιμοποιούν κλάσεις που προήλθαν από την Εφαρμοστής χωρίς να γνωρίζουν ποια κλάση ΣυγκεκριμένοςΕφαρμοστής είναι σε λειτουργία. Αυτή η διεπαφή δηλώνει τις

μεθόδους για όλες τις χαμηλού επιπέδου διαδικασίες που μια εφαρμογή πρέπει να παρέχει για την κλάση αφαίρεσης.
3). Οι κλάσεις ΣυγκεκριμένοςΕφαρμοστήςΑ και ΣυγκεκριμένοςΕφαρμοστήςΒ εφαρμόζουν την διεπαφή Εφαρμοστής, καθορίζοντας η κάθε μια, μια συγκεκριμένη εφαρμογή.
4). ΝεαΑφαίρεση: Αυτός ο ρόλος αντιστοιχεί σε οποιαδήποτε υποκλάση της κλάσης αφαίρεσης. Για κάθε τέτοια υποκλάση της κλάσης αφαίρεσης υπάρχει μια αντίστοιχη υπο-διεπαφή της διεπαφής Εφαρμοστής-Implementor.

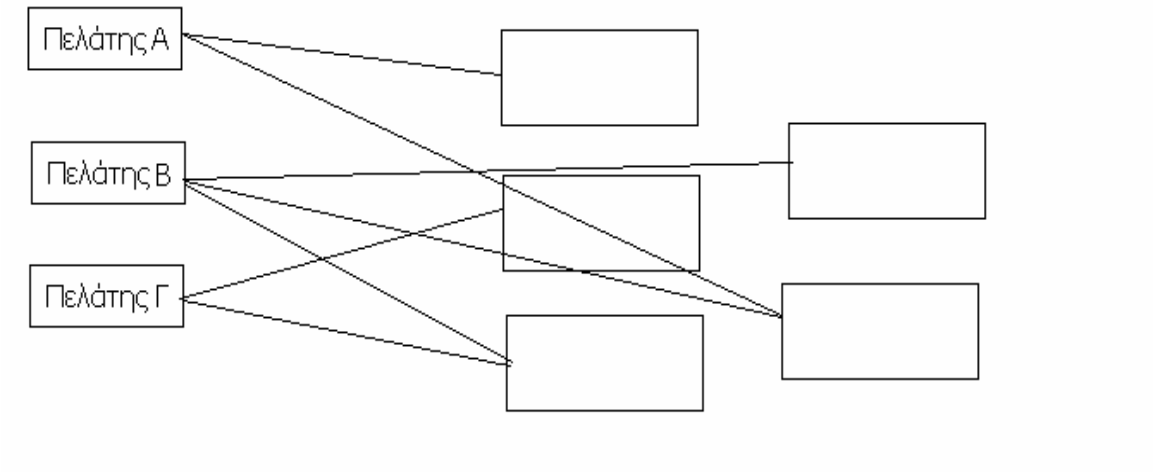
Οι συνέπειες του προτύπου γέφυρας είναι οι εξής:
1. Το πρότυπο αυτό προορίζεται να διατηρήσει τη διεπαφή στο πρόγραμμα του πελάτη σταθερή επιτρέποντας να αλλάξει το είδος κλάσης χρησιμοποιείται. Αυτό μπορεί να αποτρέψει το χρήστη από ένα σύνολο περίπλοκων ενοτήτων ενδιάμεσων με τον χρήστη.
2. Μπορεί να επεκταθούν οι κλάσεις εφαρμογής χωρίς μεγάλη αλληλεπίδραση η μια με την άλλη.
3. Μπορούν να μείνουν κρυφές οι λεπτομέρειες εκτέλεσης από το πρόγραμμα του πελάτη εύκολα.

Το πρότυπο γέφυρας είναι παρόμοιο με το πρότυπο στρατηγικής (Strategy Pattern) και το πρότυπο Template. Είναι ένα είδος συνδυασμού των δυο αυτών προτύπων. Επίσης, το εν λόγω πρότυπο χρησιμοποιεί συχνά το πρότυπο προσαρμοστή (Adapter pattern). Το πρότυπο γέφυρας χρησιμοποιείται για να χωρίσει τη διεπαφή της κλάσης από την εφαρμογή, έτσι ώστε καθεμία να μπορεί να ποικίλει χωριστά. Μοιάζει με το πρότυπο προσαρμοστή (Adapter), δεδομένου ότι μια κλάση χρησιμοποιείται για να μετατρέψει ένα είδος διεπαφής σε ένα άλλο. Εντούτοις, η πρόθεση του προσαρμοστή είναι να γίνουν μια ή περισσότερες διεπαφές να φανούν ίδιες με μια συγκεκριμένη κλάση. Το πρότυπο γέφυρας σχεδιάζεται για να χωρίσει τη διεπαφή μιας κλάσης από την εφαρμογή της, έτσι ώστε να υπάρχει η δυνατότητα να αντικατασταθεί ή να διαφοροποιηθεί η εφαρμογή, χωρίς αλλαγή του κώδικα του πελάτη.
Το πρότυπο γέφυρας μπορεί να επεκταθεί στο πρότυπο Cascading Bridge.Το πρότυπο Cascading Bridge είναι η σύνδεση των γεφυρών σε μια αλυσίδα, έτσι ώστε κάθε συστατικό έχει ευθύνη για ένα συγκεκριμένο σύνολο μετασχηματισμών, και μεταβιβάζει την υπόλοιπη εργασία στο επόμενο συστατικό στην αλυσίδα [21].

4.4 Πρότυπο Πρόσοψης (Facade pattern)

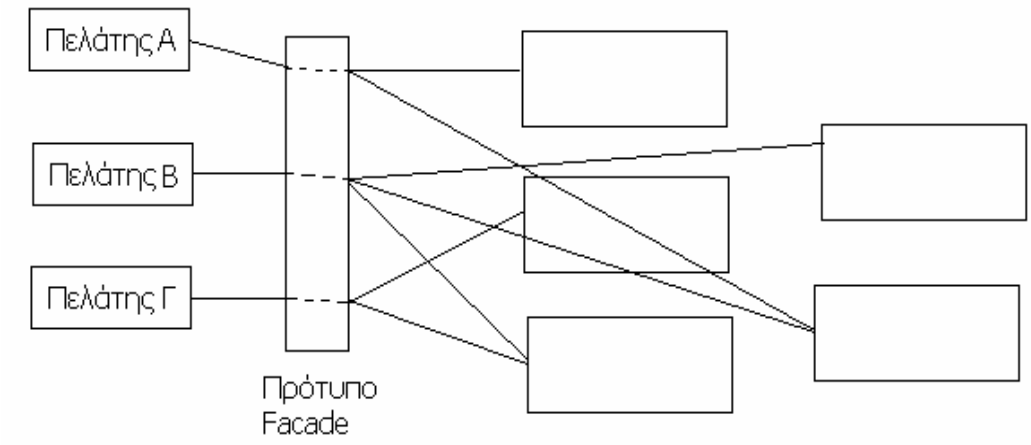
Συχνά, καθώς τα προγράμματά εξελίσσονται και αναπτύσσονται, αυξάνεται η πολυπλοκότητα τους. Στην πραγματικότητα, τα πρότυπα παράγουν μερικές φορές τόσες πολλές κλάσεις που είναι δύσκολο να καταλάβει κανείς τη ροή του προγράμματος. Επιπλέον, μπορούν να υπάρξουν διάφορα περίπλοκα υποσυστήματα. Το πρότυπο πρόσοψης (Façade Pattern) επιτρέπει να απλοποιηθεί αυτή η πολυπλοκότητα, με την παροχή μιας απλουστευμένης διεπαφής σε αυτά τα υποσυστήματα. Αυτή η απλοποίηση μπορεί σε μερικές περιπτώσεις να μειώσει την ευελιξία των κλάσεων, αλλά συνήθως παρέχει όλη τη λειτουργία που απαιτείται για να γίνει πιο κατανοητή η όλη κατάσταση [9].
Γενικά, το πρότυπο πρόσοψης έχει δυο λειτουργίες. Σύμφωνα με την πρώτη λειτουργία, απλοποιεί την διεπαφή μιας κλάσης. Η δεύτερη λειτουργία έχει να κάνει την αποσύνδεση της κλάσης αυτής από τον κώδικα του πελάτη που την χρησιμοποιεί. Το πρότυπο αυτό μπορεί να δημιουργήσει μεθόδους, οι οποίες επιτρέπουν να

χρησιμοποιηθούν σύνθετα συστήματα πιο εύκολα και πιο απλά. Οι προσόψεις (facades) δίνουν την δυνατότητα στους προγραμματιστές να αλληλεπιδράσουν έμμεσα με υποσυστήματα, με τρόπο που επιφέρει λιγότερα λάθη, αντί να αλληλεπιδράσουν με τα υποσυστήματα άμεσα.
Απλοποιεί εργασίες που επαναλαμβάνονται, όπως καταγραφή λαθών. Επιτρέπει να εμφανίζονται αντικείμενα πιο πλήρη από ότι είναι , προσθέτοντας νέες μεθόδους. Οι προσόψεις (facades) απλοποιούν σύνθετες διεπαφές. Μπορούν επίσης να κάνουν έλεγχο λαθών κρυφά και να ξεκαθαρίσουν αντικείμενα που δεν χρειάζονται άλλο [6].
Με άλλα λόγια, αυτό το πρότυπο παρέχει στο λογισμικό του πελάτη μια απλούστερη άποψη ενός σύνθετου υποσυστήματος [11]. Δεν βοηθά να γίνει η εκτέλεση πιο απλή, αλλά να γίνει πιο απλό και πιο κατανοητό ένα σύνθετο σύστημα. Στο επόμενο σχήμα 4.4.1 φαίνεται ένα σύνθετο υποσύστημα, στο οποίο οι πελάτες χρησιμοποιούν τα υποσυστήματα στα δεξιά τους [22] [25]. Αντί για υποσυστήματα θα μπορούσαν να υπάρχουν κλάσεις.



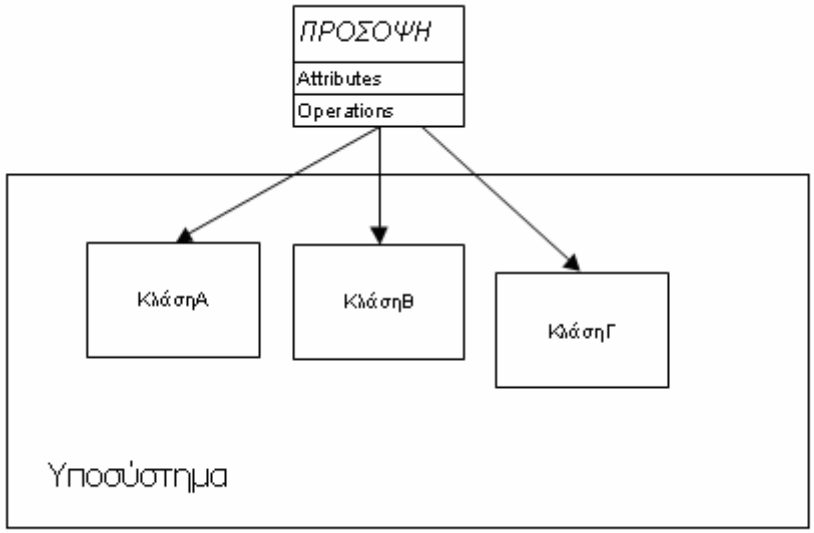
Σχήμα 4.4.1 Απεικόνιση ενός σύνθετου υποσυστήματος.

Στο σχήμα 4.4.2 απεικονίζεται το ίδιο υποσύστημα, με την διαφορά ότι έχει χρησιμοποιηθεί το πρότυπο πρόσοψης. Σε αυτό το σχήμα το υποσύστημα είναι φανερά πιο απλό σε σχέση με την απεικόνιση στο σχήμα 4.4.1.



Σχήμα 4.4.2 Απεικόνιση του σύνθετου υποσυστήματος της εικόνας 4.4.1 με χρήση του προτύπου πρόσοψης (Façade Pattern).

Το πρότυπο πρόσοψης παρέχει μια ενοποιημένη διεπαφή σε ένα σύνολο διεπαφών σε ένα υποσύστημα [24]. Η πρόσοψη καθορίζει μια υψηλότερου επιπέδου διεπαφή που καθιστά το υποσύστημα ευκολότερο να χρησιμοποιηθεί. Ο κώδικας του πελάτη απλοποιείται και οι εξαρτήσεις του πελάτη μειώνονται πολύ. Επίσης μια πρόσοψη όχι μόνο απλοποιεί μια διεπαφή, αποσυνδέει έναν πελάτη από ένα υποσύστημα συστατικών [23] [25]. Στο σχήμα 4.4.3 απεικονίζεται το διάγραμμα κλάσεων του προτύπου πρόσοψης .



Σχήμα 4.4.3 Απεικόνιση του διαγράμματος κλάσεων του προτύπου πρόσοψης (Façade Pattern)

Οι συμμετέχοντες στο διάγραμμα κλάσεων είναι η διεπαφή Πρόσοψη και οι κλάσεις του υποσυστήματος.
Οι κλάσεις του υποσυστήματος εφαρμόζουν λειτουργίες του υποσυστήματος και η Πρόσοψη γνωρίζει ποιες κλάσεις του υποσυστήματος είναι υπεύθυνες για κάθε λειτουργία [12]. Η Πρόσοψη εκπροσωπεί τα αιτήματα του πελάτη στα κατάλληλα αντικείμενα των κλάσεων υποσυστήματος.

Για να γίνει πιο κατανοητό το πρότυπο , έστω ότι υπάρχει ένα πολύπλοκο σύνολο κλάσεων, οι λειτουργίες των οποίων μπορούν να συνδυαστούν με πολλούς και διάφορους τρόπους. Για να χρησιμοποιήσει αυτές τις κλάσεις ένας προγραμματιστής πρέπει να δαπανήσει αρκετό χρόνο και κόπο ώστε να μελετήσει τις λειτουργίες τους. Επίσης ο κώδικας που συνδυάζει τις λειτουργίες των κλάσεων αυτών θα είναι διάχυτος μέσα στο σύστημα. Αυτά δυσχεραίνουν αρκετά τη διαδικασία υλοποίησης και τη διαδικασία συντήρησης ενός συστήματος.
Το πρότυπο πρόσοψης προτείνει μια λύση για την κατάσταση αυτή. Η λύση περιλαμβάνει την κατασκευή μιας κλάσης Façade που προσφέρει μεθόδους υψηλού επιπέδου, οι οποίες συνδυάζουν λειτουργίες κλάσεων ενός ή περισσοτέρων υποσυστημάτων ενός συστήματος. Οι συνδυασμοί των λειτουργιών αυτών απαντώνται συχνά στην πράξη και προσφέρουν μια πιο εύχρηστη διεπαφή στα εν λόγω υποσυστήματα.

Και το πρότυπο πρόσοψης και το πρότυπο προσαρμοστή-μετασχηματιστή μπορούν να τυλίξουν τις πολλές κλάσεις, αλλά ο σκοπός μιας πρόσοψης είναι να απλοποιήσει, ενώ του προσαρμοστή- μετασχηματιστή είναι να μετατρέψει μεταξύ των διεπαφών [23].
Στην εργασία με τίτλο ‘‘Wrapper Facade A Structural Pattern for Encapsulating Functions within Classes ’’ παρουσιάζεται ένα πρότυπο συναφές με το πρότυπο πρόσοψης. Ο σκοπός του προτύπου αυτού είναι να ενθυλακώσει χαμηλού επιπέδου μεθόδους και δομές δεδομένων με αντικειμενοστραφή διεπαφές [13].

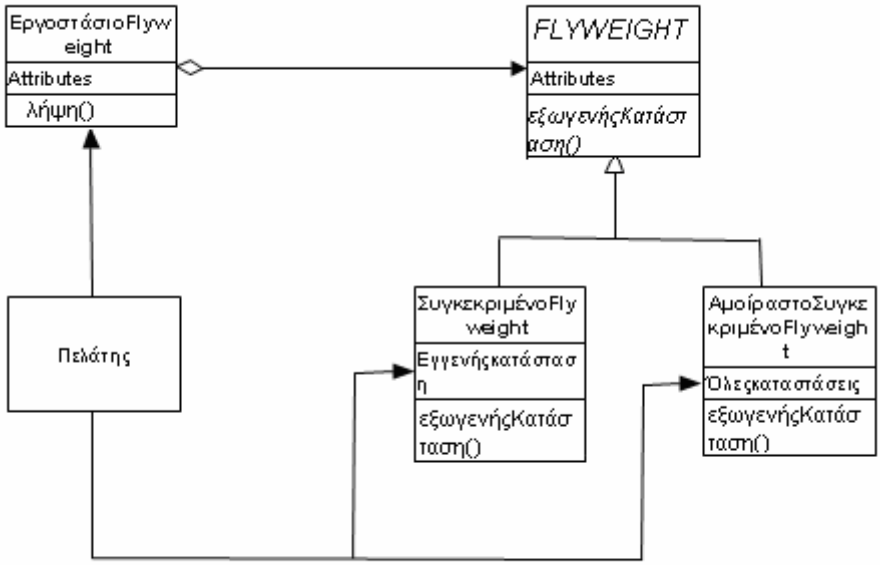
4.5 Πρότυπο Flyweight (Flyweight pattern)

Εάν ένα πρόγραμμα δημιουργεί πάρα πολλά αντικείμενα, μπορεί να καταναλώνει πάρα πολλή μνήμη. Το πρότυπο Flyweight βοηθά να μη δημιουργηθούν ισοδύναμα αντικείμενα περισσότερο από μία φορά. Το όνομα Flyweight δείχνει ότι πάρα πολλά αντικείμενα, ακόμα κι αν είναι μικρά, μπορούν να προκαλέσουν το πρόβλημα στη χρήση μνήμης [14].
Το πρότυπο αυτό χρησιμοποιείται όταν ισχύουν όλες οι παρακάτω προϋποθέσεις [15]:
1). Μια εφαρμογή χρησιμοποιεί έναν μεγάλο αριθμό αντικειμένων.
2). Οι δαπάνες αποθήκευσης είναι υψηλές, λόγω του μεγάλου αριθμού των αντικειμένων.
3). Η κατάσταση των περισσότερων αντικειμένων μπορεί να γίνει εξωγενής.
4). Πολλές ομάδες αντικειμένων μπορούν να αντικατασταθούν από λίγα κοινά αντικείμενα μόλις αφαιρεθεί η εξωγενής κατάσταση.
5). Η εφαρμογή δεν εξαρτάται από την ταυτότητα αντικειμένου. Δεδομένου ότι τα flyweight αντικείμενα μπορούν να μοιραστούν, οι δοκιμές ταυτότητας (identity tests) θα επιστρέψουν αληθινό (true) για τα εννοιολογικά ευδιάκριτα αντικείμενα.

Υπάρχουν περιπτώσεις στον προγραμματισμό όπου φαίνεται ότι πρέπει να παραχθεί ένας πολύ μεγάλος αριθμός μικρών αντικειμένων κλάσεων. Μερικές φορές μπορεί να μειωθεί πολύ ο αριθμός των διαφορετικών κλάσεων που πρέπει να αρχικοποιηθούν εάν είναι γνωστό ότι τα αντικείμενα που δημιουργούνται είναι τα ίδια εκτός από μερικές παραμέτρους.
Το flyweight είναι ένα αντικείμενο που μπορεί να χρησιμοποιηθεί σε πολλά πλαίσια ταυτόχρονα. Ενεργεί ως ανεξάρτητο αντικείμενο σε κάθε ένα πλαίσιο. Τα Flyweights δεν μπορούν να κάνουν υποθέσεις για το πλαίσιο στο οποίο λειτουργούν. Η βασική έννοια είναι ο διαχωρισμός της κατάστασης του αντικειμένου σε εγγενή (intrinsic) και εξωγενή (extrinsic) κατάσταση του αντικειμένου.
Η εγγενής κατάσταση αποθηκεύεται στο αντικείμενο flyweight και αποτελείται από πληροφορίες που είναι ανεξάρτητες από το πλαίσιο flyweight, καθιστώντας το κοινόχρηστο. Δηλαδή στην εγγενή περίπτωση τα εγγενή δεδομένα είναι οι πληροφορίες που απαιτούνται από τις εσωτερικές μεθόδους μιας κλάσης. Η κλάση αυτή δεν μπορεί να λειτουργήσει σωστά χωρίς αυτά τα στοιχεία.
Η εξωγενής κατάσταση εξαρτάται και διαφέρει από το πλαίσιο flyweight και επομένως δεν μπορεί να μοιραστεί αλλού. Δηλαδή τα εξωγενή δεδομένα είναι πληροφορίες που μπορούν να αφαιρεθούν από μια κλάση και να αποθηκευτούν εξωτερικά.
Είναι δυνατόν να συγκεντρωθούν όλα τα αντικείμενα που έχουν την ίδια εγγενή κατάσταση και να αντικαθιστούν με ένα ενιαίο κοινό αντικείμενο. Κατά συνέπεια

μειώνεται ο αριθμός των αντικειμένων στον αριθμό των μοναδικών εγγενών καταστάσεων που υπάρχουν.

Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου Flyweight.



Σχήμα 4.5.1 Απεικόνιση του διαγράμματος κλάσεων του προτύπου Flyweight.

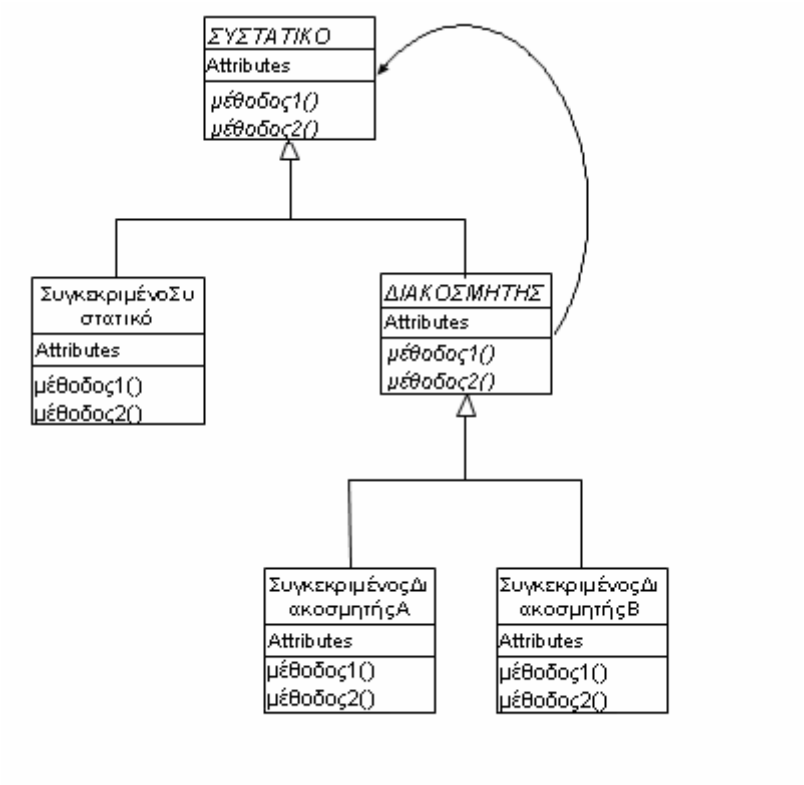
1). Flyweight: δηλώνει μια διεπαφή μέσω της οποίας τα αντικείμενα flyweights μπορούν να λάβουν και να ενεργήσουν στην εξωγενή κατάσταση. Είναι η υπερ-κλάση όλων των άλλων κλάσεων flyweight. Καθορίζει τις μεθόδους κοινές για τις κλάσεις flyweight. Αυτές οι μέθοδοι απαιτούν την πρόσβαση στις εξωγενείς πληροφορίες κατάστασης. Οι μέθοδοι αυτοί έχουν πρόσβαση στις εξωγενείς πληροφορίες κατάστασης μέσω των παραμέτρων.
2). ΣυγκεκριμένοFlyweight: εφαρμόζει τη Flyweight διεπαφή και αποθηκεύει την εγγενή κατάσταση του κοινού αντικειμένου. Το αντικείμενο αυτό πρέπει να είναι κοινόχρηστο [31].
3). ΕργοστάσιοFlyweight: δημιουργεί και διαχειρίζεται flyweight αντικείμενα. Τα αντικείμενα της κλάσης ΕργοστάσιοFlyweight παρέχουν αντικείμενα της κλάσης AbstractFlyweight στα αντικείμενα πελατών. Εάν ένα αντικείμενο πελάτη ζητά ένα αντικείμενο ΕργοστάσιοFlyweight για να παρέχει ένα αντικείμενο μιας κλάσης ΑμοιραστοΣυγκεκριμένοFlyweight, δημιουργεί απλά το αντικείμενο. Εντούτοις, εάν ένα αντικείμενο πελάτη ζητά ένα ΕργοστάσιοFlyweight αντικείμενο για να παρέχει ένα αντικείμενο μιας κλάσης ΣυγκεκριμένοFlyweight, πρώτα πρέπει να ελέγξει εάν δημιούργησε προηγουμένως ένα παρόμοιο αντικείμενο. Εάν ναι, παρέχει το προηγουμένως δημιουργημένο αντικείμενο στον πελάτη. Διαφορετικά, δημιουργεί ένα νέο αντικείμενο και το παρέχει στον πελάτη.
4). Πελάτης: διατηρεί μια αναφορά στα flyweights υπολογίζει ή αποθηκεύει την εξωγενή κατάσταση των flyweights.
5). ΑμοιραστοΣυγκεκριμένοFlyweight: εφαρμόζει τη Flyweight διεπαφή. Εντούτοις, δεν πρέπει να μοιραστούν όλες οι Flyweight υποκλάσεις. Η flyweight διεπαφή επιτρέπει τη διανομή, δεν την επιβάλλει [30]. Τα αντικείμενα των κλάσεων που συμμετέχουν στο ρόλο ΑμοιραστοΣυγκεκριμένοFlyweight δεν είναι κοινόχρηστες.

Το πρότυπο Flyweight δεν απαιτεί τη διανομή των αντικειμένων. Επιτρέπει απλά τη διανομή των αντικειμένων.

Η εγγενής και εξωγενής κατάσταση πρέπει να είναι σαφώς διακριτές. Τα Flyweight αντικείμενα αποθηκεύουν την εγγενή κατάσταση, οι πελάτες αποθηκεύουν ή υπολογίζουν την εξωγενή κατάσταση και την παρέχουν σε flyweight διαδικασίες. Οι πελάτες δεν δημιουργούν αντικείμενα flyweight άμεσα. Ένα εργοστάσιο flyweight σιγουρεύει ότι τα flyweight αντικείμενα μοιράζονται σωστά.

4.6 Πρότυπο Διακοσμητή (Decorator pattern)

Το Πρότυπο Διακοσμητή (Decorator pattern) παρέχει έναν τρόπο να τροποποιηθεί η συμπεριφορά μεμονωμένων αντικείμενων χωρίς να πρέπει να δημιουργηθεί μια νέα κλάση [9]. Προσθέτει ευθύνες δυναμικά σε ένα αντικείμενο και όχι σε ολόκληρες κλάσεις. Αυτό σημαίνει ότι οι ‘‘διακοσμήσεις’’ μπορούν να γίνουν στο χρόνο εκτέλεσης της εφαρμογής (σε runtime) και δεν χρειάζεται να αλλαχτεί ο κώδικας των αντικειμένων [34] [38]. Επίσης δεν επηρεάζονται άλλα αντικείμενα, εκτός από αυτό που προσθέτονται οι ευθύνες [33] [35].
Στην πληροφορική, ο χρόνος εκτέλεσης ή runtime περιγράφει τη λειτουργία ενός προγράμματος υπολογιστή, τη διάρκεια της εκτέλεσής του, από την αρχή μέχρι τη λήξη του [39].
Το πρότυπο διακοσμητή διατηρεί τη δομή, ενώ έχει τη δυνατότητα να κάνει τις αλλαγές με τη ‘‘διακόσμηση’’ των διαφορετικών συστατικών που συμπληρώνουν την εφαρμογή. Οι διακοσμήσεις αποτελούνται από περιγραφές ή και μεθόδους που χρησιμοποιούνται για να τυλίξουν τα διαφορετικά αντικείμενα στην εφαρμογή. Το πρότυπο αυτό επιτρέπει να αναμιχτούν και να συνδυαστούν διαφορετικά συστατικά και διακοσμήσεις για να βελτιστοποιηθεί η ευελιξία και η επεκτασιμότητα, ενώ οι κλάσεις παραμένουν αμετάβλητες.
Το σχήμα 4.6.1 είναι ένα διάγραμμα κλάσεων που παρουσιάζει γενική δομή του προτύπου διακοσμητή.



Σχήμα 4.6.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου διακοσμητή.

Εδώ είναι οι περιγραφές των ρόλων που παίζουν οι κλάσεις και οι διεπαφές στο πρότυπο διακοσμητή.

1). Συστατικό: Είναι η διεπαφή για αντικείμενα στα οποία μπορούν να προστεθούν δυναμικά διάφορες ευθύνες. Μια διεπαφή σε αυτόν τον ρόλο εφαρμόζεται από όλα τα αντικείμενα που μπορούν να διακοσμηθούν-επεκταθούν μέσω του προτύπου διακοσμητή.
2). ΣυγκεκριμένοΣυστατικό: Ορίζει ένα αντικείμενο στο οποίο μπορούν να προστεθούν ευθύνες.
3). Διακοσμητής: Εφαρμόζει την ίδια διεπαφή ή αφηρημένη κλάση, όπως το συστατικό που θέλει να διακοσμήσει. Κάθε διακοσμητής (τυλίγει-wraps) ένα συστατικό, δηλαδή διατηρεί μια αναφορά σε ένα αντικείμενο συστατικού.
4). ΣυγκεκριμένοςΔιακοσμητήςΑ και ΣυγκεκριμένοςΔιακοσμητήςΒ: Προσθέτουν ευθύνες στο συστατικό. Μπορεί να υπάρχουν περισσότερες από δυο τέτοιες κλάσεις.

Το πρότυπο αυτό παρουσιάζει δύο σημαντικά στοιχεία : συστατικού και διακοσμητή. Το συστατικό αντιπροσωπεύει αυτό που πρέπει να διακοσμηθεί, και ο διακοσμητής είναι η αφηρημένη κλάση για τις συγκεκριμένες διακοσμήσεις. Επίσης το συγκεκριμένο συστατικό είναι τι είναι πραγματικά διακοσμημένο, και οι συγκεκριμένες διακοσμήσεις είναι οι πραγματικές διακοσμήσεις [37].

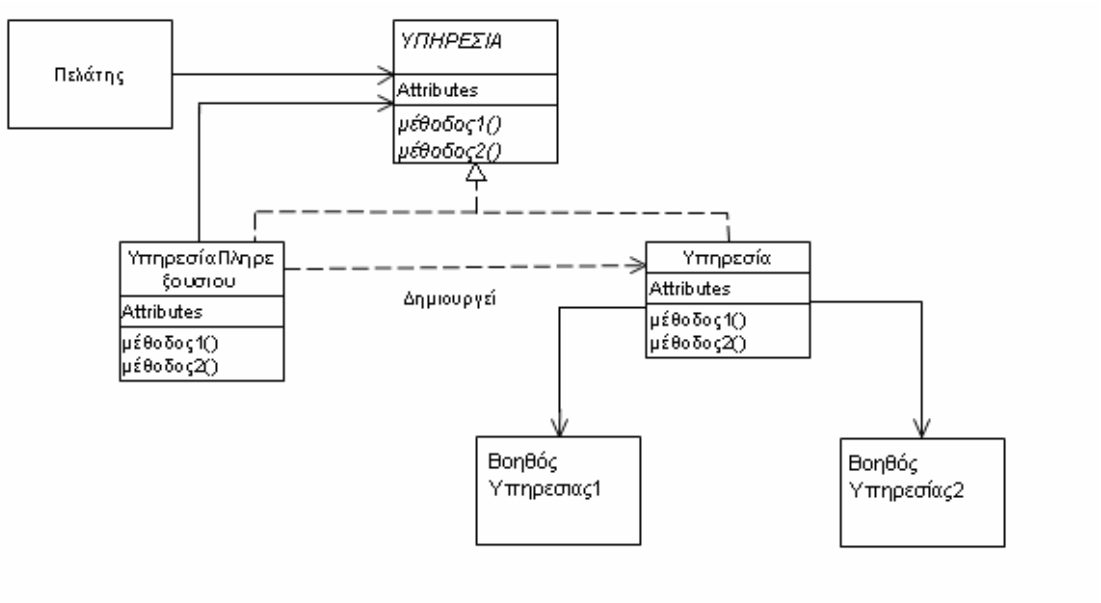
Οι συνέπειες του προτύπου διακοσμητή είναι ότι δημιουργός-συντάκτης του θέματος δεν πρέπει να κάνει τίποτα επιπλέον για το αντικείμενο που διακοσμείται. Ομοίως, οι διακοσμητές δεν προετοιμάζονται για τη διακόσμηση. Επίσης, είναι εύκολο να προστεθεί οποιοσδήποτε συνδυασμός ευθυνών. Η ίδια ευθύνη μπορεί ακόμη και να προστεθεί δύο φορές. Αυτό είναι δύσκολο χρησιμοποιώντας κληρονομικότητα.

Το ίδιο αντικείμενο μπορεί να διακοσμηθεί ταυτόχρονα με διαφορετικούς τρόπους. Οι πελάτες μπορούν να επιλέξουν ποιες ιδιότητες θέλουν να έχει, με την αποστολή των μηνυμάτων στον αρμόδιο διακοσμητή. Τα αντικείμενα δεν χρεώνονται κάτι επιπλέον, για μια ευθύνη που δεν χρειάστηκε να χρησιμοποιήσουν. Κατά συνέπεια υπάρχει αποδοτικότητα.

Μερικά πρότυπα που είναι παρόμοια με το πρότυπο διακοσμητή είναι το πρότυπο προσαρμοστή –μετασχηματιστή (Adapter Pattern), το πρότυπο στρατηγικής (Strategy Pattern) και το πρότυπο σύνθεσης (Composite Pattern).
Πρότυπο προσαρμοστή-μετασχηματιστή: Ένας διακοσμητής είναι διαφορετικός από έναν προσαρμοστή, αφού ένας διακοσμητής αλλάζει μόνο το τις ευθύνες αντικειμένου , όχι της διεπαφής. Επίσης, ένας μετασχηματιστής θα δώσει σε ένα αντικείμενο μια απολύτως νέα διεπαφή.
Πρότυπο σύνθεσης/συνάθροισης: Ένας διακοσμητής μπορεί να αντιμετωπισθεί ως συνάθροιση με μόνο ένα συστατικό. Όμως ένας διακοσμητής προσθέτει επιπλέον ευθύνες, το οποίο δεν γίνεται στη συνάθροιση αντικειμένου.
Πρότυπο στρατηγικής: Ένας διακοσμητής επιτρέπει να αλλαχτεί μόνο η εμφάνιση ενός αντικειμένου. Μια στρατηγική επιτρέπει να αλλάξει το εσωτερικό του αντικειμένου και όχι η εμφάνιση του.

4.7 Πρότυπο Εικονικού Πληρεξουσίου (Virtual Proxy pattern)

Εάν είναι δύσκολο ή ακριβό να δημιουργηθεί ένα στιγμιότυπο ενός αντικειμένου και επιπλέον δεν είναι άμεσα αναγκαίο, τότε είναι συμφέρον να αναβληθεί η δημιουργία του, έως ότου είναι σαφές ότι απαιτείται [42]. Με τον όρο ακριβός εννοείται ότι το αντικείμενο για να λειτουργήσει πρέπει να χρησιμοποιήσει πολλούς από τους πόρους του συστήματος [40]. Το πρότυπο εικονικού πληρεξούσιου κρύβει από τους πελάτες του, το γεγονός ότι ένα αντικείμενο μπορεί να μην υπάρχει ακόμα, έχοντας πρόσβαση στο αντικείμενο έμμεσα μέσω ενός αντικειμένου πληρεξούσιου (proxy object). Το αντικείμενο πληρεξούσιου εφαρμόζει την ίδια διεπαφή με το αντικείμενο που μπορεί να μην υπάρχει. Γίνεται προσπάθεια να αποφευχθεί η δημιουργία του αντικειμένου για όσο το δυνατό πιο μεγάλο χρονικό διάστημα [41]. Η τεχνική καθυστέρησης της δημιουργίας ενός αντικειμένου έως ότου απαιτείται πραγματικά καλείται μερικές φορές «lazy instantiation».
Ένα παράδειγμα είναι ένας επεξεργαστής κειμένου, όπως ο Microsoft Word, ο οποίος παρέχει την δυνατότητα εκτύπωσης. Κάθε φορά που ανοίγει η εφαρμογή του Word δεν είναι επιθυμητή η εκτύπωση του κειμένου. Οπότε κάθε φορά που ανοίγει η εφαρμογή Word δεν χρειάζεται να φορτώνεται και η λειτουργία εκτύπωσης. Η λειτουργία εκτύπωσης θα φορτώνεται μόνο όταν πραγματικά χρειάζεται εκτύπωση του κειμένου.
Στο σχήμα 4.7.1 απεικονίζεται το διάγραμμα κλάσεων του προτύπου εικονικού πληρεξουσίου.



Σχήμα 4.7.1 Απεικόνιση του διαγράμματος κλάσεων του προτύπου εικονικού πληρεξουσίου.

Εδώ εξηγούνται οι ρόλοι που διαδραματίζονται από τη διεπαφή και τις κλάσεις του προτύπου εικονικού πληρεξουσίου.

1). Υπηρεσία (Service).
Μια τέτοια κλάση παρέχει υψηλού επιπέδου λογική για μια υπηρεσία που παρέχει. Όταν πρέπει να δημιουργηθεί ένα στιγμιότυπο από αυτήν την κλάση, αυτή δημιουργεί τα άλλα αντικείμενα που χρειάζεται. Αυτές οι κλάσεις φαίνονται στο διάγραμμα ως ΒοηθόςΥπηρεσία1, ΒοηθόςΥπηρεσία2, κτλ.

2). Πελάτης.
Μια κλάση εδώ χρησιμοποιεί την υπηρεσία που παρέχεται από την κλάση Υπηρεσία. Οι κλάσεις πελατών ποτέ δεν χρησιμοποιούν άμεσα μια κλάση υπηρεσίας. Χρησιμοποιούν όμως μια κλάση ΥπηρεσίαΠληρεξουσίου, που παρέχει τη λειτουργία της κλάσης Υπηρεσία. Χρησιμοποιώντας μια κλάση Υπηρεσίας διατηρούνται οι κλάσεις πελατών ανεπηρέαστες, εάν το στιγμιότυπο της κλάσης Υπηρεσία που τα αντικείμενα πελατών χρησιμοποιούν έμμεσα, υπάρχει ήδη.

3). ΥπηρεσίαΠληρεξούσιου.
Ο σκοπός της κλάσης ΥπηρεσίαΠληρεξούσιου είναι να καθυστερήσει τη δημιουργία των αντικειμένων της κλάσης Υπηρεσίας έως ότου απαιτούνται πραγματικά. Μια κλάση ΥπηρεσίαΠληρεξουσίου παρέχει εμμεσότητα μεταξύ των κλάσεων πελατών και μιας κλάσης Υπηρεσία. Η εμμεσότητα αυτή κρύβει από τα αντικείμενα πελάτη, το γεγονός ότι όταν δημιουργείται ένα αντικείμενο ΥπηρεσίαΠληρεξουσίου, το αντίστοιχο αντικείμενο Υπηρεσίας δεν υπάρχει και η κλάση Υπηρεσία μπορεί ακόμη και να μην είχε φορτωθεί. Ένα αντικείμενο ΥπηρεσίαΠληρεξουσίου είναι αρμόδιο για τη δημιουργία του αντίστοιχου αντικειμένου Υπηρεσίας. Ένα αντικείμενο ΥπηρεσίαΠληρεξουσίου δημιουργεί το αντίστοιχο αντικείμενο Υπηρεσίας την πρώτη φορά που καλείται να εκτελέσει μια λειτουργία που απαιτεί την ύπαρξη του αντικειμένου Υπηρεσίας.

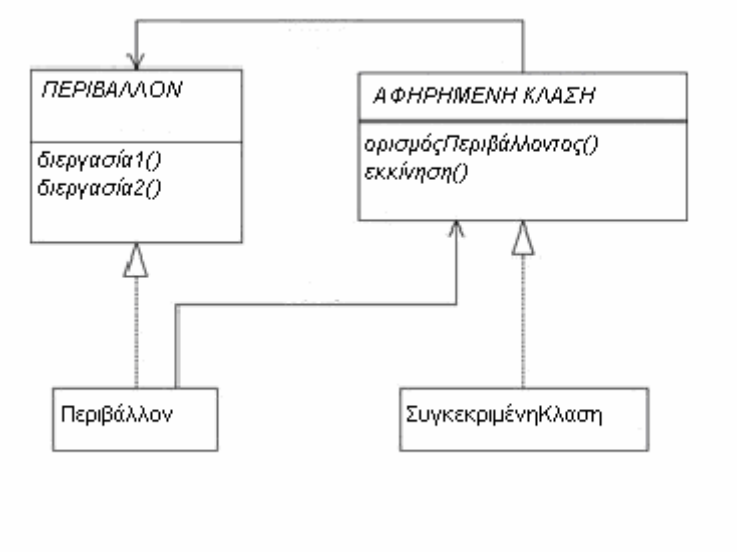
Μια κλάση ΥπηρεσίαΠληρεξουσίου κωδικοποιείται ειδικά για να λάβει την πρόσβαση στην κλάση Υπηρεσίας μέσω μιας δυναμικής αναφοράς. Συνήθως, κλάσεις έχουν αναφορά σε άλλες κλάσεις μέσω στατικών αναφορών. Μια στατική αναφορά αποτελείται απλά από το όνομα μιας κλάσης που εμφανίζεται σε μια κατάλληλη θέση στον πηγαίο κώδικα. Όταν ένας μεταγλωττιστής βλέπει εκείνο το είδος αναφοράς, παράγει ένα αποτέλεσμα που αναγκάζει την άλλη κλάση να φορτωθεί αυτόματα μαζί με την κλάση που περιέχει την αναφορά.
Το πρότυπο εικονικού πληρεξούσιου αποτρέπει τη φόρτωση της κλάσης Υπηρεσία και των σχετικών κλάσεων μαζί με το υπόλοιπο του προγράμματος, διαβεβαιώνοντας ότι το υπόλοιπο του προγράμματος δεν περιέχει στατικές αναφορές στην κλάση Υπηρεσία. Αντ' αυτού, το υπόλοιπο του προγράμματος αναφέρεται στην κλάση Υπηρεσίας μέσω της κλάσης ΥπηρεσίαΠληρεξουσίου και η κλάση ΥπηρεσίαΠληρεξουσίου αναφέρεται στην κλάση Υπηρεσία μέσω μιας δυναμικής αναφοράς.
Μια δυναμική αναφορά αποτελείται από μια κλήση μεθόδου που περνά ένα string που περιέχει το όνομα μιας κλάσης, σε μια μέθοδο που φορτώνει την κλάση εάν δεν έχει φορτωθεί και επιστρέφει μια αναφορά στην κλάση. Επειδή το όνομα της κλάσης εμφανίζεται μόνο μέσα σε ένα string, οι μεταγλωττιστές δεν γνωρίζουν ότι η κλάση δεν θα υλοποιηθεί και έτσι δεν παράγουν οποιαδήποτε αποτέλεσμα που προκαλεί την κλάση να φορτωθεί.

4). Υπηρεσία.
Μια κλάση ΥπηρεσίαΠληρεξουσίου δημιουργεί ένα στιγμιότυπο της κλάσης Υπηρεσίας, μέσω κλήσεων μεθόδων που δεν απαιτούν στατικές αναφορές στην κλάση Υπηρεσίας. Μια κλάση ΥπηρεσίαΠληρεξουσίου καλεί επίσης τις μεθόδους της κλάσης Υπηρεσία, χωρίς να έχει στατικές αναφορές στην κλάση Υπηρεσία. Καλεί τις μεθόδους της κλάσης Υπηρεσίας εκμεταλλεύοντας το γεγονός ότι η κλάση Υπηρεσία εφαρμόζει τη διεπαφή Υπηρεσία.
Η διεπαφή Υπηρεσία είναι μια διεπαφή που δηλώνει ότι όλες οι μέθοδοι που εφαρμόζει η κλάση Υπηρεσία και απαιτούνται από την κλάση ΥπηρεσίαΠληρεξουσίου. Ένα αντικείμενο ΥπηρεσίαΠληρεξουσίου μεταχειρίζεται την αναφορά στο αντικείμενο Υπηρεσίας, που δημιουργεί ως αναφορά σε ένα αντικείμενο Υπηρεσία. Η κλάση Υπηρεσία χρησιμοποιεί τις στατικές αναφορές στη διεπαφή Υπηρεσία στις μεθόδους κλήσης αντικειμένων Υπηρεσίας. Καμία στατική αναφορά δεν απαιτείται στην κλάση υπηρεσιών.

4.8 Πρότυπο Δυναμικού Συνδέσμου (Dynamic Linkage pattern)

Επιτρέπει σε ένα πρόγραμμα, αφού του ζητηθεί, να φορτωθούν και να χρησιμοποιηθούν αυθαίρετες κλάσεις που εφαρμόζουν μια γνωστή διεπαφή. Ένα πρόγραμμα πρέπει να είναι σε θέση να φορτώσει και να χρησιμοποιήσει αυθαίρετες κλάσεις, τις οποίες δεν γνωρίζει από πριν. Τα στιγμιότυπα μιας κλάσης που έχει φορτωθεί, πρέπει να είναι σε θέση να καλεστούν πίσω στο πρόγραμμα φόρτωσε την κλάση.

Το σχήμα 4.8.1 είναι ένα διάγραμμα κλάσεων που παρουσιάζει τους ρόλους των διεπαφών και των κλάσεων που συμμετέχουν στο πρότυπο δυναμικού συνδέσμου [9].



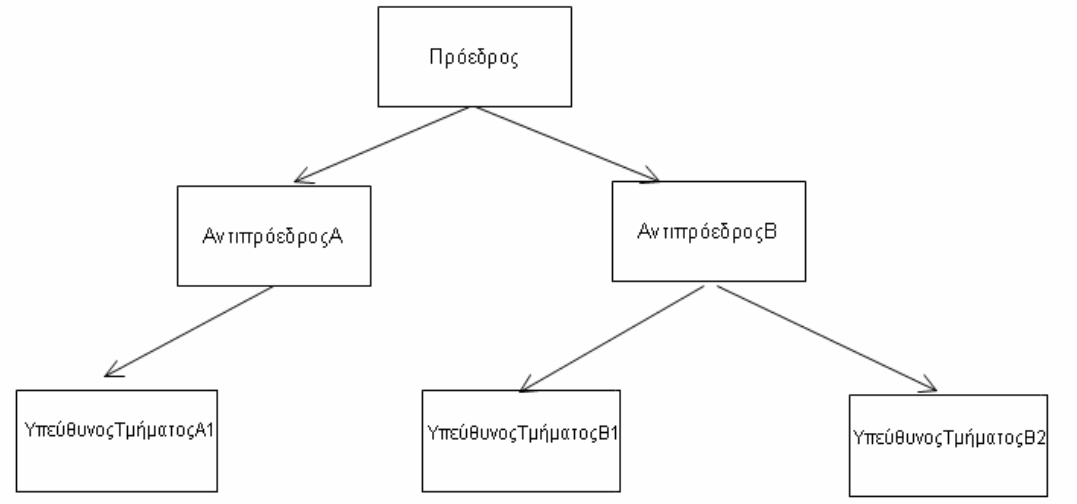
Εδώ είναι οι περιγραφές των κλάσεων και διεπαφών στο πρότυπο δυναμικού συνδέσμου, όπως φαίνονται στο σχήμα 4.8.1.

1). Περιβάλλον
Μια διεπαφή σε αυτόν τον ρόλο δηλώνει τις μεθόδους περιβάλλοντος που μπορεί να καλέσει μια φορτωμένη κλάση.
2). ΠεριβάλλονΑ.
Μια κλάση σε αυτόν τον ρόλο είναι μέρος του περιβάλλοντος που φορτώνει μια κλάση ΣυγκεκριμένηΚλάση. Εφαρμόζει τη διεπαφή Περιβάλλον. Μια αναφορά σε ένα στιγμιότυπο αυτής της κλάσης περνά (περνιέται) στα στιγμιότυπα της κλάσης ΣυγκεκριμένηΚλάση, έτσι ώστε να μπορούν να καλέσουν τις μεθόδους του αντικειμένου ΠεριβάλλονΑ, που δηλώνεται από τη διεπαφή Περιβάλλον.
3). ΑφηρημένηΚλάση.
Μια κλάση σε αυτόν τον ρόλο δηλώνει έναν αριθμό άλλων μεθόδων, συνήθως αφηρημένων, εκτός από τις δυο που παρουσιάζονται εδώ [9].
Πρέπει να υπάρξει μια μέθοδος με όνομα όπως η ορισμόςΠεριβάλλοντος, η οποία επιτρέπει σε στιγμιότυπα των υποκλάσεων των κλάσεων ΑφηρημένηΚλάση να περαστούν με αναφορά σε ένα στιγμιότυπο μιας κλάσης που εφαρμόζει τη διεπαφή Περιβάλλον. Ο σκοπός αυτής της μεθόδου είναι να μπορούν τα αντικείμενα ΑφηρημένηΚλάση να καλέσουν τις μεθόδους ενός αντικειμένου περιβάλλοντος.
Το περιβάλλον καλεί μια άλλη μέθοδο, την εκκίνηση, για να πει σε ένα στιγμιότυπο μιας φορτωμένης κλάσης να ξεκινήσει να κάνει οτιδήποτε πρέπει να κάνει.
4). ΣυγκεκριμένηΚλάση.
Οι κλάσεις σε αυτόν τον ρόλο είναι υποκλάσεις της ΑφηρημένηςΚλάσης που μπορούν να φορτωθούν δυναμικά.

4.9 Πρότυπο Σύνθεσης (Composite Pattern)

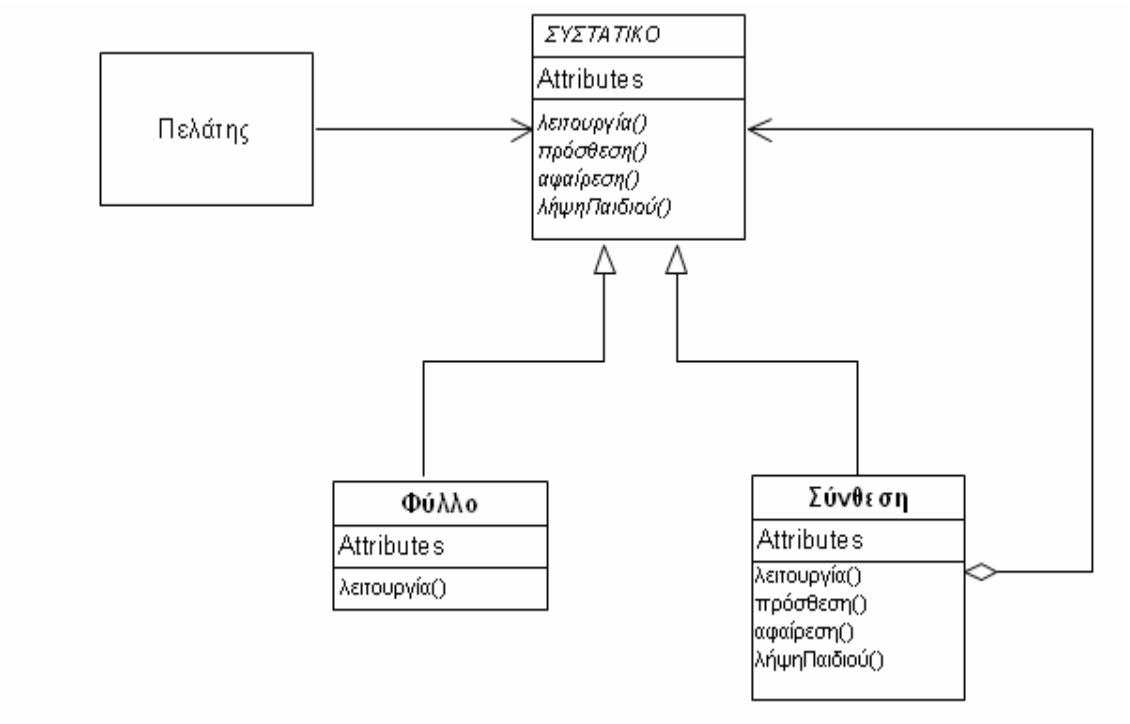
Ένα σχεδιαστικό πρότυπο σύνθεσης καταρχήν είναι ένα σχεδιαστικό πρότυπο. Είναι η αφαίρεση από μια συγκεκριμένη επαναλαμβανόμενη λύση, που λύνει ένα πρόβλημα σε ένα ορισμένο πλαίσιο [18] [26] [27]. Ένα πρότυπο είναι ένα σύνθετο πρότυπο, εάν μπορεί να εξηγηθεί καλύτερα ως σύνθεση των περαιτέρω ατομικών ή σύνθετων προτύπων. Ένα ατομικό πρότυπο είναι ένα πρότυπο που, σε ένα δεδομένο επίπεδο αφαίρεσης, δεν μπορεί να περιγραφεί λογικά ως σύνθεση άλλων προτύπων.

Σε ένα σύνθετο πρότυπο, τα πρότυπα που το αποτελούν ενσωματώνονται το ένα μέσα στο άλλο, για να επιτύχουν μια σύμπραξη, που δίνει στο σύνθετο πρότυπο την ταυτότητά του, πέρα από την ύπαρξη απλώς ατομικών προτύπων. Αυτό διακρίνει ένα σύνθετο πρότυπο από μια αυθαίρετη σύνθεση προτύπων, η οποία μπορεί να είναι μια κατάλληλη λύση για ένα συγκεκριμένο σχεδιαστικό πρόβλημα, αλλά δεν επαναλαμβάνεται ως πρότυπο.
Το χρησιμοποιούμε όταν πρέπει να αναπαρασταθούν ιεραρχίες κλάσεων (δένδρα) και να συντεθούν αυτές οι κλάσεις για μια πιο σύνθετη εικόνα (κληρονομικότητας). Ακόμη χρησιμοποιείται όταν πρέπει να χρησιμοποιηθούν μέθοδοι με τις ίδιες ονομασίες και στα συνιστώντα μέλη και στη σύνθετη κλάση, δηλαδή και στα κομμάτια ξεχωριστά και σε ολόκληρο το σύνθετο κομμάτι.
Ένα δένδρο συχνά χρησιμοποιείται για να αναπαραστήσει μια ιεραρχία. Αυτό γίνεται επειδή οι σχέσεις μεταξύ των τμημάτων στην ιεραρχία μπορούν να απεικονιστούν με τα κλαδιά στο δένδρο. Μια ιεραρχία μπορεί πολύ απλά να αναπαρασταθεί χρησιμοποιώντας ένα δένδρο, όπως φαίνεται στο επόμενο σχήμα.[16].



Σχήμα 4.9.1. Απεικονίζεται μια ιεραρχία χρησιμοποιώντας ένα δένδρο.

Το δένδρο φυσικά και είναι ανάποδα, γιατί αυτός είναι ο συνηθισμένος τρόπος που αναπαρίσταται ένα δένδρο. Η ρίζα του δένδρου είναι ο Πρόεδρος και τα φύλλα είναι οι ΥπεύθυνοςΤμημάτοςΑ1, ΥπεύθυνοςΤμημάτοςΒ1, ΥπεύθυνοςΤμημάτοςΒ2. Οι ΑντιπρόεδροςΑ και ΑντιπρόεδροςΒ είναι παιδιά του Προέδρου.
Το πρότυπο αυτό επιτρέπει να συντεθούν αντικείμενα σε δομές δέντρων για να αναπαρασταθούν οι ιεραρχίες με τα διαφορετικά μέρη (whole part hierarchies). Η σύνθεση αφήνει τους πελάτες να μεταχειριστούν τα μεμονωμένα αντικείμενα και τη σύνθεση των αντικειμένων.
Ένα απλό παράδειγμα είναι η ομαδοποίηση στο Word. Το τελικό στοιχείο που είναι η ένωση όλων των επιμέρους στοιχείων σε ένα, έχει τις ίδιες ιδιότητες που έχει κάθε ξεχωριστό στοιχείο. Στο σχήμα 4.9.2 απεικονίζεται το διάγραμμα κλάσεων του προτύπου σύνθεσης.



Σχήμα 4.9.2 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου σύνθεσης.

Εδώ είναι οι περιγραφές της διεπαφής και των κλάσεων που συμμετέχουν στο πρότυπο σύνθεσης:

1). Συστατικό.
Μια διεπαφή σε αυτόν τον ρόλο εφαρμόζεται από όλα τα αντικείμενα στην ιεραρχία των αντικειμένων που αποτελούν ένα σύνθετο αντικείμενο. Εφαρμόζεται και από τη κλάση Σύνθεση και από την κλάση Φύλλο. Τα αντικείμενα σύνθεσης μεταχειρίζονται τα αντικείμενα που περιέχουν ως στιγμιότυπα κλάσεων που εφαρμόζουν τη διεπαφή Συστατικό, παρά ως στιγμιότυπα της πραγματικής κλάσης τους [9].
2). Φύλλο.
Αυτή η κλάση καθορίζει τη συμπεριφορά για τα στοιχεία στη σύνθεση. Αυτό επιτυγχάνεται εφαρμόζοντας τις μεθόδους που υποστηρίζει η Σύνθεση.
3). Σύνθεση.
Μια κλάση σε αυτόν τον ρόλο είναι η αφηρημένη υπέρ-κλάση όλων των σύνθετων αντικειμένων που συμμετέχουν στο πρότυπο σύνθεσης. Η Σύνθεση καθορίζει και παρέχει τις εφαρμογές των μεθόδων για τα τμήματα-συστατικά ενός αντικειμένου σύνθεσης. Η μέθοδος Πρόσθεση προσθέτει ένα συστατικό σε ένα αντικείμενο σύνθεσης. Η μέθοδος Αφαίρεση αφαιρεί ένα συστατικό από ένα αντικείμενο σύνθεσης. Η μέθοδος ΛήψηΠαιδιού, επιστρέφει μια αναφορά σε ένα συστατικό αντικείμενο ενός αντικειμένου σύνθεσης.
4). Πελάτης.
Ο Πελάτης χρησιμοποιεί την διεπαφή Συστατικό, για να χειριστεί τα αντικείμενα στην σύνθεση.

Μια συνέπεια του σύνθετου προτύπου είναι ότι επιτρέπει να καθοριστεί μια κλάση ιεραρχίας από απλά αντικείμενα και σύνθετα αντικείμενα σύνθεσης, έτσι ώστε να

εμφανίζονται ίδια στο πρόγραμμα του πελάτη. Λόγω αυτής της απλότητας, ο πελάτης μπορεί να είναι πολύ απλός, δεδομένου ότι οι κόμβοι και τα φύλλα αντιμετωπίζονται με τον ίδιο τρόπο. Το πρότυπο αυτό διευκολύνει επίσης, να προστεθούν τα νέα είδη συστατικών εύκολα, εφ' όσον υποστηρίζουν μια παρόμοια διεπαφή προγραμματισμού. Όμως, γίνεται δυσκολότερη η τοποθέτηση περιορισμών σε μια σύνθεση, στον τύπο των συστατικών [36].

Πολύ χρήσιμο είναι το έγγραφο με τίτλο ‘‘Composite Design Patterns’’ [28]. Αυτό το έγγραφο καθορίζει την έννοια του σύνθετου προτύπου και την επεξηγεί χρησιμοποιώντας μικρά παραδείγματα. Δείχνει πώς τα σύνθετα πρότυπα μπορούν να γίνουν κατανοητά και παρουσιάζεται μια ανάλυση και μια τεχνική παραγωγής, που βοηθούν τους συντάκτες προτύπων να επιλύσουν την ουσία των σύνθετων προτύπων. Στο έγγραφο με τίτλο ‘‘ Permissions to Specify the Composite Design Pattern ’’ [29] παρουσιάζεται μια εξειδίκευση του σύνθετου προτύπου, επιτρέποντας στον πελάτη να προσθέσει παιδιά σε οποιοδήποτε κόμβο σε ένα σύνθετο δέντρο οποιαδήποτε στιγμή.

Αναφορές :

[1]: Απόστολος Ζάρρας, *Πρότυπα Δόμησης (Structural Patterns),* Online Upgrade of Object-Oriented Middleware http://www.cs.uoi.gr/~zarras/
[2]: W.Liu (2003), Slides, *CSC407 T8: Structural Pattern 2*, www.cs.toronto.edu
[3]: Woei-Kae Chen , (1998), Slides, *Object-Oriented Programming Adapter Pattern*, CSIE Department, NTUT.
[4]: Jonathan S. Ostroff, (2004), Slides, *COSC3311 – Adapter pattern and Report2*, Department of Computer Science, York University 2004.
[5]: Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M,(1996), *Pattern-Oriented Software Architecture - A System of Patterns*, John Wiley and Sons Ltd, Chichester, UK.
[6]: Ross Harmes and Dustin Diaz, (2008), Pro JavaScript Design Patterns Publisher:Apress 2008.
[7]: Alex Martelli, (2004), *Masquerading and Adaptation Design Patterns*, AB Strakt EuroPython 2004 .
[8]: Kai Qian, Larry L. Wang, *Adapter Pattern in Component and Service Levels vs.Class and Object Levels*, (2006), Software Engineering Research and Practice 2006, pp.226-232.
[9]: Mark Grand, (2002), *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML, Second Edition* John Wiley & Sons © 2002.
[10]: Bob Tarr, (2000), *The Adapter Pattern Design Patterns In Java*, http://userpages.umbc.edu/~tarr/dp/fall00/cs491.html
[11]: JAMES W. COOPER, (1998), *The Design Patterns, Java Companion*, Addison-Wesley Design Patterns Series 1998.
[12]: http://www.dofactory.com/Patterns/PatternFacade.aspx
[13]: Douglas C. Schmidt, (1999), *Wrapper Façade A Structural Pattern for Encapsulating Functions within Classes*, This paper appeared in the C++ Report magazine, February, 1999.
[14]: Daqing Hou, (2007), Slides *Design patterns Lecture 9 EE 564* , Winter 2007.
[15]: William Sanders, Chandima Cumaranatunge, (2007), *ActionScript 3.0 Design Patterns: Object Oriented Programming Techniques*,O’Reily.
[16]: Bruno R. Preiss B.A.Sc., M.A.Sc., Ph.D., P.Eng., (1999), *Data Structures and Algorithms with Object-Oriented Design Patterns in Java* , John Wiley and Sons.
[17]: http://www.netobjectives.com/files/design-patterns-explained-ch10.pdf
[18]: GOF95 Gamma, E., Helm, R., Johnson, R., Vlissides, J., (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Boston: Addison-Wesley, 1995
[19]: Alan Shalloway and James R. Trott, (2004), *Design Patterns Explained: A New Perspective on Object-Oriented Design 2nd Ed.*, Software Patterns Series, Addison-Wesley Professional 2005, pp. 480.
[20]: HansWerner and Pohl Jens Gerlach,(2006), *Using the Bridge Design Pattern for OSGi Service Update*, Computer Architecture and Software Technology (FIRST) Berlin Germany.
[21]: Brendan McCarthy, (1998), *The Cascading Bridge Design Pattern* ,Sun Java Center PLoP-98 Conference.

[22]: Douglas Lyon, (2004), *JOURNAL OF OBJECT TECHNOLOGY*, Published by ETH Zurich 2004. Project Imperion: New Semantics, Façade and Command Design Patterns for Swing.
[23]: Kenneth M. Anderson, (2007), Slides *Adapter*, University of Colorado, Boulder, CSCI 4448/6448 — Lecture 23 — 11/13/2007.
[24]: Brian d foy, *The Facade Design Pattern* , The Perl Review 2002.
[25]: Bob Tarr, (2000), *The Facade Pattern Design Patterns In Java*, http://userpages.umbc.edu/~tarr/dp/fall00/cs491.html
[26]: Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad and Michael Stal, (1996), *Pattern-Oriented Software Architecture*. POSA96, Wiley & Sons, 1996.
[27]: Dirk Riehle and Heinz Züllighoven, (1996), *Understanding and Using Patterns in Software Development*, Theory and Practice of Object Systems 2, 1 RZ96, pp.3-13.
[28]: Dirk Riehle, (1997), *Composite Design Patterns*, Published in Proceedings of the 1997 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '97). ACM Press, 1997, pp. 218-228.
[29]: Kevin Bierhoff and Jonathan Aldrich, (2008), *Permissions to Specify the Composite Design Pattern*, 7th InternationalWorkshop on Specification and Verification of ComponentBased Systems (SAVCBS 2008), November 9–10, 2008, Atlanta, GA, USA.
[30]: Cay Horstman, (2006), Object-Oriented Design and Patterns 2nd Ed., 2006 John Wiley and Sons.
[31]: Erich Gama, Richard Helm, Ralph Johnson, John Vlissides, (1995), *Design patterns: elements of reusable object-oriented software*, Addison-Wesley Professional Computing Series 1995, pp. 395.
[32]: http://en.wikipedia.org/wiki/Flyweight_pattern
[33]: Nigamanth Sridhar, (2008), *Decorator Pattern* ,Software Engineering, Cleveland State University Electrical and Computer Engineering, Fall 2008 EEC 521.
[34]: http://www.codeproject.com/KB/architecture/patterns_in_real_life.aspx
[35]: Bob Tarr, (2000), *The Decorator Pattern Design Patterns In Java*, http://userpages.umbc.edu/~tarr/dp/fall00/cs491.html
[36]: Mel Ó Cinnéide, (2005), Slides: *The Composite Pattern*, Department of Computer Science University College Dublin, Apr-05, Software Engineering MSc: Design Patterns.
[37]: William Sanders and Chandima Cumaranatunge, (2007), *Object-Oriented Programming Techniques ActionScript 3.0 Design Patterns*, 2007 Published by O'Reilly Media, Inc.,
[38]: http://exciton.cs.rice.edu/JavaResources/DesignPatterns/DecoratorPattern.htm
[39]: http://en.wikipedia.org/wiki/Runtime
[40]: http://www.javabeat.net/tips/26-working-with-virtual-proxy-pattern.html
[41]: Kenneth M. Anderson , (2005), Slides *Lecture 21: Design Patterns (Part 3),* Object-Oriented Analysis and Design CSCI 6448 - Spring Semester, 2005.
[42]: Rene de Jong , *Slides Lesson 6: Patterns: Proxy and Chain of Responsibility, Engaging the Managing Intelligence of Nature Advanced Software Development*, Computer Science Department MAHARISHI INTERNATIONAL UNIVERSITY of MANAGEMENT.

5. ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ ΣΥΜΠΕΡΙΦΟΡΑΣ
Behavioral Design Patterns

5.1 Σύντομη αναφορά των σχεδιαστικών προτύπων συμπεριφοράς.

Στην τεχνολογία λογισμικού, τα σχεδιαστικά πρότυπα συμπεριφοράς είναι σχεδιαστικά πρότυπα που προσδιορίζουν κοινή επικοινωνία προτύπων μεταξύ των

αντικειμένων και πραγματοποιούν αυτά τα πρότυπα. Με αυτόν τον τρόπο, αυτά τα πρότυπα αυξάνουν την ευελιξία στην πραγματοποίηση αυτής της επικοινωνίας [1].
Τα πρότυπα συμπεριφοράς είναι εκείνα που ενδιαφέρονται για τις αλληλεπιδράσεις μεταξύ των αντικειμένων. Οι αλληλεπιδράσεις μεταξύ των αντικειμένων πρέπει να είναι τέτοιες που να επικοινωνούν το ένα με το άλλο ακόμα και όταν συνδέονται αόριστα-χαλαρά. Η χαλαρή σύζευξη είναι το κλειδί στις σε αυτήν την αρχιτεκτονική. Η εφαρμογή και ο πελάτης πρέπει να συνδεθούν αόριστα-χαλαρά, προκειμένου να αποφύγουν δύσκολη δημιουργία κώδικα και εξαρτήσεις [2].
Τα πρότυπα συμπεριφοράς αφορούν τον καταμερισμό αρμοδιοτήτων σε διάφορες κλάσεις και τον ορισμό του τρόπου επικοινωνίας μεταξύ των αντικειμένων τους κατά τον χρόνο εκτέλεσης. Σε αντίθεση με τα δομικά πρότυπα, τα πρότυπα συμπεριφοράς βρίσκουν εφαρμογή στον αρχικό σχεδιασμό μίας ιεραρχίας κλάσεων και όχι στην εκ των υστέρων επέκταση κάποιας υπάρχουσας ιεραρχίας. Ο γενικός κανόνας είναι ότι έχουμε να κάνουμε με μία ομάδα κλάσεων οι οποίες υλοποιούν με διαφορετικούς τρόπους μία κοινή διασύνδεση Α. Ως συνήθως το εξωτερικό πρόγραμμα είναι προτιμότερο να χειρίζεται μόνον αναφορές του αφηρημένου τύπου Α και να βασίζεται στον πολυμορφισμό ώστε να μην παραβιάζεται η αρχή ανοιχτότητας-κλειστότητας. Η επικοινωνία μεταξύ αντικειμένων γίνεται με παρόμοιο τρόπο, καθώς όχι σπάνια μία κλήση μεθόδου δέχεται ως όρισμα μία αναφορά αφηρημένου τύπου κι έτσι το αντίστοιχο αντικείμενο μπορεί να προσπελαύνει με τυποποιημένο τρόπο δημόσιες μεθόδους στιγμιότυπων κάθε κλάσης της επίμαχης ιεραρχίας [3].

Τα παραδείγματα αυτού του τύπου σχεδιαστικών προτύπων περιλαμβάνουν:

Αλυσίδα του προτύπου ευθύνης (Chain of responsibility pattern) : Τα αντικείμενα εντολής (Command Objects) αντιμετωπίζονται ή μεταφέρονται προς άλλα αντικείμενα, όπως γίνεται συνήθως με την επεξεργασία αντικειμένων. Επιτρέπει μια ομαλή περαιτέρω αποσύζευξη μεταξύ των κλάσεων, περνώντας μια αίτηση μεταξύ των κλάσεων έως ότου αναγνωριστεί.

Πρότυπο εντολής (Command Pattern): Τα αντικείμενα του προτύπου εντολής ενθυλακώνουν μια δράση και τις παραμέτρους της. Παρέχει έναν απλό τρόπο να διαχωριστεί η εκτέλεση μιας εντολής από τη διεπαφή της.

Πρότυπο διερμηνέα (Interpreter Pattern): Εφαρμόζεται μια εξειδικευμένη γλώσσα υπολογιστών για να λυθεί γρήγορα ένα συγκεκριμένο σύνολο προβλημάτων. Παρέχει έναν ορισμό για το πώς να συμπεριληφθούν στοιχεία γλώσσας σε ένα πρόγραμμα.

Πρότυπο Iterator (Iterator Pattern): Τα Iterators χρησιμοποιούνται για να υπάρχει πρόσβαση στα στοιχεία ενός συνολικού- αθροιστικού αντικειμένου, διαδοχικά, χωρίς να γίνεται έκθεση της βαθύτερης αναπαράστασής του. Δίνει λύση στο πρόβλημα επεξεργασίας των στοιχείων ενός συνόλου/συνάθροισης σε μια εφαρμογή.

Πρότυπο μεσολαβητή (Mediator Pattern): Παρέχει μια ενοποιημένη διεπαφή σε ένα σύνολο διεπαφών σε ένα υποσύστημα. Καθορίζει επίσης, πως μπορεί η επικοινωνία μεταξύ των κλάσεων να απλοποιηθεί, χρησιμοποιώντας μια άλλη κλάση, η οποία θα κρατά όλες τις κλάσεις, για να μη γνωρίζουν τίποτα η μια κλάση για την άλλη.

Πρότυπο ενθυμίου (Memento Pattern): Παρέχει τη δυνατότητα να αποκατασταθεί ένα αντικείμενο στην προηγούμενη κατάστασή του (μείωση των τιμών-rollback).

Πρότυπο μηδενικού αντικειμένου (Null Object Pattern): Είναι σχεδιασμένο για να ενεργεί ως προκαθορισμένη (default) τιμή ενός αντικειμένου.

Πρότυπο παρατηρητή (Observer Pattern): Τα αντικείμενα καταχωρούνται για να παρατηρούν ένα γεγονός, που μπορεί να αυξηθεί από ένα άλλο αντικείμενο. Ορίζει

ένα τρόπο με τον οποίο μπορεί ένας αριθμός από κλάσεις να ενημερωθεί για μια αλλαγή που έγινε.

Πρότυπο κατάστασης (State Pattern): Είναι ένας καθαρός τρόπος για ένα αντικείμενο, να αλλαχτεί μερικώς ο τύπος του στο χρόνο εκτέλεσης.

Πρότυπο στρατηγικής (Strategy Pattern): Ενθυλακώνει έναν αλγόριθμο σε μια κλάση.

Πρότυπο προδιαγραφών (Specification Pattern): Ανασυνδέει την επιχειρησιακή λογική (Business logic) με τρόπο Boolean.

Πρότυπο Template μεθόδου (Template method Pattern): Περιγράφει τον προγραμματισμό σκελετού (skeleton programming) ενός προγράμματος.

Πρότυπο επισκέπτη (Visitor Pattern): Είναι ένας τρόπος να χωριστεί ένας αλγόριθμος από ένα αντικείμενο. Επίσης, μπορεί να προσθέσει λειτουργία σε μια κλάση.

Πρότυπο ενιαίου εξυπηρετητή επισκέπτη (Single-serving visitor Pattern): Βελτιστοποιείται η εφαρμογή ενός επισκέπτη που διατίθεται, χρησιμοποιείται μόνο μια φορά, και μετά διαγράφεται.

Πρότυπο ιεραρχικού επισκέπτη (Hierarchical visitor Pattern): Παρέχει έναν τρόπο να επισκεφτεί κάθε κόμβος σε μια ιεραρχική δομή δεδομένων, όπως ένα δέντρο.

5.2 Πρότυπο Εντολής (Command Pattern)

Στον αντικειμενοστρεφή προγραμματισμό, το πρότυπο εντολής είναι ένα σχεδιαστικό πρότυπο, στο οποίο τα αντικείμενα χρησιμοποιούνται για να αντιπροσωπεύσουν τις ενέργειες. Ένα αντικείμενο εντολής ενθυλακώνει μια δράση και τις παραμέτρους του [4].

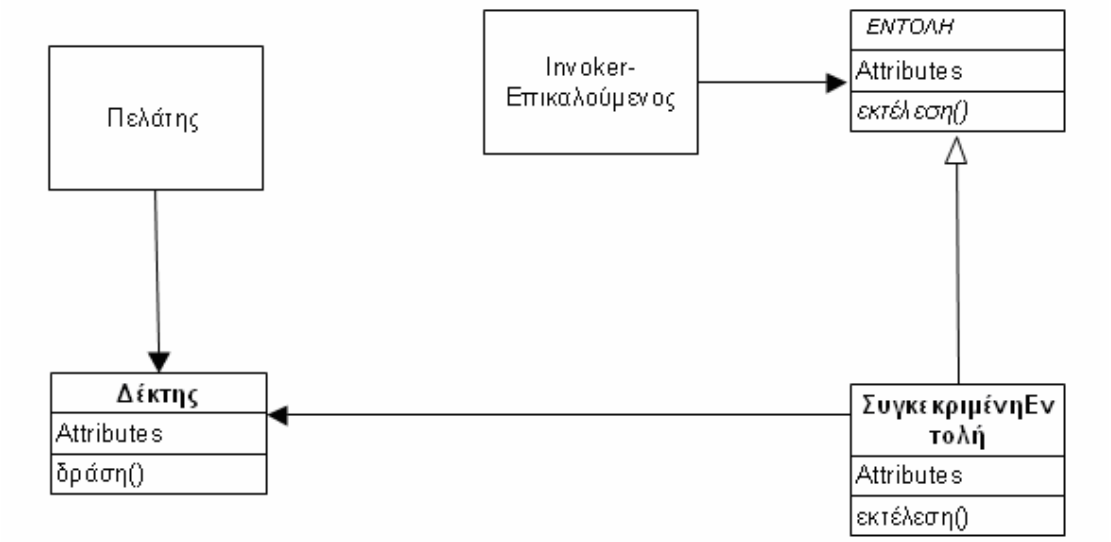
Η βασικότερη χρησιμότητα του προτύπου εντολής, είναι η ευελιξία και η επεκτασιμότητα, κατά τον καθορισμό της συμπεριφοράς στις εφαρμογές. Το πρότυπο εντολής ενθυλακώνει τη συμπεριφορά σε ένα φορητό αντικείμενο εντολής [11]. Επίσης, αποσυνδέει τις κλάσεις και όσες μεθόδους σε εκείνες τις κλάσεις εκτελούν απαιτούμενη συμπεριφορά από τη θέση όπου καλείται η συμπεριφορά. Ακόμη, το πρότυπο εντολής επιτρέπει σε έναν πελάτη να δημιουργήσει δυναμικά νέα συμπεριφορά, δημιουργώντας νέα αντικείμενα εντολής [5].

Αυτό το πρότυπο χρησιμοποιείται για να αναπαραστήσει ένα αίτημα και να εκτελέσει κάποιες λειτουργίες σε ένα αντικείμενο χωρίς δύσκολη κωδικοποίηση. Αντ' αυτού, ένα αντικείμενο εντολής αρχικοποιείται και διαμορφώνεται για να δείξει τη λειτουργία που εκτελεί και οποιεσδήποτε παραμέτρους έχει να περάσει. Αυτό επιτρέπει επίσης στα αντικείμενα εντολής να χειριστούν, να αποθηκευτούν, να τοποθετηθούν διαδοχικά στη σειρά - ουρά και σε μερικές περιπτώσεις να αναιρεθούν [6] [7] [8]. Για παράδειγμα, χρησιμοποιείται πολύ στην κρυπτογράφηση και αποκρυπτογράφηση, όπου τμήμα του αλγόριθμου αποκρυπτογράφησης στέλνεται μαζί με το κρυπτογραφημένο μήνυμα [12].

Ένας σημαντικός σκοπός του προτύπου εντολής είναι να κρατήσει εντελώς χωριστά το πρόγραμμα και τα αντικείμενα διεπαφής του χρήστη (το ενδιάμεσο με το χρήστη-user interface UI), από τις ενέργειες που κάνουν. Με άλλα λόγια, αυτά τα αντικείμενα προγράμματος πρέπει να είναι απολύτως χωριστά μεταξύ τους και δεν πρέπει να πρέπει να ξέρουν πώς λειτουργούν τα άλλα αντικείμενα. Το ενδιάμεσο με τον χρήστη λαμβάνει μια εντολή και λέει σε ένα αντικείμενο εντολής για να πραγματοποιήσει όποια καθήκοντα έχει καθοδηγηθεί για να κάνει. Το UI δεν ξέρει και δεν πρέπει να ξέρει ποιοι στόχοι θα εκτελεστούν.

Το αντικείμενο εντολής μπορεί επίσης να χρησιμοποιηθεί, όταν πρέπει το πρόγραμμα να εκτελέσει την εντολή, δηλαδή όταν οι πόροι είναι διαθέσιμοι, παρά αμέσως. Σε τέτοιες περιπτώσεις, οι εντολές μπαίνουν σε ουρά. Τέλος, μπορούν να χρησιμοποιηθούν τα αντικείμενα εντολής, για να μη ξεχνιούνται οι διαδικασίες, έτσι ώστε μπορούν να υποστηριχτούν αιτήματα αναίρεσης (undo) [8].

Το σχήμα πιο κάτω δείχνει ένα διάγραμμα κλάσεων ,το οποίο απεικονίζει τις κλάσεις, οι οποίες συμμετέχουν στο πρότυπο εντολής (Command Pattern) [8] [11] [12] [14].



Σχήμα 5.2.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου εντολής.

Εδώ είναι οι εξηγήσεις των ρόλων που διαδραματίζουν οι κλάσεις στο πρότυπο εντολής (Command Pattern) [11] [12].

1). Εντολή.
Μια κλάση σε αυτόν τον ρόλο είναι η υπέρ-κλάση άλλων κλάσεων που ενσωματώνουν εντολές. Εκτός από τη μέθοδο εκτέλεση (do) μπορεί να υπάρχει και η μέθοδος αναίρεση (undo). Η Εντολή καθορίζει ελάχιστα την αφηρημένη μέθοδο εκτέλεσης που άλλες κλάσεις καλούν, για να εκτελέσουν την εντολή που ενσωματώνεται από τις υποκλάσεις της. Εάν απαιτείται η αναίρεση, μια κλάση Εντολή καθορίζει επίσης μια μέθοδο αναίρεση που ανατρέπει τα αποτελέσματα της τελευταίας κλήσης στη μέθοδο εκτέλεση.

2). ΣυγκεκριμένηΕντολή.
Οι κλάσεις σε αυτόν τον ρόλο είναι συγκεκριμένες κλάσεις που περιέχουν μια συγκεκριμένη εντολή. Άλλες κλάσεις επικαλούνται την εντολή μέσω μιας κλήσης στη μέθοδο εκτέλεση της κλάσης. Η λογική αναίρεσης για την εντολή επικαλείται μέσω μιας κλήσης στη μέθοδο αναίρεση της κλάσης.

3).Invoker- Επικαλούμενος.
Μια κλάση σε αυτόν τον ρόλο δημιουργεί συγκεκριμένα αντικείμενα εντολής, εάν πρέπει να επικαλεστεί μια εντολή. Μπορεί να καλέσει τη μέθοδο εκτέλεση ενός αντικειμένου εντολής ή αυτή να το αφήσει και να το κάνει το αντικείμενο Πελάτη.

4). Δέκτης.

Ο Δέκτης περιέχει την λειτουργία που πρέπει να γίνει. Η μέθοδος δράση περιέχει τη λειτουργία αυτή.

5) Πελάτης.
Δημιουργεί και εκτελεί εντολές. Μια κλάση Πελάτη είναι αρμόδια για τη διαχείριση μιας συλλογής των αντικειμένων εντολής που δημιουργούνται από ένα αντικείμενο Invoker. Οι συγκεκριμένες ευθύνες μιας κλάσης Πελάτη μπορούν να διαχειρίζονται τις εντολές εκτέλεσης και αναίρεσης, τοποθετώντας τις εντολές σε σειρά, και σχεδιάζοντας τις. Οι κλάσεις Πελάτη είναι συνήθως ανεξάρτητες από τις εφαρμογές στις οποίες χρησιμοποιούνται και μπορούν να είναι επαναχρησιμοποιήσιμες.

Υπάρχουν μερικά ζητήματα που εξετάζουν πότε εφαρμόζεται το πρότυπο εντολής. Το πρώτο και ενδεχομένως το σημαντικότερο πρόκειται να αποφασίσει ποιες εντολές θα υπάρχουν. Εάν υπάρχουν ιδιαίτερα σύνθετες εντολές, θα υπάρξουν εξίσου σύνθετες κλάσεις εντολών.
Ένα άλλο ζήτημα εφαρμογής είναι η αποθήκευση-διατήρηση των πληροφοριών κατάστασης, απαραίτητη να αναιρέσει τις εντολές. Προκειμένου να είναι σε θέση να αναιρέσει τα αποτελέσματα μιας εντολής, είναι απαραίτητο να σωθούν αρκετές πληροφορίες της κατάστασης των αντικειμένων που λειτουργούν, για να είναι σε θέση για να αποκαταστήσει εκείνη την κατάσταση.
Μπορούν να υπάρξουν εντολές που δεν μπορούν να ανατραπούν, επειδή περιλαμβάνουν τη διάσωση ενός υπερβολικού ποσού πληροφοριών κατάστασης. Μπορούν να υπάρξουν εντολές που δεν μπορούν ποτέ να ανατραπούν, επειδή δεν είναι δυνατό να αποκατασταθεί η κατάσταση εκείνων των εντολών που αλλάζουν. Οι εντολές που περιλαμβάνουν τη διαγραφή των αρχείων ανήκουν συχνά σε αυτήν την κλάση. Το αντικείμενο Πελάτη πρέπει να γνωρίσει όταν εκτελείται μια εντολή που δεν μπορεί να αναιρεθεί.
Μπορεί να απλοποιηθεί το πρότυπο, εάν δεν πρέπει να υποστηρίζονται διαδικασίες αναίρεσης. Εάν καμία δεν απαιτείται καμία αναίρεση, τότε η κλάση Εντολή δεν χρειάζεται να καθορίσει μια μέθοδο αναίρεση.

Είναι γνωστό και ως πρότυπο Δράσης και Διεκπεραίωσης [12]. Το πρότυπο εντολής είναι χρήσιμο ως λύση σε διάφορα σενάρια, όπως η ανάγκη για συμπεριφορά διεξαγωγής- διεκπεραίωσης ή τις ενέργειες αναίρεσης (undoable actions). Ανεξάρτητα από τον τρόπο με τον οποίο μια εντολή χρησιμοποιείται, εφαρμόζει πάντα μια γνωστή διεπαφή που έχει, τουλάχιστον, μια μέθοδο (εκτέλεση ()) που περιέχει τη δράση. Στις περισσότερες περιπτώσεις, η εντολή έχει επίσης έναν δέκτη της δράσης, ένα πελάτη που δημιουργεί την εντολή, και έναν επικαλούμενο (invoker) που καλεί τη μέθοδο που τρέχει τη δράση. Ο δέκτης της δράσης, ο πελάτης, και ο επικαλούμενος (invoker) για μια αλλαγή εντολής, βασίζονται στον τρόπο με τον οποίο η εντολή χρησιμοποιείται [9].

Οι συνέπειες του προτύπου είναι οι εξής [7]:
1). Το αντικείμενο που επικαλείται μια εντολή δεν είναι το ίδιο αντικείμενο που εκτελεί μια εντολή. Αυτός ο χωρισμός παρέχει την ευελιξία στο συγχρονισμό και την αλληλουχία των εντολών. Η υλοποίηση των εντολών ως αντικείμενα σημαίνει ότι μπορούν να συλλεχθούν, να εξουσιοδοτηθούν, και να χειριστούν όπως οποιοδήποτε άλλο είδος αντικειμένου.
2). Αφού είναι σε θέση να συλλεχθεί και να ελεγχθεί η αλληλουχία των εντολών ,σημαίνει ότι μπορεί να χρησιμοποιηθεί το πρότυπο εντολής ως βάση ενός

μηχανισμού που υποστηρίζει τις μακροεντολές πληκτρολογίων (keyboard macros). Αυτό είναι ένας μηχανισμός που καταγράφει μια ακολουθία εντολών και επιτρέπει να επαναληφθεί αργότερα. Το πρότυπο εντολής μπορεί επίσης να χρησιμοποιηθεί για να δημιουργήσει άλλα είδη σύνθετων προτύπων.
3). Η προσθήκη νέων εντολών είναι συνήθως εύκολη, επειδή δεν σπάζει εξαρτήσεις.

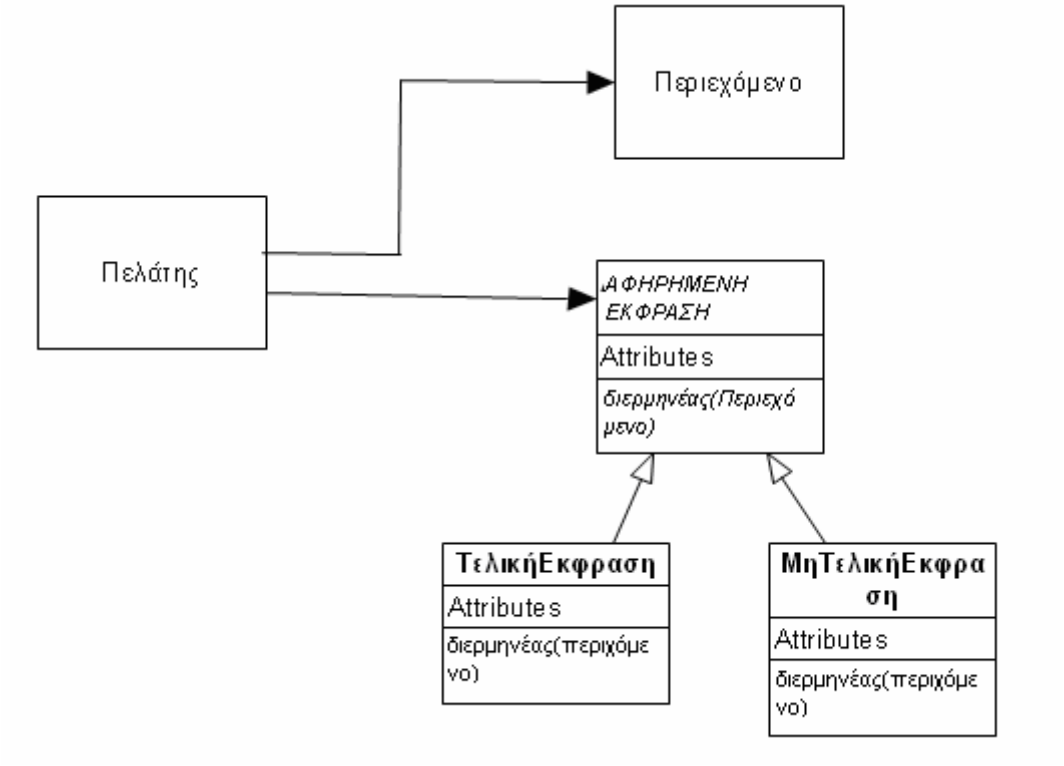
Στην εργασία με τίτλο ‘‘Killer “Killer Examples” for Design Patterns ’’ [13] παρουσιάζονται κάποια παραδείγματα, τα οποία βοηθούν τους νέους μαθητές να κατανοήσουν τα πρότυπα και τις λύσεις που αυτά προσφέρουν. Επίσης αναφέρεται ότι το πρότυπο αυτό χρησιμοποιείται για να αναπαραστήσει συμπεριφορές που συνδέονται με τα ιδιαίτερα κουμπιά στον χειριστή. Επειδή αυτές οι συμπεριφορές είναι σχετικές με αντικείμενα, όπως τα αντικείμενα εντολής, το σύστημα διατηρεί ευελιξία για να προσαρμόσει εύκολα νέες επιλογές με νέα χαρακτηριστικά γνωρίσματα [13].
Άλλα πρότυπα χρησιμοποιούνται σε συνδυασμό με το πρότυπο εντολής. Το πρότυπο σύνθεσης (Composite Pattern) για παράδειγμα μπορεί να χρησιμοποιηθεί για να επιτευχθούν μακρο-εντολές (macro commands). Το πρότυπο ενθυμίου (Memento Pattern) μπορεί να διατηρήσει την κατάσταση που χρειάζεται μια εντολή, έτσι ώστε να μπορεί να εφαρμοστεί η αναίρεση.
Στην εργασία με τίτλο ‘‘Applicability of the Command design pattern for Undo/Redo support An alternative: Undo via MVC ’’ [10] παρουσιάζεται ένας μηχανισμός για να υποστηρίζεται η επιλογή αναίρεση (undo) σε πολλά επίπεδα σε διάφορες εφαρμογές. Χρησιμοποιείται η αρχιτεκτονική του προτύπου Model View Controller.

5.3 Πρότυπο Διερμηνέα (Interpreter pattern)

Στον προγραμματισμό υπολογιστών, το πρότυπο διερμηνέα είναι ένα ιδιαίτερο σχεδιαστικό πρότυπο. Η βασική ιδέα είναι να υπάρξει μια κλάση για κάθε σύμβολο (τελικό ή όχι τελικό) σε μια εξειδικευμένη γλώσσα υπολογιστών, έτσι ώστε το δέντρο σύνταξης της γλώσσας, είναι ένα στιγμιότυπο του σύνθετου προτύπου [21].
Στην επιστήμη Υπολογιστών, ένα αφηρημένο δέντρο σύνταξης (AbstractSyntaxTree-AST), ή απλά δέντρο σύνταξης, είναι μια αναπαράσταση δέντρων της αφηρημένης (απλουστευμένης) συντακτικής δομής του πηγαίου κώδικα, ο οποίος γράφεται σε μια συγκεκριμένη γλώσσα προγραμματισμού. Κάθε κόμβος του δέντρου δείχνει ένα κατασκεύασμα που εμφανίζεται στον πηγαίο κώδικα. Η σύνταξη είναι αφηρημένη υπό την έννοια ότι δεν αναπαριστά κάθε λεπτομέρεια που εμφανίζεται στην πραγματική σύνταξη [22].
Το πρότυπο διερμηνέα καθορίζει μια γλώσσα ενός πεδίου (δηλ. χαρακτηρισμός προβλήματος) ως απλή γραμματική γλώσσας, που αναπαριστά τους κανόνες του πεδίου ως γλωσσικές προτάσεις, και που ερμηνεύει αυτές τις προτάσεις για να λύσει το πρόβλημα [23]. Το πρότυπο χρησιμοποιεί μια κλάση για να αναπαραστήσει κάθε κανόνα γραμματικής. Και αφού οι γραμματικές είναι συνήθως δομημένες ιεραρχικά, μπορεί να χρησιμοποιηθεί μια ιεραρχία κληρονομικότητας των κανόνων των κλάσεων.
Το πρότυπο αυτό είναι πολύ χρήσιμο για να προσθέτει λειτουργίες σε δομές σύνθετων προτύπων. Τα σύνθετα πρότυπα αναπαριστούν συστήματα με πολλά τμήματα που προσεγγίζονται ως ενιαίες αφηρημένες οντότητες. Το πρότυπο Διερμηνέα απαιτεί μια αφηρημένη μέθοδο στον υψηλό επίπεδο της σύνθετης δομής που χρησιμοποιείται από τα συστατικά του πελάτη, για να επεξεργαστεί την

ολόκληρη δομή. Συγκεκριμένες μέθοδοι εφαρμόζονται έτσι στα τελικά και όχι τελικά συστατικά για να κάνουν την πραγματική επεξεργασία.

Στο σχήμα 5.3.1 απεικονίζεται το διάγραμμα κλάσεων του πρότυπου διερμηνέα (Interpreter Pattern) [21][27][28][29].



Σχήμα 5.3.1 Απεικονίζεται το διάγραμμα κλάσεων του πρότυπου διερμηνέα (Interpreter Pattern).

Οι κλάσεις ή/και τα αντικείμενα που συμμετέχουν σε αυτό το πρότυπο είναι:
1). ΑφηρημένηΕκφραση.
Δηλώνει μια διεπαφή για την εκτέλεση μιας λειτουργίας. Η λειτουργία είναι κοινή για όλους τους κόμβους στο αφηρημένο συντακτικό δέντρο.

2). ΤελικήΕκφραση.
Εφαρμόζει μια μέθοδο διερμηνέας, που συνδέεται με τα τελικά σύμβολα στη γραμματική. Ένα αντικείμενο απαιτείται για κάθε τελικό σύμβολο στην πρόταση.

3). ΜηΤελικήΕκφραση.
Μια τέτοια κλάση απαιτείται για κάθε κανόνα R:: = R1R2… Rn στη γραμματική. Διατηρεί τις μεταβλητές αντικειμένου του τύπου ΑφηρημένηΕκφραση για κάθε ένα από τα σύμβολα R1 μέχρι Rn. Εφαρμόζει μια μέθοδο διερμηνέας για τα μη τερματικά σύμβολα στη γραμματική. Ο διερμηνέας καλεί κατ' επανάληψη τον εαυτό του, στις μεταβλητές που αναπαριστούν τα R1 μέχρι Rn.

4). Περιεχόμενο.
Περιέχει τις πληροφορίες που είναι όλες στο διερμηνέα.

5). Πελάτης.
Δημιουργεί ένα αφηρημένο δέντρο σύνταξης που αναπαριστά μια ξεχωριστή πρόταση στη γλώσσα που καθορίζει η γραμματική. Το αφηρημένο συντακτικό δέντρο συγκεντρώνεται από τα αντικείμενα των κλάσεων ΤελικήΕκφραση και ΜηΤελικήΕκφραση. Επικαλείται τη λειτουργία διερμηνέα.

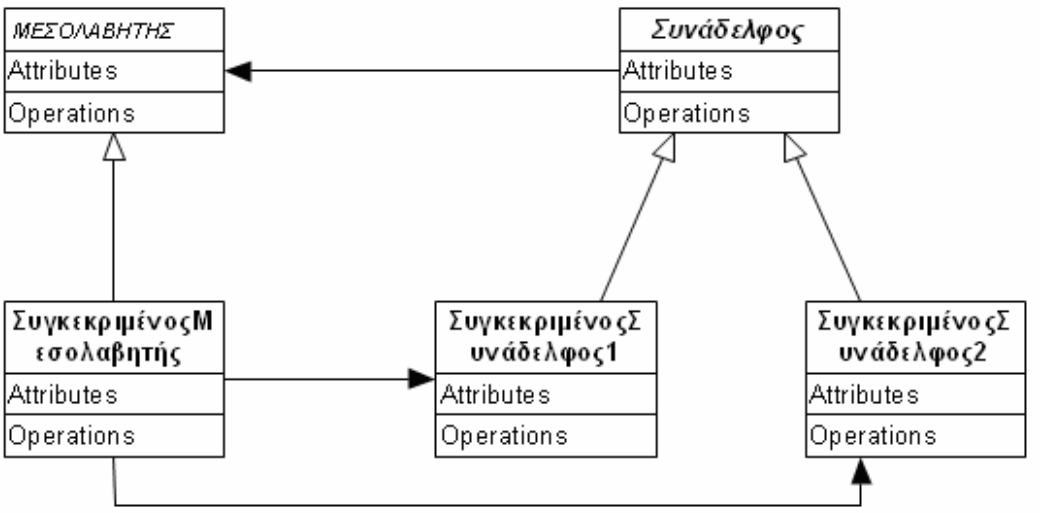
Μερικές από τις θετικές συνέπειες του προτύπου είναι ότι είναι εύκολο να αλλαχτεί και να επεκταθεί η γραμματική χρησιμοποιώντας κληρονομικότητα. Η εφαρμογή της γραμματικής είναι εύκολη, δεδομένου ότι οι κλάσεις είναι εύκολες να γραφθούν και μπορεί να αυτοματοποιηθεί η δημιουργία τους.

Το πρότυπο διερμηνέα χρησιμοποιείται και εξειδικεύεται σε γλώσσες διατύπωσης ερωτήσεων βάσεων δεδομένων όπως η SQL. Επίσης, ειδικεύεται στις γλώσσες υπολογιστών που χρησιμοποιούνται συχνά για να περιγράψουν τα πρωτόκολλα επικοινωνίας. Οι περισσότερες γλώσσες υπολογιστών γενικής χρήσης ενσωματώνουν διάφορες εξειδικευμένες γλώσσες [21].
Το σχεδιαστικό πρότυπο διορθωτικού διερμηνέα (Debuggable Interpreter Design Pattern), όπως αναλύεται στην εργασία ‘‘THE DEBUGGABLE INTERPRETER DESIGN PATTERN ’’ [24], περιγράφει έναν διερμηνέα γλώσσας προγραμματισμού που προσφέρει τη δυνατότητα διόρθωσης.
Επίσης, στο έγγραφο με τίτλο ‘‘Tiling Design Patterns -A Case Study Using the Interpreter Pattern ’’ [25] εξηγείται πώς τα πρότυπα μπορούν να χρησιμοποιηθούν για να περιγράψουν την εφαρμογή άλλων προτύπων. Περιγράφεται πώς ορισμένα σχεδιαστικά πρότυπα μπορούν να περιγράψουν τον σχεδιασμό τους. Η διαδικασία συναρμολόγησης προτύπων από άλλα πρότυπα ονομάζεται επικεράμωση προτύπων (pattern tiling). Η επικεράμωση επιτρέπει να αναμειχθούν οι απλές κατανοητές έννοιες των προτύπων με τη σύνθετη εφαρμογή τους στην πραγματικότητα. Αρκετές διαμορφώσεις επικεράμωσης για το πρότυπο διερμηνέα παρουσιάζονται στο έγγραφο [25].

5.4 Πρότυπο Μεσολαβητή (Mediator Pattern)

Το πρότυπο μεσολαβητή, παρέχει μια ενοποιημένη διεπαφή (unified interface) σε ένα σύνολο διεπαφών σε ένα υποσύστημα. Αυτό το πρότυπο θεωρείται ένα πρότυπο συμπεριφοράς λόγω του τρόπου που μπορεί να αλλάξει την τρέχουσα συμπεριφορά (running behavior) του προγράμματος [30].
Συνήθως ένα πρόγραμμα αποτελείται από έναν (μερικές φορές μεγάλο) αριθμό κλάσεων. Έτσι η λογική και ο υπολογισμός διανέμονται μεταξύ αυτών των κλάσεων. Όμως όσο περισσότερες κλάσεις αναπτύσσονται σε ένα πρόγραμμα, ειδικά κατά τη διάρκεια της συντήρησης, το πρόβλημα της επικοινωνίας μεταξύ αυτών των κλάσεων μπορεί να γίνει πιο σύνθετο. Αυτό καθιστά το πρόγραμμα δυσκολότερο να διαβαστεί και να διατηρηθεί. Επιπλέον, μπορεί να γίνει δύσκολο να αλλαχτεί το πρόγραμμα, δεδομένου ότι οποιαδήποτε αλλαγή μπορεί να έχει επιπτώσεις στον κώδικα σε διάφορες άλλες κλάσεις.
Με το πρότυπο μεσολαβητή η επικοινωνία μεταξύ των αντικειμένων ενθυλακώνεται με ένα αντικείμενο μεσολαβητή. Τα αντικείμενα δεν επικοινωνούν πλέον άμεσα το ένα με το άλλο, αλλά αντί για αυτό επικοινωνούν μέσω του μεσολαβητή. Αυτό μειώνει τις εξαρτήσεις μεταξύ της επικοινωνίας των αντικειμένων, μειώνοντας τη σύζευξη [33].

Ο μεσολαβητής (Mediator) είναι ένα σχεδιαστικό πρότυπο συμπεριφοράς που παρέχει ένα κεντρικό κόμβο (hub) για να καθοδηγήσει τις αλληλεπιδράσεις μεταξύ πολλών αντικειμένων. Σύμφωνα με το βιβλίο ‘‘Design Patterns: Elements of Reusable Object-Oriented Software’’[6], η πρόθεση του προτύπου μεσολαβητή είναι: «Να καθοριστεί ένα αντικείμενο που ενθυλακώνει τον τρόπο που αλληλεπιδρά ένα σύνολο αντικειμένων. Ο μεσολαβητής προάγει τη χαλαρή σύζευξη, αποτρέποντας τα αντικείμενα να αναφέρεται το ένα στο άλλο, και επιτρέπει να διαφοροποιηθεί η αλληλεπίδρασή τους ανεξάρτητα.»
Ένα παράδειγμα για να γίνει κατανοητό το πρότυπο είναι το εξής. Ο πύργος ελέγχου σε ένα ελεγχόμενο αεροδρόμιο δείχνει πολύ καλά αυτό το πρότυπο. Οι πιλότοι των αεροπλάνων που πλησιάζουν ή που αναχωρούν από την τελική-τερματική περιοχή, επικοινωνούν με τον πύργο αντί να επικοινωνούν το ένα με το άλλο. Οι περιορισμοί σε ποιος μπορεί να απογειωθεί ή να προσγειωθεί επιβάλλονται από τον πύργο. Είναι σημαντικό να σημειωθεί ότι ο πύργος δεν ελέγχει ολόκληρη την πτήση. Υπάρχει μόνο για να επιβάλει τους περιορισμούς στην τελική περιοχή [32] [33] [34].
Το διάγραμμα κλάσεων του προτύπου μεσολαβητή παρουσιάζεται στο επόμενο σχήμα [30] [31].



Σχήμα 5.4.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου μεσολαβητή.

Οι κλάσεις και οι διεπαφές που συμμετέχουν στο διάγραμμα είναι οι εξής [7] [30] [31]:

1). Μεσολαβητής (Mediator).
Καθορίζει τη διεπαφή για την επικοινωνία μεταξύ των αντικειμένων της κλάσεως Συνάδελφος. Ο Μεσολαβητής ενημερώνεται για τα γεγονότα που συμβαίνουν.

2). ΣυγκεκριμένοςΜεσολαβητής.
Εφαρμόζει τη διεπαφή Μεσολαβητή και ορίζει την επικοινωνία μεταξύ των αντικειμένων των Συναδέλφων. Γνωρίζει όλους τους Συναδέλφους και το σκοπό τους σε σχέση με τη επικοινωνία.

3). Κλάσεις Συναδέλφων.
Κάθε κλάση Συνάδελφος γνωρίζει το αντικείμενο Μεσολαβητή του. Κάθε Συνάδελφος επικοινωνεί με άλλους Συναδέλφους μέσω του μεσολαβητή του.

Προσφέρει υπηρεσίες-αιτήσεις στον μεσολαβητή. Μπορεί να υπάρχουν αιτήσεις που είναι κοινές σε όλους τους μεσολαβητές.

Μερικά πλεονεκτήματα του προτύπου είναι τα ακόλουθα:

1). Δεδομένου ότι ο μεσολαβητής κρύβει όλες τις δραστηριότητες συντονισμού που εφαρμόζει, ο χρήστης είναι ικανότερος να καταλάβει το σύστημα και τις αλληλεπιδράσεις που έχουν οι συνάδελφοι.
2). Οι συνάδελφοι μπορούν να προστεθούν, να απομακρυνθούν, να τροποποιηθούν ευκολότερα αφού εξαρτώνται λιγότερο ο ένας από τον άλλον. Αυτό ισχύει επίσης μεταξύ του συναδέλφου και του μεσολαβητή του που μπορούν να χρησιμοποιηθούν ανεξάρτητα.
3). Μόνο οι μεσολαβητές χρειάζεται δημιουργήσουν υποκλάσεις όταν αλλάζει η συμπεριφορά. Οι συνάδελφοι μπορούν να παραμείνουν οι ίδιοι.

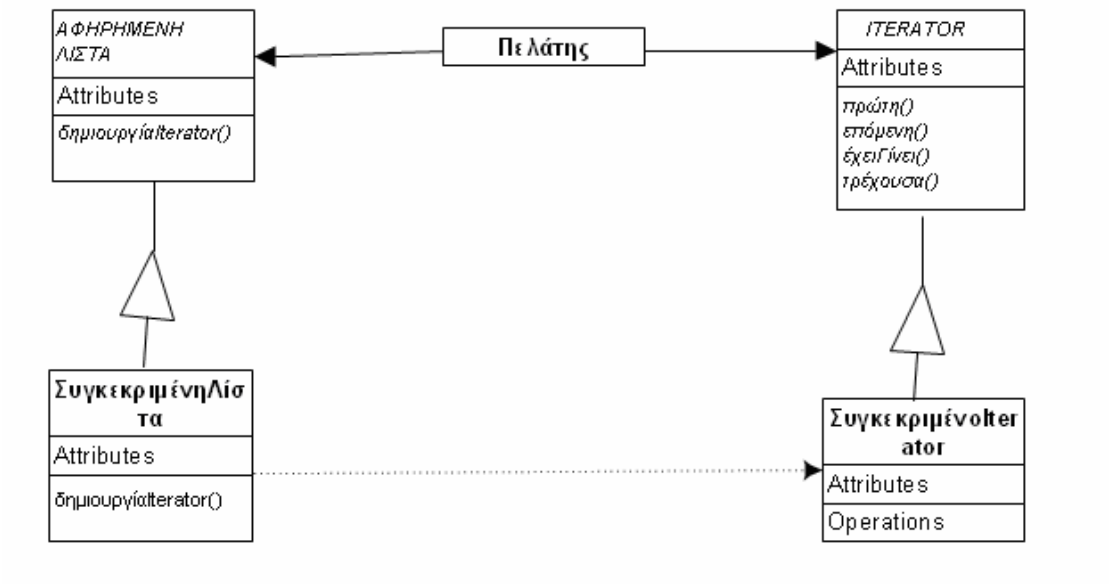
Ένα σημαντικό μειονέκτημα του προτύπου είναι η πολυπλοκότητα. Επειδή ο μεσολαβητής μπορεί να χειριστεί έναν ενδεχομένως μεγάλο αριθμό συναδέλφων, το περιεχόμενο του μεσολαβητή μπορεί να είναι πολύ σύνθετο. Κατά συνέπεια, θα μπορούσε να είναι δύσκολο να τροποποιηθεί και να γίνει κατανοητό το περιεχόμενό του.

Μερικά άλλα σχεδιαστικά πρότυπα είναι στενά συνδεδεμένα με το πρότυπο μεσολαβητή, και χρησιμοποιούνται συχνά με αυτό.
Το πρότυπο πρόσοψης (Façade Pattern) παρέχει μια ενοποιημένη διεπαφή σε ένα σύνολο διεπαφών σε ένα υποσύστημα. Η πρόσοψη καθορίζει μια υψηλότερου επιπέδου διεπαφή που καθιστά το υποσύστημα ευκολότερο να χρησιμοποιηθεί [6]. Η πολιτική του προτύπου πρόσοψης είναι φανερή, ενώ η πολιτική του μεσολαβητή είναι κρυφή.
Στο πρότυπο παρατηρητή (Observer Pattern) όσοι συνεργάζονται, μπορούν να επικοινωνήσουν με έναν μεσολαβητή χρησιμοποιώντας το πρότυπο παρατηρητή. Στην πραγματικότητα, το πρότυπο του παρατηρητή χρησιμοποιείται πολύ συχνά μαζί με το πρότυπο μεσολαβητή, δεδομένου ότι παρέχει μια δυνατότητα, σύμφωνα με την οποία, είναι καλό ο μεσολαβητής της ευθύνης να πρέπει να ξέρει ακριβώς ποιες κλάσεις πρέπει να ενημερωθούν για τα γεγονότα. Ο μεσολαβητής και ο παρατηρητής είναι ανταγωνιστικά πρότυπα. Η διαφορά μεταξύ τους είναι ότι ο παρατηρητής διανέμει την επικοινωνία με την εισαγωγή των αντικειμένων «παρατηρητών» και «θεμάτων», ενώ ένα αντικείμενο μεσολαβητή ενθυλακώνει την επικοινωνία μεταξύ άλλων αντικειμένων [33].

5.5 Πρότυπο Iterator (Iterator Pattern)

Το Iterator είναι ένα από τα πιο απλά και πιο συχνά χρησιμοποιούμενα σχεδιαστικά πρότυπα. Το πρότυπο Iterator επιτρέπει την κίνηση μέσω ενός καταλόγου/λίστας ή μιας συλλογής στοιχείων χρησιμοποιώντας μια τυποποιημένη διεπαφή. Δεν είναι απαραίτητο να είναι γνωστές οι λεπτομέρειες της εσωτερικής αναπαράστασης των στοιχείων [18].
Παρέχει έναν τρόπο να υπάρχει πρόσβαση στα στοιχεία ενός συνολικού/αθροιστικού αντικειμένου διαδοχικά χωρίς εκτίθεται η εσωτερική δομή του. Ένα συνολικό αντικείμενο είναι ένα αντικείμενο, το οποίο περιέχει άλλα αντικείμενα, με σκοπό να

ομαδοποιήσει τα αντικείμενα αυτά σαν μια μονάδα. Είναι γνωστό και ως συλλογή (collection- container) [17].
Το πρότυπο Iterator χρησιμοποιείται όταν ,ένα συνολικό αντικείμενο όπως ένας κατάλογος (list) πρέπει να έχει έναν τρόπο να προσπελαστούν/διαβαστούν τα στοιχεία του χωρίς να εκτεθεί η εσωτερική δομή του [19]. Επίσης ανάλογα με το σκοπό, μπορεί τα στοιχεία να πρέπει να διαβαστούν με διαφορετική σειρά ή και διαφορετικό τρόπο. Χρησιμοποιείται επίσης όταν πρέπει να επιτρέπεται προσπέλαση από διαφορετικές μεθόδους [20]. Μια ακόμα περίπτωση είναι για να παρέχει μια ομοιόμορφη διεπαφή για να προσπελαστούν διαφορετικές συνολικές δομές.
Η βασική ιδέα σε αυτό το πρότυπο είναι η ευθύνη για την πρόσβαση και προσπέλαση στον κατάλογο-λίστα του αντικειμένου, να ανήκει σε ένα αντικείμενο iterator. Η κλάση Iterator καθορίζει μια διεπαφή για την πρόσβαση στα στοιχείων του καταλόγου. Ένα αντικείμενο iterator είναι αρμόδιο για την παρακολούθηση του τρέχοντος στοιχείου, δηλαδή ξέρει ποια στοιχεία έχουν ήδη προσπελαστεί. Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του πρότυπου Iretator.



Σχήμα 5.5.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Iterator.

Εδώ είναι οι περιγραφές των ρόλων που παίζουν οι κλάσεις και οι διεπαφές στην οργάνωση που παρουσιάζεται στο σχήμα 5.5.1.

1). Η κλάση ΣυγκεκριμένηΛίστα ενθυλακώνει μια συλλογή των αντικειμένων.
2). Η διεπαφή Iterator καθορίζει τις μεθόδους, για να έχει πρόσβαση και προσπέλαση στα αντικείμενα που ενθυλακώνονται σε μια δομή συνάθροισης.
3). Η κλάση ΣυγκεκριμένοIterator εφαρμόζει μια διεπαφή Iterator. Τα στιγμιότυπά της παρέχουν διαδοχική πρόσβαση στο περιεχόμενο του αντικειμένου συλλογής που συνδέεται με το αντικείμενο Iterator. Γνωρίζουν κάθε φορά σε ποιο σημείο βρίσκεται η προσπέλαση.
4). Η διεπαφή ΑφηρημένηΛίστα δημιουργεί το αντικείμενο Iterator [20].
5). Ο Πελάτης χειρίζεται την διεπαφή Iterator με τη μέθοδο Επόμενη().

Η βασικότερη συνέπεια του Iterator προτύπου είναι ότι απλοποιεί τη διεπαφή του Συνόλου χωρίς να επιβαρύνεται με τις μεθόδους προσπέλασης. Επίσης υποστηρίζει διάφορες τεχνικές προσπέλασης και διαφορετικές και ταυτόχρονες προσπελάσεις.

5.6 Πρότυπο Παρατηρητή (Observer Pattern)

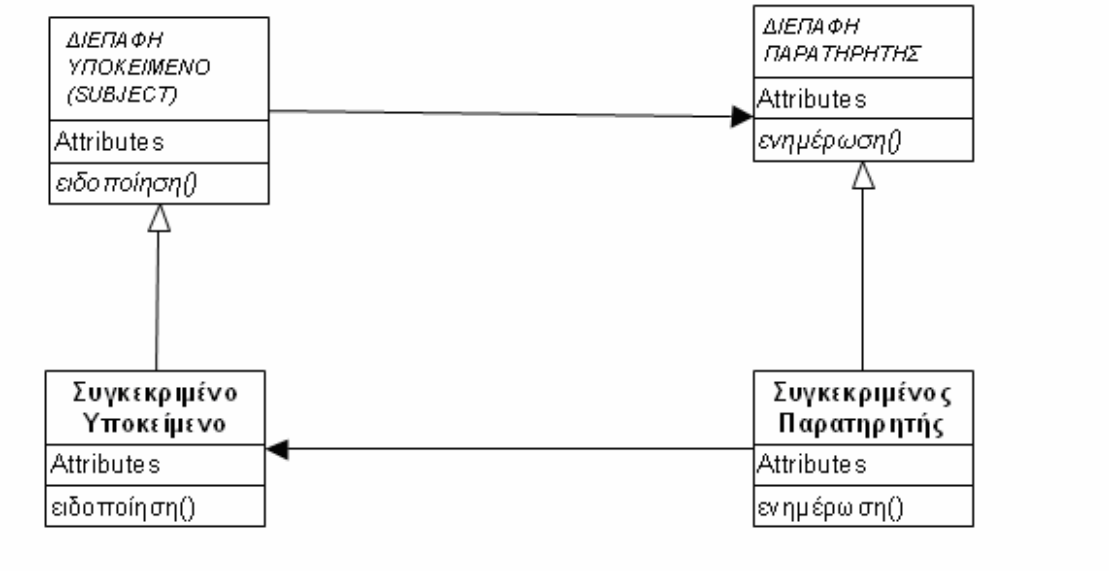
Μια από τις εξαιρετικά σημαντικές αρχές της αντικειμενοστρεφούς ανάπτυξης είναι η κατάλληλη ανάθεση αρμοδιότητας για τη κάθε εφαρμογή. Κάθε αντικείμενο στο σύστημα πρέπει να εστιάσει σε μια ιδιαίτερη αφαίρεση μέσα στην περιοχή προβλήματος και τίποτα περισσότερο. Εν ολίγοις, ένα αντικείμενο πρέπει να κάνει ένα πράγμα και το κάνει καλά. Αυτή η προσέγγιση εξασφαλίζει ότι υπάρχει ένα όριο μεταξύ των αντικειμένων, επιτρέποντας τη μεγαλύτερη συντηρησιμότητα επαναχρησιμοποίηση των συστημάτων [37].
Η ανάγκη να παρασχεθεί ένα ευδιάκριτο όριο μεταξύ του ενδιάμεσου με τον χρήστη και της λογικής είναι ένα πρόβλημα που υπάρχει στις εφαρμογές. Κατά συνέπεια, διάφορα αντικειμενοστρεφή πλαίσια αναπτύχθηκαν δεδομένου ότι η έναρξη του GUI υποστηρίζει το χωρισμό του ενδιάμεσου με τον χρήστη από το υπόλοιπο της εφαρμογής. Όπως ήταν αναμενόμενο, τα περισσότερα από αυτά υιοθέτησαν ένα παρόμοιο σχεδιαστικό πρότυπο για να παρέχουν αυτήν την λειτουργία. Αυτό το πρότυπο, συνήθως γνωστό ως παρατηρητής (observer) και βοηθά στη δημιουργία μιας σαφούς διάκρισης μεταξύ των διάφορων αντικειμένων στο σύστημα. Επιπλέον, δεν είναι ασυνήθιστο να βρεθεί αυτή η λύση να χρησιμοποιείται μέσα στα μη UI (user interface) τμήματα ενός πλαισίου ή μιας εφαρμογής. Όπως με τα περισσότερα άλλα πρότυπα, η χρησιμότητα του προτύπου παρατηρητή επεκτείνεται αρκετά πέρα από την αρχική πρόθεσή του.
Σε περιπτώσεις που κάποια αντικείμενα (παρατηρητές) ενδιαφέρονται να λαμβάνουν ειδοποιήσεις για τυχόν αλλαγές στην κατάσταση κάποιου άλλου αντικειμένου Α (π.χ. τροποποιήσεις τιμών κάποιων πεδίων του), υπάρχουν δύο προσεγγίσεις για την υλοποίηση αυτών των ενημερώσεων: είτε τις κατάλληλες στιγμές το Α να καλεί προκαθορισμένες μεθόδους των παρατηρητών και να τους μεταβιβάζει έτσι δεδομένα, είτε οι παρατηρητές να καλούν μία μέθοδο του Α για να λαμβάνουν κατά βούληση πληροφορίες. Η ενδιαφέρουσα περίπτωση είναι η πρώτη καθώς μόνον το Α γνωρίζει πότε υπάρχουν αλλαγές στην κατάσταση του και τέτοιες αλλαγές είναι που πρέπει να πυροδοτούν τις ενημερώσεις. Ο απλούστερος τρόπος είναι να διατηρεί μία συνδεδεμένη λίστα με όλους τους παρατηρητές που κατά καιρούς έχουν εκδηλώσει ενδιαφέρον και να καλεί τις κατάλληλες μεθόδους τους τις κατάλληλες στιγμές. Όμως αυτή η λύση προκαλεί προβλήματα κλειστότητας καθώς ένας άλλος τύπος παρατηρητή μπορεί να έχει διαφορετικές μεθόδους, ενώ το Α παρουσιάζει υψηλή σύζευξη με άλλες κλάσεις [3].
Το πρότυπο παρατηρητή δίνει μία λύση σε αυτό το πρόβλημα. Ορίζει μία διασύνδεση, η οποία περιέχει ορισμένες μεθόδους. Η κλάση κάθε αντικειμένου Α πρέπει να υλοποιεί τη διασύνδεση αυτή και η κλάση κάθε παρατηρητή τη διασύνδεση παρατηρητή [3]. Χρησιμοποιείται συνήθως για τις δυναμικές σχέσεις μεταξύ των αντικειμένων.

Το πρότυπο παρατηρητή χρησιμοποιείται σε οποιεσδήποτε από τις ακόλουθες τρεις καταστάσεις [41]:

1). Όταν μια αφαίρεση έχει δύο πτυχές, μια εξαρτώμενη από άλλη. Η τοποθέτηση των πτυχών σε χωριστά αντικείμενα επιτρέπει να επαναχρησιμοποιηθούν ανεξάρτητα.
2). Όταν μια αλλαγή σε ένα αντικείμενο απαιτεί και αλλαγή άλλων, και δεν είναι γνωστό πόσα αντικείμενα πρέπει να αλλαχτούν.

3). Όταν ένα αντικείμενο πρέπει να είναι σε θέση να δηλώσει άλλα αντικείμενα χωρίς να κάνει υποθέσεις για το ποιο από αυτά τα αντικείμενα είναι. Με άλλα λόγια, δεν είναι επιθυμητό αυτά τα αντικείμενα να συνδέονται στενά.

Στο σχήμα 5.6.1 απεικονίζεται το διάγραμμα κλάσεων του προτύπου παρατηρητή [7] [38] [39] [40] [41].



Σχήμα 5.6.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου παρατηρητή.

Οι κλάσεις και οι διεπαφές που συμμετέχουν στο πρότυπο παρατηρητή είναι οι εξής:

1). Υποκείμενο
Γνωρίζει τους παρατηρητές του. Οποιοσδήποτε αριθμός αντικειμένων παρατηρητών μπορεί να παρατηρήσει ένα υποκείμενο. Παρέχει μια διεπαφή για την ένωση και την αποσύνδεση των αντικειμένων παρατηρητών.

2). Παρατηρητής
Καθορίζει μια διεπαφή η οποία ενημερώνεται, για αντικείμενα που πρέπει να δηλωθούν για τις αλλαγές σε ένα υποκείμενο. Όλοι οι πιθανοί παρατηρητές πρέπει να εφαρμόσουν τη διεπαφή Παρατηρητή. Αυτή η διεπαφή έχει μόνο την μέθοδο ενημέρωση, η οποία καλείται όταν η κατάσταση του Υποκειμένου αλλάξει [42].

3). ΣυγκεκριμένοΥποκείμενο
Το ΣυγκεκριμένοΥποκείμενο εφαρμόζει πάντα τη διεπαφή Υποκείμενο. Αποθηκεύει την κατάσταση ενδιαφέροντος στα αντικείμενα ΣυγκεκριμένοςΠαρατηρητής. Στέλνει μια ανακοίνωση στους παρατηρητές του όταν αλλάζει η κατάστασή του. Αυτό γίνεται με τη μέθοδο ειδοποίηση.

4). ΣυγκεκριμένοςΠαρατηρητής
Διατηρεί μια αναφορά σε ένα αντικείμενο ΣυγκεκριμένοΥποκείμενο. Αποθηκεύει την κατάσταση που πρέπει να μείνει σύμφωνη με του Υποκειμένου. Εφαρμόζει την διεπαφή παρατηρητή που ενημερώνεται για να κρατήσει την κατάστασή του σύμφωνη με του υποκειμένου.

Τα βασικά αντικείμενα σε αυτό το πρότυπο είναι το Υποκείμενο και ο Παρατηρητής. Ένα Υποκείμενο μπορεί να έχει οποιοδήποτε αριθμό εξαρτώμενων Παρατηρητών. Όλοι οι Παρατηρητές δηλώνονται, όταν το Υποκείμενο υποβάλλεται σε μια αλλαγή κατάστασης. Έτσι, κάθε Παρατηρητής θα ρωτήσει το Υποκείμενο για να συγχρονίσει την κατάσταση του με την κατάσταση του Υποκειμένου.
Αυτό το είδος αλληλεπίδρασης είναι επίσης γνωστό ως εκδίδει- προσυπογράφει (publish-subscribe) . Το Υποκείμενο είναι ο εκδότης των ανακοινώσεων. Στέλνει αυτές τις ανακοινώσεις χωρίς να πρέπει να είναι γνωστό ποιοι είναι οι παρατηρητές του. Οποιοσδήποτε αριθμός παρατηρητών μπορεί να προσυπογράψει για να λάβει τις ανακοινώσεις.

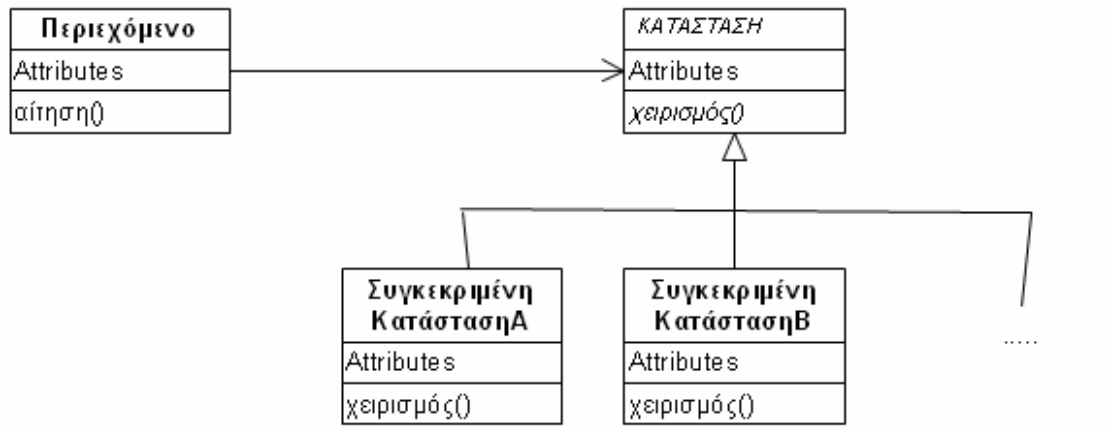
Το πρότυπο παρατηρητή έχει ορισμένα πλεονεκτήματα. Το πρότυπο παρατηρητή είναι βασικά ένας τρόπος να διατηρηθούν εφαρμογές που βασίζονται σε δράσεις σε μεγάλες αρχιτεκτονικές. Σε οποιαδήποτε δεδομένη εφαρμογή μπορεί να υπάρχουν δεκάδες, εκατοντάδες, ή ακόμα και χιλιάδες γεγονότα που συμβαίνουν σποραδικά σε μια μηχανή αναζήτησης. Επιπλέον, μπορεί να μειωθεί η σύνδεση γεγονότων και να επιτραπεί σε αντικείμενα να χειριστούν τις ενέργειες για τον χειριστή, μειώνοντας τη μνήμη και επιταχύνοντας την απόδοση αλληλεπίδρασης [44].
Επίσης το Υποκείμενο ξέρει ότι υπάρχει ένας κατάλογος παρατηρητών, κάθε ένας που προσαρμόζεται στην απλή διεπαφή της αφηρημένης κλάσης παρατηρητών. Το Υποκείμενο δεν ξέρει τη συγκεκριμένη κλάση οποιουδήποτε παρατηρητή. Κατά συνέπεια η σύζευξη μεταξύ των υποκειμένων και των παρατηρητών είναι αφηρημένη και ελάχιστη.
Επειδή το υποκείμενο και ο παρατηρητής δεν συνδέονται στενά, μπορούν να ανήκουν σε διαφορετικά στρώματα της αφαίρεσης σε ένα σύστημα. Ένα χαμηλότερο υποκείμενο μπορεί να επικοινωνήσει και να ενημερώσει έναν υψηλότερου επιπέδου παρατηρητή, κρατώντας έτσι τη διάταξη στα στρώματα του συστήματος άθικτη. Εάν το υποκείμενο και ο παρατηρητής συσσωρεύονται μαζί, τότε το προκύπτον αντικείμενο πρέπει, είτε να εκταθεί δύο στρώματα (και να παραβιάσουν τη διάταξη σε στρώματα), είτε πρέπει να αναγκαστεί να υπάρξει σε κάποιο στρώμα.
Αντίθετα από ένα συνηθισμένο αίτημα, η ανακοίνωση που στέλνει ένα υποκείμενο δεν χρειάζεται να διευκρινίσει το δέκτη της. Η ανακοίνωση μεταδίδεται αυτόματα σε όλα τα ενδιαφερόμενα αντικείμενα. Το υποκείμενο δεν φροντίζει πόσα ενδιαφερόμενα αντικείμενα υπάρχουν. Η μόνη ευθύνη του είναι να ειδοποιήσει τους παρατηρητές της. Αυτό δίνει την ελευθερία να προσθέσει και να απομακρύνει τους παρατηρητές οποιαδήποτε στιγμή. Εξαρτάται από τον παρατηρητή να αντιμετωπίσει ή να αγνοήσει μια ανακοίνωση.

Ένα μειονέκτημα στη χρησιμοποίηση αυτής της διεπαφής παρατηρητή είναι το κόστος στο χρόνο φόρτωσης (load time) όταν δημιουργούνται τα αντικείμενα. Αυτό μπορεί να μετριαστεί - μειωθεί λίγο με τη χρησιμοποίηση μιας τεχνικής αποκαλούμενης οκνηρή φόρτωση (lazy loading), η οποία επιτρέπει να αναβληθεί η δημιουργία των νέων αντικείμενων μέχρι το χρόνο παράδοσης.

Χρησιμοποιώντας την προσανατολισμένη πτυχή προγραμματισμού, το έγγραφο ‘‘The Observer Pattern Using Aspect Oriented Programming ’’ [43] παρέχει μια νέα λύση και μια επαναχρησιμοποιήσιμη εφαρμογή του προτύπου παρατηρητή. Οι γλώσσες εφαρμογής είναι AspectJ και AspectJ.

5.7 Πρότυπο Κατάστασης (State Pattern)

Το πρότυπο κατάστασης είναι ένα σχεδιαστικό πρότυπο συμπεριφοράς λογισμικού, επίσης γνωστό ως αντικείμενο για το πρότυπο κατάστασης. Αυτό το πρότυπο χρησιμοποιείται στον προγραμματισμό υπολογιστών για να αναπαραστήσει την κατάσταση ενός αντικειμένου. Αυτός είναι ένας καθαρός τρόπος για ένα αντικείμενο να αλλαχτεί μερικώς ο τύπος του στο χρόνο εκτέλεσης [45].
Τα αντικείμενα συζητούνται συχνά από την άποψη ότι έχουν μια «κατάσταση» που περιγράφει τις ακριβείς τους συνθήκες σε μια δεδομένη στιγμή, που βασίζονται στις τιμές των ιδιοτήτων τους. Οι ιδιαίτερες τιμές των ιδιοτήτων έχουν επιπτώσεις στη συμπεριφορά του αντικειμένου [46].
Πολλά αντικείμενα απαιτούνται για να έχουν ένα δυναμικά μεταβαλλόμενο σύνολο ιδιοτήτων αποκαλούμενο κατάσταση. Τέτοια αντικείμενα καλούνται stateful αντικείμενα. Η κατάσταση ενός αντικειμένου είναι συνήθως ένα προκαθορισμένο σύνολο τιμών. Όταν ένα stateful αντικείμενο ενημερώνεται για ένα εξωτερικό γεγονός, η κατάστασή του μπορεί να αλλάξει. Η συμπεριφορά ενός stateful αντικειμένου καθορίζεται με κάποιους τρόπους από την κατάστασή του [7].
Για παράδειγμα, κάποιος μπορεί να πει ότι η ακριβής συμπεριφορά της μεθόδου δώσεΧρώμα() ενός αντικειμένου είναι διαφορετική εάν το χρώμα του αντικειμένου τίθεται « μπλε» αντί « κόκκινου» επειδή η δώσεΧρώμα() επιστρέφει μια διαφορετική τιμή στις δύο καταστάσεις.
Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου κατάστασης [7] [45] [46] [47] [48].



Σχήμα 5.7.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου κατάστασης.

Εδώ επεξηγούνται οι ρόλοι των κλάσεων στο πρότυπο κατάστασης.

1). Περιεχόμενο
Καθορίζει τη διεπαφή που ενδιαφέρει τον πελάτη. Τα στιγμιότυπα του Περιεχομένου καθορίζουν τη τρέχουσα κατάσταση τους, διατηρώντας μια αναφορά σε ένα στιγμιότυπο μιας συγκεκριμένης υποκλάσης της κλάσης Κατάσταση. Η υποκλάση της κλάσης Κατάσταση καθορίζει την κατάσταση [7].

2). Κατάσταση
Η κλάση Κατάσταση είναι η υπέρ-κλάση όλων των κλάσεων που χρησιμοποιούνται για να αναπαραστήσουν την κατάσταση των αντικειμένων Περιεχόμενο. Μια κλάση Κατάστασης καθορίζει τις μεθόδους που χρειάζεται.

3). ΣυγκεκριμένηΚατάστασηΑ, ΣυγκεκριμένηΚατάστασηΒ και ούτω καθεξής.
Αυτές είναι συγκεκριμένες υποκλάσεις της Κατάστασης. Πρέπει να εφαρμόσουν τις μεθόδους με έναν κατάλληλο τρόπο η κάθε μια. Δηλαδή κάθε υποκλάση εφαρμόζει μια συμπεριφορά που σχετίζεται με μια κατάσταση του Περιεχομένου.

Μερικές συνέπειες του προτύπου αυτού είναι ότι ο κώδικας για κάθε κατάσταση βρίσκεται μέσα στην κλάση του. Αυτή η οργάνωση το καθιστά εύκολο να προστεθούν νέες καταστάσεις χωρίς δυσάρεστες συνέπειες. Για αυτόν τον λόγο, το πρότυπο κατάστασης λειτουργεί καλά για μικρούς και μεγάλους αριθμούς καταστάσεων. Επίσης, η χρησιμοποίηση του προτύπου αυτού οδηγεί σε λιγότερες γραμμές κώδικα και λιγότερες πορείες εκτέλεσης [7]. Το πρότυπο κατάστασης είναι ένας πολύ καλός τρόπος να ρυθμιστούν οι καταστάσεις μέσα στην εφαρμογή. Με την ενθυλάκωση των καταστάσεων στα αντικείμενά τους, καθίσταται ο κώδικας πιο εξελικτικός και συντηρήσιμος [9].

Στο έγγραφο ‘‘A Pattern Based Approach to Aspect-Orientation for State Based Systems ’’ παρουσιάζει τα οφέλη του προσανατολισμού-πτυχής (Aspect-Orientation) σε συστήματα βασισμένα στην κατάσταση, χρησιμοποιώντας πρότυπα αντί των πτυχή-προσανατολισμένων (Aspect-Oriented) γλωσσών προγραμματισμού, πλαισίων, ή και εργαλείων. Υποσυστήματα βασισμένα στη κατάσταση εφαρμόζονται με το πρότυπο κατάστασης αλληλεπιδρούν με τη δέσμευση των γεγονότων από μηχανές κατάστασης. Η δέσμευση εμφανίζεται χρησιμοποιώντας τα πρότυπα μεσολαβητή και αφηρημένου εργοστασίου [49].
Επίσης το έγγραφο με τίτλο ‘‘The Reflective State Pattern ’’ [50] παρουσιάζει το πρότυπο ανακλαστικής κατάστασης, το οποίο μοιάζει με το σχεδιαστικό πρότυπο κατάστασης αλλά είναι βασισμένο στο αρχιτεκτονικό πρότυπο ανάκλασης. Το εν λόγω πρότυπο προτείνει μια λύση για μερικές σχεδιαστικές αποφάσεις που πρέπει να ληφθούν για να μπορεί να εφαρμοστεί το πρότυπο κατάστασης επιτυχημένα.
Ένα ακόμη σημαντικό έγγραφο παρουσιάζει ένα νέο αντικειμενοστρεφές σχεδιαστικό πρότυπο, το σχεδιαστικό πρότυπο κατάστασης μηχανής. Αυτό το πρότυπο επεκτείνει τις ικανότητες του σχεδιαστικού προτύπου κατάστασης. Αυτά τα πρότυπα επιτρέπουν σε ένα αντικείμενο να αλλάξει τη συμπεριφορά του όταν αλλάζει η εσωτερική κατάστασή του. Οι κλάσεις που σχεδιάζονται με το σχεδιαστικό πρότυπο κατάστασης μηχανής είναι πιο επαναχρησιμοποιήσιμες από τους αυτές που σχεδιάζονται με το πρότυπο κατάστασης [51].

5.8 Strategy pattern (Πρότυπο Στρατηγικής)

Όταν είναι διαθέσιμοι πολλαπλοί τρόποι επίλυσης ενός προβλήματος, για παράδειγμα διαφορετικοί αλγόριθμοι, είναι προτιμότερο ο καθένας από αυτούς να μην υλοποιείται μέσα στις κλάσεις-πελατών που τον χρησιμοποιούν (π.χ. ως ιδιωτική μέθοδος), έτσι ώστε οι πελάτες να έχουν μικρότερη πολυπλοκότητα και οι διάφοροι αλγόριθμοι να είναι επαναχρησιμοποιήσιμοι και προσπελάσιμοι από πολλά εξωτερικά προγράμματα. Με το πρότυπο Στρατηγικής (Strategy Pattern) όλοι οι διαφορετικοί αλγόριθμοι ορίζονται ως ξεχωριστές κλάσεις που υλοποιούν μία κοινή διασύνδεση Α

και οι πελάτες διατηρούν ως πεδίο μία αναφορά προς τον αφηρημένο τύπο Α. Στο πεδίο αυτό δίνεται τιμή μέσω του ορίσματος κάποιας μεθόδου του πελάτη (πιθανώς του κατασκευαστή του), έτσι ώστε η ταυτότητα του εκάστοτε χρησιμοποιούμενου αλγορίθμου από όλους τους διαθέσιμους για την εκτέλεση της ζητούμενης εργασίας να είναι εύκολα παραμετροποιήσιμη, με μία απλή αλλαγή αυτού του ορίσματος κατά την κλήση της προαναφερθείσας μεθόδου [52].

Το σχεδιαστικό πρότυπο στρατηγικής αποτελείται βασικά από την αποσύζευξη-αποσύνδεση ενός αλγορίθμου από τον οικοδεσπότη του (host), και ενθυλάκωση του αλγορίθμου σε μια χωριστή κλάση. Πιο απλά, ένα αντικείμενο και η συμπεριφορά του είναι χωρισμένα και τοποθετούνται σε δύο διαφορετικές κλάσεις. Αυτό επιτρέπει να μετατραπεί ο αλγόριθμος που χρησιμοποιείται οποιαδήποτε στιγμή [53].

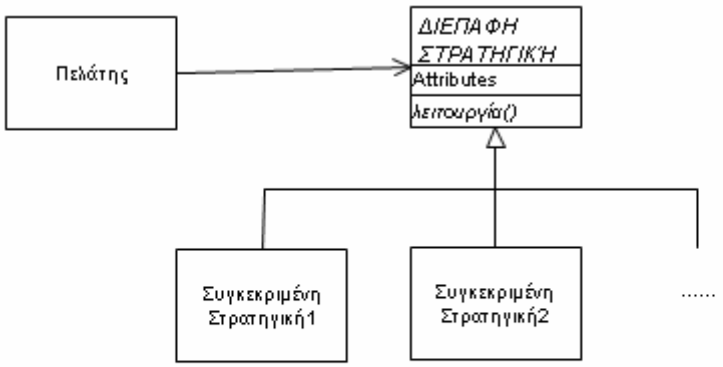
Υπάρχουν διάφορα πλεονεκτήματα για να γίνει αυτό. Κατ' αρχάς, εάν υπάρχουν διαφορετικές συμπεριφορές που πρέπει να εκτελέσει ένα αντικείμενο, είναι πολύ απλούστερο να παρακολουθηθούν εάν κάθε συμπεριφορά είναι μια χωριστή κλάση, και να μην υπάρχουν μέσα σε κάποια μέθοδο. Εάν χρειαστεί να προστεθεί, να αφαιρεθεί, ή να αλλαχτεί οποιαδήποτε από τις συμπεριφορές, είναι πολύ απλούστερο, δεδομένου ότι καθεμία βρίσκεται στην κλάση της. Κάθε τέτοια συμπεριφορά ή αλγόριθμος που ενθυλακώνεται στην κλάση της, λέγεται στρατηγική [53].

Όταν υπάρχουν αντικείμενα που είναι βασικά τα ίδια, και διαφέρουν μόνο στη συμπεριφορά τους, είναι μια καλή ιδέα να χρησιμοποιηθεί το πρότυπο στρατηγικής. Χρησιμοποιώντας τις στρατηγικές, μπορούν να μειωθούν αυτά τα διάφορα αντικείμενα σε μια κλάση που χρησιμοποιεί διάφορες στρατηγικές. Η χρήση των στρατηγικών παρέχει επίσης μια συμπαθητική εναλλακτική λύση ένα αντικείμενο να επιτύχει διαφορετικές συμπεριφορές. Όταν δημιουργείται υποκλάση ενός αντικείμενο για να αλλάξει τη συμπεριφορά του, η συμπεριφορά που εκτελεί είναι στατική. Εάν πρέπει να αλλάξει τι κάνει, θα έπρεπε να δημιουργηθεί ένα νέο στιγμιότυπο μιας διαφορετικής υποκλάσης και να αντικατασταθεί εκείνο το αντικείμενο με αυτό. Όμως με τις στρατηγικές, το μόνο που πρέπει να γίνει είναι να αλλάξει η στρατηγική του αντικειμένου, και θα αλλάξει αμέσως ο τρόπος με τον οποίο συμπεριφέρεται [53].

Μερικές επιπτώσεις του προτύπου αυτού είναι ότι η στρατηγική επιτρέπει να επιλεχτεί ένας από διάφορους αλγορίθμους δυναμικά. Αυτοί οι αλγόριθμοι μπορούν να αφορούν σε μια ιεραρχία κληρονομικότητας ή μπορούν να είναι ανεξάρτητοι εφ' όσον εφαρμόζουν μια κοινή διεπαφή. Επίσης, υπάρχει μεγάλη ευελιξία.

Οι στρατηγικές όμως δεν τα κρύβουν όλα. Ο κώδικας πελατών πρέπει να γνωρίζει ότι υπάρχουν διάφορες εναλλακτικές στρατηγικές και έχουν μερικά κριτήρια για την επιλογή μεταξύ τους. Αυτό μετατοπίζει μια αλγοριθμική απόφαση στον προγραμματιστή πελατών ή το χρήστη. Επίσης, ο αριθμός των αντικειμένων είναι μεγάλος [59].

Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου στρατηγικής [7] [53] [54] [55] [56] [57] [59].



Σχήμα 5.8.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου στρατηγικής.

Οι συμμετέχοντες στο προηγούμενο διάγραμμα κλάσεων είναι [7] [58]:
1). Πελάτης.
Μια κλάση στο ρόλο Πελάτη εκπροσωπεί μια λειτουργία σε μια διεπαφή. Αυτό το κάνει χωρίς να γνωρίζει την πραγματικής κλάση του αντικειμένου που εκπροσωπεί η λειτουργία ή πώς η κλάση εφαρμόζει τη λειτουργία. Η διεπαφή είναι κοινή σε όλους τους αλγορίθμους. Διατηρεί μια αναφορά στο αντικείμενο Στρατηγικής.
2). Στρατηγική.
Μια διεπαφή σε αυτόν τον ρόλο παρέχει έναν κοινό τρόπο να προσεγγιστούν οι διαδικασίες που ενθυλακώνονται από τις υποκλάσεις της.
3). ΣυγκεκριμένηΣτρατηγική1, ΣυγκεκριμένηΣτρατηγική2 κτλ.
Οι κλάσεις σε αυτόν τον ρόλο παρέχουν τις εναλλακτικές εκτελέσεις της λειτουργίας που εκπροσωπεί η κλάση πελατών.

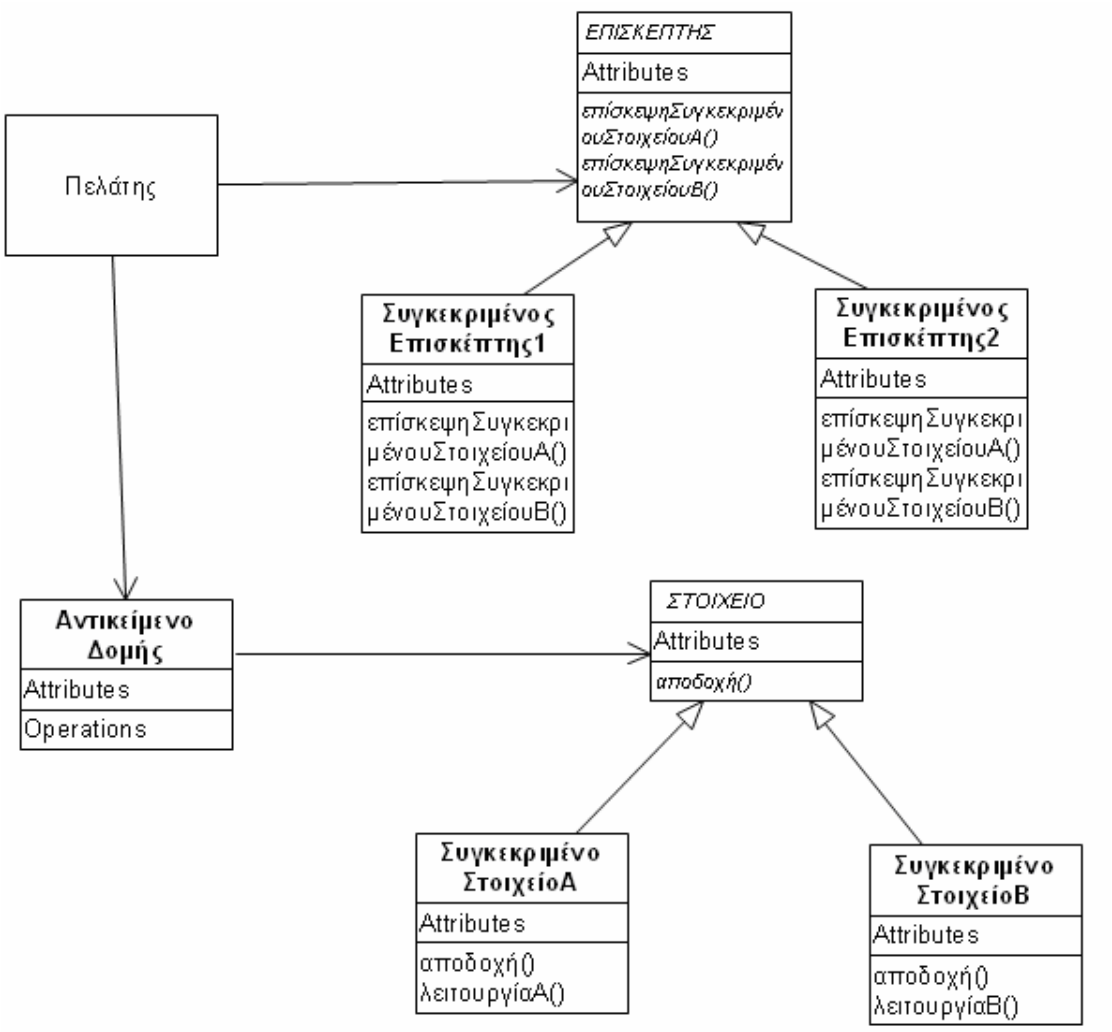
Στο έγγραφο ‘‘Using the Strategy Design Pattern to Compose Reliable Distributed Protocols ’’ παρουσιάζεται πώς τα αντικείμενα πρωτοκόλλου μπορούν να βοηθήσουν στην οικοδόμηση των αξιόπιστων διανεμημένων συστημάτων. Εστιάζεται στον τρόπο με τον οποίο το πρότυπο στρατηγικής επιτρέπει τους περιορισμούς της κληρονομικότητας να υπερνικηθούν, όταν γίνεται προσπάθεια να συντεθούν τα πρωτόκολλα [60].
Σύμφωνα με τον ορισμό στο έγγραφο **‘‘**Workshop on Adaptable and Adaptive Software**’’** [61] ένα πρόγραμμα λέγεται προσαρμοστικό (adaptive) εάν αλλάζει τη συμπεριφορά του αυτόματα σύμφωνα με το περιεχόμενό (πλαίσιο) του.
Στο έγγραφο με τίτλο ‘‘Adaptive Strategy Design Pattern’’ [62], προτείνεται ένα προσαρμοστικό σχεδιαστικό πρότυπο στρατηγικής που μπορεί να χρησιμοποιηθεί για να αναλύσει ή να σχεδιάσει τα μόνο-προσαρμοστικά συστήματα (self-adaptive systems). Παρουσιάζονται τα συστατικά που συμμετέχουν στη διαδικασία προσαρμογής, καθώς και μερικές από τις ιδιότητές τους.

5.9 Visitor pattern (Πρότυπο Επισκέπτη)

Στον αντικειμενοστρεφή προγραμματισμό και την τεχνολογία λογισμικού, το σχεδιαστικό πρότυπο επισκέπτη είναι ένας τρόπος διαχωρισμού ενός αλγόριθμο από

μια δομή αντικειμένου επάνω στην οποία λειτουργεί. Ένα πρακτικό αποτέλεσμα αυτού του χωρισμού, είναι η δυνατότητα να προστεθούν οι νέες διαδικασίες στις υπάρχουσες δομές αντικειμένου χωρίς τροποποίηση εκείνων των δομών. Κατά συνέπεια, η χρησιμοποίηση του προτύπου επισκέπτη βοηθά την προσαρμογή με την αρχή κλειστότητας/ανοιχτότητας (open/close principle) [63].
Η αρχή κλειστότητας/ανοιχτότητας δηλώνεται ως «οντότητες αρχής λογισμικού (κλάσεις, ενότητες, μέθοδοι, κ.λπ.) πρέπει να είναι ανοικτές για την επέκταση, αλλά κλειστές για τροποποίηση» [66] δηλαδή μια τέτοια οντότητα μπορεί να επιτρέψει να τροποποιηθεί η συμπεριφορά της, χωρίς αλλαγή του πηγαίου κώδικα της.
Ο επισκέπτης επιτρέπει να προστεθούν νέες εικονικές λειτουργίες σε μια οικογένεια των κλάσεων χωρίς τροποποίηση των κλάσεων. Δημιουργείται μια κλάση επισκέπτη που εφαρμόζει όλες τις κατάλληλες ειδικεύσεις της εικονικής λειτουργίας. Ο επισκέπτης παίρνει την αναφορά στιγμιότυπου ως εισαγωγή, και εφαρμόζει το στόχο μέσω της διπλής αποστολής.
Η διπλή αποστολή είναι ένας μηχανισμός που αποστέλλει μια κλήση λειτουργίας στις διαφορετικές συγκεκριμένες λειτουργίες ανάλογα με το χρόνο εκτέλεσης πολλών αντικειμένων που περιλαμβάνονται στην κλήση [67].
Παρόλο που το πρότυπο είναι ισχυρό, είναι πιο περιορισμένο από τις συμβατικές εικονικές λειτουργίες. Δεν είναι δυνατό να δημιουργηθούν οι επισκέπτες για τα αντικείμενα χωρίς προσθήκη μιας μικρής μεθόδου επανάκλησης μέσα σε κάθε κλάση και η μέθοδος επανάκλησης σε κάθε μια από τις κατηγορίες δεν είναι κληρονομική.

Το πρότυπο αυτό χρησιμοποιείται κυρίως στις ακόλουθες περιπτώσεις:
1). Όταν μια δομή αντικειμένου περιέχει πολλές κλάσεις αντικειμένων με διαφορετικές διεπαφές, και πρέπει να εκτελεστούν διαδικασίες σε αυτά τα αντικείμενα που εξαρτώνται από τις συγκεκριμένες κλάσεις τους.
2). Όταν οι κλάσεις που καθορίζουν τη δομή αλλάζουν σπάνια, αλλά πρέπει να καθορίζονται συχνά νέες διαδικασίες πέρα από τη δομή.
Στο επόμενο σχήμα φαίνεται το διάγραμμα κλάσεων του προτύπου επισκέπτη [63] [64] [65].



Σχήμα 5.9.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου επισκέπτη.

Παρακάτω εξηγούνται οι συμμετέχοντες στο πρότυπο αυτό.
1). Επισκέπτης.
Δηλώνει μια λειτουργία επίσκεψης για κάθε κλάση ΣυγκεκριμένοΣτοιχείο στη δομή αντικειμένου. Το όνομα και η υπογραφή της λειτουργίας προσδιορίζουν την κλάση που στέλνει το αίτημα επίσκεψης στον επισκέπτη. Αυτό αφήνει τον επισκέπτη να καθορίσει τη συγκεκριμένη κλάση του στοιχείου που επισκέπτεται. Κατόπιν ο επισκέπτης μπορεί να έχει πρόσβαση στα στοιχεία άμεσα μέσω της ιδιαίτερης διεπαφής του.

2). ΣυγκεκριμένοςΕπισκέπτης [71].
Εφαρμόζει κάθε λειτουργία που δηλώνεται από τον επισκέπτη. Κάθε λειτουργία εκτελεί ένα τεμάχιο του αλγορίθμου που καθορίζεται για την αντίστοιχη κλάση ή το αντικείμενο στη δομή. Ο ΣυγκεκριμένοςΕπισκέπτης παρέχει το πλαίσιο για τον αλγόριθμο και αποθηκεύει την κατάστασή του. Αυτή η κατάσταση περιέχει συχνά τα αποτελέσματα κατά τη διάρκεια του διάβασης/προσπέλασης της δομής.

3). Στοιχείο.
Καθορίζει μια μέθοδο αποδοχή, που παίρνει έναν επισκέπτη σαν όρισμα.

4). ΣυγκεκριμένοΣτοιχείο.
Εφαρμόζει μια μέθοδο αποδοχή, η οποία παίρνει έναν επισκέπτη σαν όρισμα.

5). ΑντικείμενοΔομής.
Μπορεί να απαριθμήσει τα στοιχεία του και να παρέχει μια υψηλού επιπέδου διεπαφή για να επιτρέψει στον επισκέπτη για να επισκεφτεί τα δεδομένα του. Μπορεί είτε να είναι μια Σύνθεση (πρότυπο) είτε μια συλλογή όπως ένας κατάλογος ή ένα σύνολο.

Δηλαδή το πρότυπο επισκέπτη είναι ένας τρόπος διαχωρισμού κάποιου αλγορίθμου, ο οποίος πιθανώς εκφράζει μία επέκταση στη λειτουργικότητα μίας ιεραρχίας κλάσεων, από τις εν λόγω κλάσεις. Στόχος είναι η σταδιακή προσθήκη νέων δυνατοτήτων σε όλες τις κλάσεις που υλοποιούν μία διασύνδεση Α χωρίς να χρειάζεται να τροποποιηθούν οι κλάσεις αυτές και χωρίς το εξωτερικό πρόγραμμα να γνωρίζει τους ακριβείς τύπους των αντικειμένων (μπορεί να χειρίζεται τα τελευταία με αφηρημένες αναφορές τύπου Α. Το μέσον για την επίτευξη του σκοπού αυτού είναι η ομαδοποίηση όλων των μεθόδων που περιγράφουν τη νέα λειτουργικότητα για κάθε κλάση της ιεραρχίας σε μία νέα, ξεχωριστή κλάση Επισκέπτη η οποία δεν κληρονομεί από την Α. Η Επισκέπτης περιέχει πολυμορφικές μεθόδους επίσκεψη, η καθεμία από τις οποίες δέχεται διαφορετικού τύπου όρισμα. Οι πιθανοί τύποι είναι οι παραγόμενες κλάσεις που υλοποιούν την Α. Κάθε μέθοδος επίσκεψη περιγράφει τη νέα λειτουργικότητα που αντιστοιχεί στην κλάση του ορίσματος της (π.χ. μία επεξεργασία των δεδομένων της εν λόγω κλάσης) αλλά οι μέθοδοι αυτές δεν καλούνται άμεσα από τον προγραμματιστή. Αντιθέτως το εξωτερικό πρόγραμμα, για κάθε αντικείμενο Β της ιεραρχίας στο οποίο θέλει να προσθέσει τη νέα λειτουργικότητα, καλεί μία μέθοδο αποδοχή του Β με όρισμα το αντικείμενο Επισκέπτης. Η αποδοχή πρέπει να προδιαγράφεται από την Α και να υλοποιείται σε κάθε παραγόμενη κλάση, ενώ το μόνον που κάνει είναι να καλεί με τη σειρά της τη μέθοδο επίσκεψη του ορίσματος της με όρισμα το αντικείμενο όπου περιέχεται. Έτσι τελικώς γίνεται η κατάλληλη επεξεργασία, με κλήση της κατάλληλης μεθόδου, χωρίς το εξωτερικό πρόγραμμα να γνωρίζει καν τον ακριβή παραγόμενο τύπο της εκάστοτε Β. Αν πολλές διαφορετικές επεκτάσεις είναι επιθυμητές, τότε μπορεί η κλάση Επισκέπτης να γίνει διασύνδεση και να υλοποιείται με διαφορετικές κλάσεις για κάθε ζητούμενη επέκταση.

Οι συνέπειες του προτύπου επισκέπτη είναι ότι η προσθήκη νέων διεργασιών γίνεται εύκολα. Αυτό συμβαίνει επειδή εάν η δομή περιλαμβάνει πολλές διαφορετικές κλάσεις, η πρόσθεση μιας νέας λειτουργίας στη δομή απαιτεί όλες εκείνες τις κλάσεις. Οι επισκέπτες συλλέγουν τις σχετικές διαδικασίες, χωρίζουν τις άσχετες. Όμως για να προστεθεί ένα ΣυγκεκριμένοΣτοιχείο πρέπει να αλλάξουν όλοι οι επισκέπτες που υπάρχουν. Άρα η προσθήκη των νέων κλάσεων ΣυγκεκριμένοΣτοιχείο είναι δύσκολη.
Η εργασία με τίτλο ‘‘A Pattern Language To Visitors ’’ [68] παρουσιάζει μια γλώσσα προτύπου για τους Επισκέπτες. Οι συντάκτες του ελπίζουν ότι η γλώσσα προτύπου μπορεί να βοηθήσει τον υπεύθυνο για την ανάπτυξη εφαρμογής να καταλάβει καλύτερα την περίπτωση που ένα πρότυπο επισκέπτη μπορεί να εφαρμοστεί και τα πλεονεκτήματα και μειονεκτήματα του, έτσι ώστε να μπορεί να ληφθεί μια σωστή απόφαση. Εντούτοις, αυτή η γλώσσα προτύπου δεν τελειώνει εδώ. Εφ' όσον συνεχίζουν να προκύπτουν νέα πρότυπα επισκεπτών, αυτή η γλώσσα προτύπου θα εξελιχθεί με αυτά.

Στο έγγραφο ‘‘The Essence of the Visitor Pattern’’ [70] επεξηγείται πώς να προγραμματίζονται οι επισκέπτες χωρίς να βασίζονται σε ορισμένες μεθόδους και χωρίς να γνωρίζουν όλες τις κλάσεις των αντικειμένων εκ των προτέρων. Στον αντικειμενοστρεφή σχεδιασμό, διάφορες τεχνικές ανάκλασης (reflection) υποστηρίζουν την πρόσβαση στα υπό-αντικείμενα, όπως φαίνονται και στην κλάση Walkabout της Java, η οποία χρησιμοποιείται στο έγγραφο.

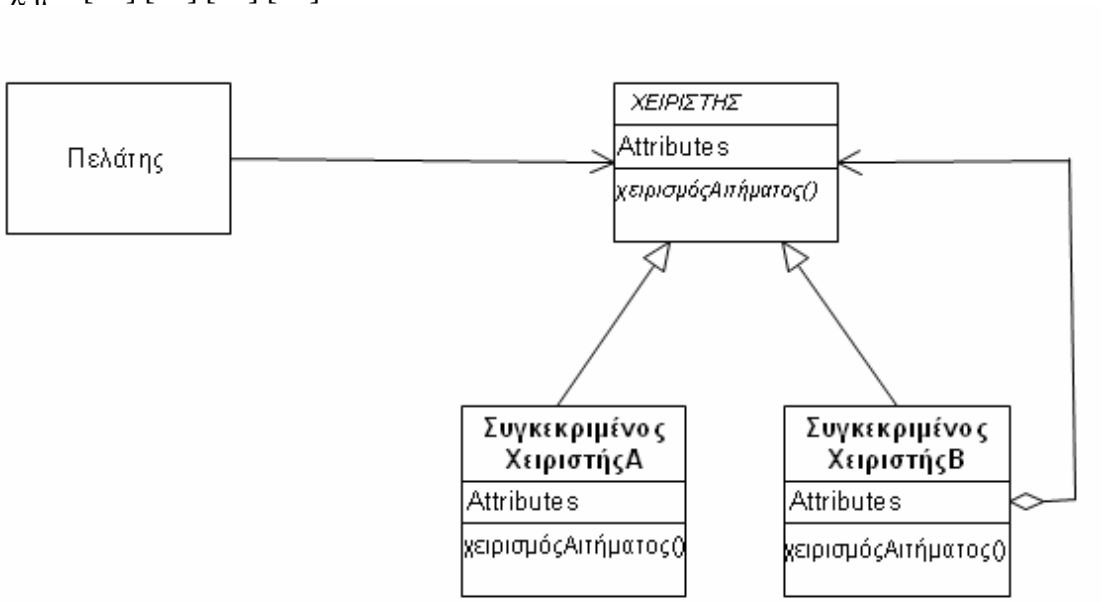
Ένα πρότυπο που σχετίζεται πολύ στενά με το πρότυπο του επισκέπτη είναι το Πρότυπο Single-Serving Visitor (Single-Serving Visitor pattern).
Το πρότυπο single-serving visitor πρέπει να χρησιμοποιηθεί όταν δεν πρέπει οι επισκέπτες (Visitors) να παραμείνουν στη μνήμη. Η πρόθεσή του είναι να βελτιστοποιήσει την εφαρμογή ενός επισκέπτη που διατίθεται, χρησιμοποιείται μόνο μια φορά, και μετά διαγράφεται. Αυτή είναι συχνά η περίπτωση που χρησιμοποιείται το πρότυπο Επισκέπτη (Visitor Pattern) μαζί με το πρότυπο Σύνθεσης (Composite Pattern) για να εκτελέσει έναν ενιαίο στόχο σε αυτό.
Το κανονικό πρότυπο Επισκέπτη (Visitor Pattern) πρέπει να χρησιμοποιηθεί όταν πρέπει ο επισκέπτης να παραμείνει στη μνήμη. Αυτό εμφανίζεται όταν διαμορφώνεται ο επισκέπτης με έναν αριθμό παραμέτρων, που πρέπει να κρατηθούν στη μνήμη για μια χρήση του επισκέπτη αργότερα [69].
Εντούτοις, εάν υπάρχει μόνο ένα στιγμιότυπο ενός τέτοιου επισκέπτη σε ένα ολόκληρο πρόγραμμα, μπορεί να είναι μια καλή ιδέα να εφαρμοστεί και ως single serving visitor και ως singleton. Με αυτό τον τρόπο, εξασφαλίζεται ότι το single serving visitor μπορεί να κληθεί αργότερα με τις παραμέτρους του αμετάβλητες .

Ένα άλλο πρότυπο το οποίο σχετίζεται και αυτό με το πρότυπο του επισκέπτη είναι το πρότυπο του ιεραρχικού επισκέπτη (Hierarchical Visitor).
Αντιπροσωπεύει μια λειτουργία που εκτελείται στους κόμβους μιας ιεραρχικής δομής αντικειμένου. Ο ιεραρχικός επισκέπτης (Hierarchical Visitor) αφήνει να καθοριστούν οι νέες διαδικασίες, χωρίς αλλαγή των κλάσεων των κόμβων στους οποίους λειτουργεί. Ο ιεραρχικός επισκέπτης υπερνικά τους περιορισμούς του παραδοσιακού προτύπου επισκέπτη (Visitor Pattern) επιτρέποντας σε έναν προγραμματιστή να ακολουθήσει προσπέλαση/διάσχιση σε βάθος. Δηλαδή, προτείνει ένα τρόπο, να επισκεφτεί κάθε κόμβος στην ιεραρχική δομή δεδομένων, όπως ένα δέντρο [75].

5.10 Πρότυπο Αλυσίδας Ευθύνης (Chain of Responsibility Pattern)

Στο αντικειμενοστρεφές σχεδιασμό, το πρότυπο αλυσίδας ευθύνης είναι ένα σχεδιαστικό πρότυπο που αποτελείται από μια πηγή αντικειμένων εντολής και μια σειρά αντικειμένων επεξεργασίας. Κάθε αντικείμενο επεξεργασίας περιέχει ένα σύνολο λογικής που περιγράφει τους τύπους αντικειμένων εντολής που μπορεί να χειριστεί, και πώς να απομακρύνει εκείνους που δεν μπορεί στο επόμενο αντικείμενο επεξεργασίας στην αλυσίδα. Ένας μηχανισμός υπάρχει επίσης για την προσθήκη των νέων αντικειμένων επεξεργασίας στο τέλος αυτής της αλυσίδας.
Σε μια παραλλαγή του τυποποιημένου προτύπου αλυσίδας ευθύνης, μερικοί χειριστές μπορούν να ενεργήσουν ως αποστολείς, ικανοί να στέλνουν εντολές σε διάφορες κατευθύνσεις, που διαμορφώνουν ένα δέντρο ευθύνης [72]. Φυσικά, το πρότυπο δεν πρέπει απαραίτητα να είναι γραμμικό, αφού μπορεί να δημιουργηθεί δέντρο.
Το πρότυπο επίσης, επιτρέπει σε διάφορες κλάσεις να χειριστούν ένα αίτημα, χωρίς να ξέρει καμία από αυτές για τις ικανότητες των άλλων κλάσεων. Παρέχει μια χαλαρή σύζευξη μεταξύ αυτών των κλάσεων. Η μόνη κοινή σύνδεση είναι το αίτημα

που περνούν μεταξύ τους. Το αίτημα το περνούν εμπρός, έως ότου μπορεί να το χειριστεί μια από τις κλάσεις.
Χρησιμοποιείται σε περιπτώσεις όπου, υπάρχουν περισσότεροι από ένας χειριστές που μπορούν να χειριστούν ένα αίτημα και δεν υπάρχει κανένας τρόπος να γίνει γνωστό ποιος χειριστής θα το χρησιμοποιήσει. Ο χειριστής πρέπει να καθοριστεί αυτόματα από την αλυσίδα. Επίσης, όταν πρέπει να διανεμηθεί ένα αίτημα σε ένα από διάφορα αντικείμενα χωρίς να διευκρινιστεί ρητά ποιό. Μια περίπτωση ακόμα είναι όταν ο πελάτης Θέλει να είναι σε θέση να τροποποιήσει δυναμικά το σύνολο των αντικειμένων, που μπορεί να χειριστεί τα αιτήματα [74] [76].
Το διάγραμμα κλάσεων του προτύπου αλυσίδας ευθύνης απεικονίζεται στο επόμενο σχήμα [73] [74] [77] [78].



Σχήμα 5.10.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου αλυσίδας ευθύνης.

Οι κλάσεις και τα αντικείμενα που συμμετέχουν σε αυτό το πρότυπο είναι:

1). Χειριστής.
Καθορίζει μια διεπαφή για το χειρισμό των αιτημάτων.

2). ΣυγκεκριμένοςΧειριστήςΑ, ΣυγκεκριμένοςΧειριστήςΒ κτλ.
Χειρίζεται αιτήματα για τα οποία είναι αρμόδιο. Μπορεί να έχει πρόσβαση στο διάδοχό του. Εάν ο ΣυγκεκριμένοςΧειριστής μπορεί να χειριστεί το αίτημα, το χειρίζεται, διαφορετικά διαβιβάζει το αίτημα στο διάδοχό του.

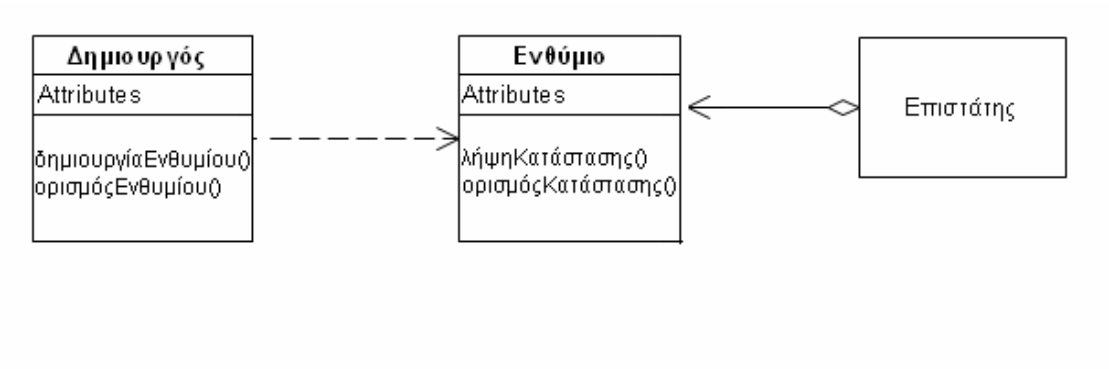
3). Πελάτης
Αρχίζει το αίτημα σε ένα αντικείμενο ΣυγκεκριμένοςΧειριστής στην αλυσίδα.

Οι συνέπειες του προτύπου αλυσίδας ευθύνης είναι οι εξής [74] [76]:
1). Ο κύριος σκοπός για αυτό το πρότυπο, όπως και σε διάφορα άλλα, είναι να μειωθεί η σύζευξη/σύνδεση μεταξύ των αντικειμένων. Ένα αντικείμενο πρέπει μόνο να ξέρει πώς να διαβιβάσει το αίτημα σε άλλα αντικείμενα.
2). Αυτή η προσέγγιση δίνει ευελιξία στη διανομή των ευθυνών μεταξύ των αντικειμένων. Οποιοδήποτε αντικείμενο μπορεί να ικανοποιήσει μερικά ή όλα τα

αιτήματα, και μπορεί να αλλάξει και την αλυσίδα και τις ευθύνες στο χρόνο εκτέλεσης (run time) [78].
3). Ένα πλεονέκτημα είναι ότι μπορεί να μην υπάρξει κανένα αντικείμενο που μπορεί να χειριστεί το αίτημα. Το τελευταίο αντικείμενο στην αλυσίδα μπορεί απλά να απορρίψει οποιαδήποτε αιτήματα που δεν μπορεί να χειριστεί.

5.11 Πρότυπο Ενθύμιου (Memento pattern)

Το πρότυπο ενθυμίου είναι ένα σχεδιαστικό πρότυπο λογισμικού που παρέχει τη δυνατότητα να αποκατασταθεί ένα αντικείμενο στην προηγούμενη κατάστασή του. Διατηρεί και εξωτερικεύει την εσωτερική κατάσταση ενός αντικειμένου, χωρίς παραβίαση της ενθυλάκωσης, έτσι ώστε η κατάσταση του αντικειμένου μπορεί να αποκατασταθεί αργότερα [83].
Το πρότυπο ενθυμίου είναι χρήσιμο όταν υπάρχει ένα αντικείμενο, που θα παίρνει ένα «στιγμιότυπο» έτσι ώστε, αργότερα να μπορεί να χρησιμοποιηθεί το στιγμιότυπο για να αποκατασταθεί η αρχική κατάστασή, όπως η λειτουργία αναίρεση (undo). Αλλά συγχρόνως, δεν ενδιαφέρουν οι λεπτομέρειες για το πώς αυτό εμφανίζεται ή πώς η κατάσταση αντιπροσωπεύεται εσωτερικά.
Δηλαδή, το πρότυπο ενθυμίου χρησιμοποιείται από δύο αντικείμενα: τον δημιουργό και τον επιστάτη. Ο δημιουργός είναι κάποιο αντικείμενο που έχει μια εσωτερική κατάσταση. Ο επιστάτης πρόκειται να κάνει κάτι στο δημιουργό, αλλά θέλει να είναι σε θέση να αναιρέσει την αλλαγή. Ο επιστάτης ρωτά αρχικά το δημιουργό για ένα αντικείμενο ενθύμιου. Κατόπιν κάνει οποιαδήποτε λειτουργία (ή ακολουθία διαδικασιών) επρόκειτο να κάνει. Για να πάει πίσω στην κατάσταση πριν από τις διαδικασίες, επιστρέφει το αντικείμενο ενθυμίου στο δημιουργό. Το ίδιο το αντικείμενο ενθυμίου είναι ένα αδιαφανές αντικείμενο, δηλαδή που ο επιστάτης δεν μπορεί, ή δεν πρέπει να αλλάξει. Όταν χρησιμοποιείται αυτό το πρότυπο, πρέπει να δοθεί προσοχή, εάν ο δημιουργός μπορεί να αλλάξει άλλα αντικείμενα ή πόρους [79].
Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου ενθυμίου [80] [84] [85].



Σχήμα 5.11.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου ενθυμίου.

Οι κλάσεις ή/και τα αντικείμενα που συμμετέχουν σε αυτό το πρότυπο είναι:

1). Ενθύμιο.
Αποθηκεύει την εσωτερική κατάσταση του αντικειμένου Δημιουργού. Το ενθύμιο μπορεί να αποθηκεύσει τόσο πολύ ή τόσο λίγα της εσωτερικής κατάστασης του Δημιουργού ανάλογα με τις ανάγκες κατά την κρίση του δημιουργού του. Επίσης,

προστατεύει από την πρόσβαση από τα αντικείμενα εκτός από το δημιουργό. Τα Ενθύμια έχει αποτελεσματικά δύο διεπαφές. Ο Επιστάτης βλέπει μια στενή διεπαφή στο Ενθύμιο. Μπορεί μόνο να περάσει το ενθύμιο στα άλλα αντικείμενα. Ο Δημιουργός, αντίθετα, βλέπει μια ευρεία διεπαφή, μια που την αφήνει να έχει πρόσβαση σε όλα τα στοιχεία απαραίτητα να αποκατασταθούν στην προηγούμενη κατάστασή τους.

2). Δημιουργός
Δημιουργεί ένα ενθύμιο που περιέχει ένα στιγμιότυπο της επικρατούσας εσωτερικής κατάστασής του. Επίσης χρησιμοποιεί το ενθύμιο για να αποκαταστήσει την εσωτερική κατάστασή του.

3). Επιστάτης
Είναι αρμόδιος για τη διαφύλαξη του ενθυμίου. Δεν λειτουργεί ποτέ επάνω ή εξετάζει το περιεχόμενο ενός ενθυμίου.

Μερικές συνέπειες του προτύπου ενθυμίου είναι ότι το ενθύμιο παρέχει έναν τρόπο να συντηρηθεί η κατάσταση ενός αντικειμένου συντηρώντας ενθυλάκωση, στις γλώσσες όπου αυτό είναι δυνατό. Κατά συνέπεια, στοιχεία που μόνο η κλάση δημιουργού πρέπει να έχει πρόσβαση παραμένουν ιδιωτικά. Συντηρεί επίσης την απλότητα της κλάσης δημιουργού, εξουσιοδοτώντας την αποταμίευση και την αποκατάσταση των πληροφοριών στην κλάση ενθυμίου.
Από την άλλη πλευρά, το ποσό πληροφοριών που πρέπει να σώσει/διατηρήσει ένα ενθύμιο μπορεί να είναι αρκετά μεγάλο, καταλαμβάνοντας κατά συνέπεια μεγάλα ποσά αποθήκευσης. Αυτό έχει μια επίδραση στην κλάση επιστάτη, η οποία πρέπει να σχεδιάσει στρατηγικές για να περιορίσει τον αριθμό αντικειμένων των οποίων σώζει την κατάσταση. Η συνεχής αποθήκευση και ανάκτηση καταστάσεων μπορεί να σπαταλά χρόνο άσκοπα.

5.12 Πρότυπο μεθόδου Template (Template Method Pattern)

Σε πολλές περιπτώσεις όπου υπάρχουν διάφορες κλάσεις που κληρονομούν από την ίδια υπέρ-κλάση, μερικές απ' αυτές εφαρμόζουν αλγορίθμους που είναι ίδιοι στη δομή. Δημιουργώντας τον κώδικα μέσα στον αλγόριθμο θα επιτρέψει να μετακινηθεί κάποιος κοινός κώδικας έξω από τις υποκλάσεις και στα μέσα στην υπέρ-κλάση. Η υπέρ-κλάση μπορεί μετά να είναι αρμόδια για το χειρισμό του αλγορίθμου και των κοινών μεθόδων ,ενώ οι υποκλάσεις μπορούν να εφαρμόσουν τις μεθόδους που είναι μοναδικές σε αυτές [86].
Το πρότυπο έχει οριστεί «Καθορίζει το σκελετό ενός αλγορίθμου σε μια λειτουργία, αναβάλλοντας μερικά βήματα στις υποκλάσεις. Η μέθοδος Template αφήνει τις υποκλάσεις να επαναπροσδιορίσουν ορισμένα βήματα ενός αλγορίθμου χωρίς να αλλάξουν τη δομή του αλγορίθμου». [87] [26].

Το πρότυπο αυτό χρησιμοποιείται για να εφαρμόσει τα αμετάβλητα μέρη ενός αλγορίθμου και να αφήσει τις υποκλάσεις να εφαρμόσουν τη διαφορετική συμπεριφορά. Επίσης, για να ελέγξει πώς μια υποκλάση μπορεί να επεκτείνει τις μεθόδους της υπέρ-κλάσης της [26]. Ακόμα, χρησιμοποιείται για να εντοπίσει την κοινή συμπεριφορά μεταξύ των υποκλάσεων και να την τοποθετήσει σε μια κοινή κλάση (σε αυτήν την περίπτωση, υπέρ-κλάση) για να αποφύγει το διπλασιασμό

κώδικα. Μπορεί να καθορίσει μια μέθοδο template που καλεί τις μεθόδους σε συγκεκριμένα σημεία, επιτρέποντας τις επεκτάσεις μόνο σε εκείνα τα σημεία [35].

Το διάγραμμα κλάσεων του προτύπου μεθόδου Template είναι [7] [86] [87] [26] [35] [36]:

Σχήμα 5.12.1 Απεικονίζεται το διάγραμμα κλάσεων του πρότυπο μεθόδου Template.

Οι κλάσεις ή/και τα αντικείμενα που συμμετέχουν στο διάγραμμα αυτό είναι [7] [91]:

1). ΑφηρημένηΚλάση.
Καθορίζει τις αφηρημένες αρχικές διαδικασίες που καθορίζουν οι συγκεκριμένες υποκλάσεις για να εφαρμόσουν τα βήματα ενός αλγορίθμου. Εφαρμόζει μια μέθοδο Template καθορίζοντας το σκελετό ενός αλγορίθμου. Η μέθοδος αυτή περιέχει την υψηλού επιπέδου λογική της κλάσης. Καλεί τις αρχικές διεργασίες καθώς επίσης και τις διεργασίες ορισμένες σε ΑφηρημένηΚλάση ή άλλων αντικειμένων. Έτσι επικαλείται τη χαμηλού επιπέδου λογική που ποικίλλει με κάθε υποκλάση της κλάσης ΑφηρημένηΚλάση.
2). ΣυγκεκριμένηΚλάση.
Μια κλάση σε αυτό το ρόλο είναι μια συγκεκριμένη υποκλάση μιας ΑφηρημένηΚλάσης. Εφαρμόζει τις αρχικές διεργασίες που πραγματοποιούν βήματα συγκεκριμένων υποκλάσεων του αλγορίθμου.

5.13 Πρότυπο Ειδίκευσης (Specification Pattern)

Το πρότυπο ειδίκευσης (specification pattern) είναι ένα ισχυρό σχεδιαστικό πρότυπο που μπορεί να χρησιμοποιηθεί για να αφαιρέσει άχρηστα από τη διεπαφή μιας κλάσης

μειώνοντας τη σύζευξη και αυξάνοντας την επεκτασιμότητα. Η αρχική του χρήση είναι να επιλέγει ένα υποσύνολο των αντικειμένων βασισμένο σε μερικά κριτήρια, και να ανανεώνει την επιλογή σε διάφορους χρόνους. Έτσι ορισμένα μόνο αντικείμενα επιλέγονται για ένα ρόλο και όχι όλα [82].

Ένα πρότυπο ειδίκευσης περιγράφει μια μονάδα της επιχειρησιακής λογικής που μπορεί να συνδυαστεί με τις μονάδες λογικής άλλων επιχειρήσεων. Σε αυτό το πρότυπο, μια μονάδα της επιχειρησιακής λογικής κληρονομεί τη λειτουργία της από την αφηρημένη σύνθετη κλάση ειδίκευσης (Composite Specification class). Η σύνθετη κλάση ειδίκευσης καλεί μια μέθοδο που επιστρέφει μια τιμή boolean. Η τιμή αυτή είναι αλήθεια (true) αν ένα συγκεκριμένο αντικείμενο πληρεί κάποια κριτήρια. Αν δεν τα πληροί, η τιμή γίνεται ψέματα (false). Μετά από τη δημιουργία, η ειδίκευση συνδέεται με άλλες ειδικεύσεις, καθιστώντας τις νέες ειδικεύσεις εύκολα συντηρήσιμες. Η επιχειρησιακή λογική μπορεί, μέσω της επίκλησης μεθόδου, να αλλάξει την κατάστασή της προκειμένου να γίνει εκπρόσωπος άλλων κλάσεων [81].

Με άλλα λόγια, μια ειδίκευση είναι μια κλάση που λέει πως πρέπει να μοιάσει ένα αντικείμενο, προκειμένου να χρησιμοποιηθεί, αντί για το πώς πρέπει να χρησιμοποιηθεί. Αυτό επιτρέπει στην ειδίκευση να χρησιμοποιηθεί σε διάφορα σενάρια, ενώ έχει οριστεί μόνο μια φορά στην εφαρμογή.

Αναφορές:

[1]: http://en.wikipedia.org/wiki/Behavioral_patterns
[2]: http://www.allapplabs.com/java_design_patterns/behavioral_patterns.htm
[3]:http://el.wikipedia.org/wiki/%CE%A3%CF%87%CE%B5%CE%B4%CE%B9%CE%B1%CF%83%CF%84%CE%B9%CE%BA%CE%AC_%CF%80%CF%81%CF%8C%CF%84%CF%85%CF%80%CE%B1
[4]: http://en.wikipedia.org/wiki/Command_pattern
[5]: Joey Lott, Danny Patterson, (2006), *ActionScript 3 Design Pattern Chapter 7 Command Pattern*, pp. 247 Publisher: Adobe Press.
[6]: Gamma,E.,Helm,R., Johnson,R. and Vlissades, J. (1995).*Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA.
[7]: Grand, M. (2002) *Patterns in Java: A Catalog of Reusable Design Patterns Illustrated with UML, 2nd edn, Vol. 1*. John Wiley & Sons,New York.
[8]: Functions and the Command Pattern by Bob Tarr Design Patterns in Java
[9]: Joey Lott, Danny Patternson, (2006), Advanced ActionScript 3 with Design Patterns , publisher Adobe Press 2-11-2006.
[10]: Andreas Naderlinger, (2007), *Applicability of the Command design pattern for Undo/Redo support An alternative: Undo via MVC*, Software Research Group.
[11]: Mark J. Sebern, (2000), *Slides: Command Design Pattern 4/3/00* SE281-S00-10.
[12]: David L. Levine and Christopher D. Gill, (1997), *Command Pattern and Combinations*, CS 342: Object-Oriented Software Development Lab.
[13]: Carl Alphonce, Michael E. Caspersen, Adrienne Decker, (2007), *Killer "killer examples" for design patterns*. SIGCSE 2007, pp.228-232.
[14]: Judith Bishop, (2008), *C# 3.0 Design Patterns*, Published by O'Reilly Media 2008.
[15]: http://en.wikipedia.org/wiki/Template_method_pattern
[16]: Brian d foy, (2002) *The Iterator Design Pattern* , The Perl Review 2002.
[17]: Bob Tarr, (2000), *The Iterator Pattern Design Patterns In Java.*
[18]: Niamanth Sridhar, (2008), Slides *Iterator Pattern*. Fall 2008, Software Engineering ,Cleveland State University.
[19]: J.S. Sventek, (2004), Slides: *Elements of Reusable Object-Oriented Software*, Ch11 Design Patterns 2004, University of Glasgow.
[20]: Erich Gama, Richard Helm, Ralph Johnson, John Vlissides, (1995), *Design patterns: elements of reusable object-oriented software*, Addison-Wesley Professional Computing Series 1995.
[21]: http://en.wikipedia.org/wiki/Interpreter_pattern
[22]: http://en.wikipedia.org/wiki/Abstract_syntax_tree

[23]: http://home.earthlink.net/~huston2/dp/interpreter.html
[24]: Jan Vrany and Alexandre Bergel, (2007), *THE DEBUGGABLE INTERPRETER DESIGN PATTERN*, ICSOFT (PL/DPS/KE/MUSE) 2007, pp.22-29.
[25]: DAVID H. LORENZ,(1997), *Tiling Design Patterns -A Case Study Using the Interpreter Pattern*, OOPSLA 1997 pp.206-217.
[26]: Mel Ó Cinnéide, (2005), Slides: *The Template Method Pattern*, Department of Computer Science, University College Dublin , Apr-05 Software Engineering MSc: Design Patterns.
[27]: http://sourcemaking.com/design_patterns/interpreter
[28]:http://www.cs.mcgill.ca/~hv/classes/CS400/01.hchen/doc/interpreter/interpreter.html#Design%20Patterns
[29]: http://www.dofactory.com/Patterns/PatternInterpreter.aspx
[30]: http://en.wikipedia.org/wiki/Mediator_pattern
[31]: http://www.dofactory.com/Patterns/PatternMediator.aspx
[32]: http://www.allapplabs.com/java_design_patterns/mediator_pattern.htm
[33]: http://www.vincehuston.org/dp/mediator.html#similar
[34]: Bains K., Lau E. (2002). *Mediator Design Pattern*. University of Calgary http://sern.ucalgary.ca/courses/SENG/443/W02/assignments/Mediator/.
[35]: Bob Tarr, (2000), *The Template Method Pattern Design Patterns In Java.*
[36]: Woei-Kae Chaen, Slides: *Object- Oriented Programming Template Method Pattern*, CSIE Department, NTUT.
[37]: http://msdn.microsoft.com/en-us/library/ms954621.aspx
[38]: Bob Tarr, (2000), *The Observer Pattern Design Patterns In Java.*
[39]: Aspect Oriented Development Of P2P File Sharing Application by Eren Algan, Dogan Kaya Berktas and Ridvan Tekdogan
[40]: Slides:, *Some Design Patterns*, March 6, 2003, CMSC 433 – Programming Language Technologies and Paradigms Spring 2003.
[41]: Mel Ó Cinnéide, (2005), Slides: *The Observer Pattern*, Department of Computer Science, University College Dublin.
[42]: Dominik Gruntz, (2002), *Java Design: On the Observer Pattern*, University of Applied Sciences, Aargau.
[43]: Jacob Borella, (2003), *The Observer Pattern Using Aspect Oriented Programming*, IT-University of Copenhagen VikingPLOP September 19-22 2003.
[44]: Ross Harmes and Dustin Diaz, (2007), *Pro JavaScript™ Design Patterns*, published by Apress.
[45]: http://en.wikipedia.org/wiki/State_pattern
[46]: http://exciton.cs.rice.edu/JavaResources/DesignPatterns/StatePat.htm
[47]: http://www.dofactory.com/Patterns/PatternState.aspx
[48]: Bob Tarr, (2000), *The State and Strategy Patterns Design Patterns In Java.*
[49]: Mark Mahoney, (2007), *A Pattern Based Approach to Aspect-Orientation for State Based Systems*, Aspect-oriented software development; Vol. 211, Proceedings of the 2nd workshop on Best practices in applying aspect-oriented software development Vancouver, British Columbia, Canada, Article No. 5, 2007.
[50]: Luciane Lamour Ferreira, Cecilia M. F. Rubira, (1998), *Reflective Design Patterns to Implement Fault Tolerance*, Proceedings of OOPSLA'98 Workshop on Reflective Programming in C++ and Java Vancouver, Canada - Sunday, October 18th, 1998.
[51]: Anatoly Shalyto, Nikita Shamgunov and Georgy Korneev, *State Machine Design Pattern*, UNION Agency – Science Press, Plzen, Czech Republic.
[52]:http://el.wikipedia.org/wiki/%CE%A3%CF%87%CE%B5%CE%B4%CE%B9%CE%B1%CF%83%CF%84%CE%B9%CE%BA%CE%AC_%CF%80%CF%81%CF%8C%CF%84%CF%85%CF%80%CE%B1
[53]: http://www.exciton.cs.rice.edu/JavaResources/DesignPatterns/StrategyPattern.htm
[54]: http://en.wikipedia.org/wiki/Strategy_pattern
[55]: http://www.dofactory.com/Patterns/PatternStrategy.aspx
[56] William Sanders and Chandima Cumaranatunge, (2006), *ActionScript 3.0 Design Patterns*, published by O'REILLY.
[57]: Απόστολος Ζάρρας, *Πρότυπα Συμπεριφοράς (Behavioral Patterns)* http://www.cs.uoi.gr/~zarras/
[58]: Eric Freeman & Elisabeth Freeman With Kathy Sierra & Bert Bates, (2004), *Strategy Pattern Head First Design Pattern* O'Reilly, First Ed. Oct 2004.
[59]: Woei-Kae Chaen, Slides: *Object-Oriented Programming Strategy Pattern*, CSIE Department, NTUT.

[60]: Benoˆıt Garbinato and Rachid Guerraoui, (1997), *Using the Strategy Design Pattern to Compose Reliable Distributed Protocols*, Conference on Object-Oriented Technology and Systems Proceedings of the 3rd conference on USENIX Conference on Object-Oriented Technologies (COOTS) - Volume 3, Portland, Oregon 1997, pp.17-17.
[61]: Cristina Videira Lopes and Karl Lieberherr, (1996), *AP/S++: Case-study of a MOP for purposes of software evolution*. In proceedings of Reflection'96, S. Francisco, USA, April 1996.
[62]: Olivier Aubert and Antoine Beugnard, (2001), *Adaptive Strategy Design Pattern*, Koalaplop 2001.
[63]: http://en.wikipedia.org/wiki/Visitor_pattern
[64]: http://www.dofactory.com/Patterns/PatternVisitor.aspx
[65]: http://www.exciton.cs.rice.edu/javaresources/DesignPatterns/VisitorPattern.htm
[66]: Meyer, Bertrand (1988). *Object-Oriented Software Construction*. Prentice Hall
[67]: http://en.wikipedia.org/wiki/Double_dispatch
[68]: Yun Mai and Michel de Champlain, (2001), *A Pattern Language To Visitors*, EuroPLoP™ 2001.
[69]: http://en.wikipedia.org/wiki/Single-serving_visitor_pattern
[70]: Jens Palsberg ,W Lafayette and C. Barry Jay, (1998), *The Essence of the Visitor Pattern*, Computer Software and Applications Conference, 1998. COMPSAC apos;98. Proceedings. The Twenty-Second Annual International, 19-21 Aug 1998 pp.9 – 15.
[71]: Toni Sellares, Slides: *The Visitor Pattern* ,Universitat de Girona
[72]: http://en.wikipedia.org/wiki/Chain-of-responsibility_pattern
[73]: http://www.dofactory.com/Patterns/PatternChain.aspx
[74]: Alexandre Denault, (2004), *Chain of Responsibility Comp-303 : Programming Techniques Lecture 21*, Computer Science, McGill University, Winter 2004.
[75]: http://en.wikipedia.org/wiki/Hierarchical_visitor_pattern
[76]: Bob Tarr, (2000), *The Chain of ResponsibilityPattern Design Patterns In Java*.
[77]: Jim Whitehead, (2009), *Slides: Decorator and Chain of Responsibility Patterns*, *Game Design Experience*, UC Santa Cruz.
[78]: Markus L. Dechert, (2005), Slides: *Design Patterns - Chain of Responsibility*, Department of Computer Science Technische Universitat Darmstadt, Jan 17th 2005.
[79]: http://en.wikipedia.org/wiki/Memento_pattern
[80]: http://www.dofactory.com/Patterns/PatternMemento.aspx
[81]: http://en.wikipedia.org/wiki/Specification_pattern
[82]: Eric Evans and Martin Fowler *Specifications*, The 4th Pattern Languages of Programming Conference Washington University Technical Report 97-34.
[83]: David L. Levine and Christopher D. Gill, Slides: *The Memento Pattern*, CS 342: Object-Oriented Software Development Lab, Department of Computer Science, Washington University, St. Louis
[84]: Raman Ramsin, Richard F. Paige, (2008), Process-centered review of object oriented software development methodologies. ACM Comput. Surv. 40(1): (2008)
[85]: Howard Cheng, *Design Patterns*, CS 2720 Practical Software Development, University of Lethbridge.
[86]: http://gpwiki.org/index.php/Template_method_pattern
[87]: http://www.dofactory.com/Patterns/PatternTemplate.aspx

6. ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ ΔΙΑΜΕΡΙΣΗΣ
Partitioning Design Patterns

6.1 Σύντομη αναφορά των Partitioning προτύπων

Μια κοινή στρατηγική επίλυσης προβλήματος είναι η διαίρει και βασίλευε. Περιλαμβάνει τη διαίρεση ενός σύνθετου προβλήματος που είναι δύσκολο να λυθεί σε απλούστερα προβλήματα που είναι ευκολότερα να λυθούν. Τα πρότυπα σε αυτήν την κατηγορία παρέχουν τις οδηγίες σχετικά με το πώς να χωρίσουν τις κλάσεις και τις διεπαφές έτσι ώστε να είναι ευκολότερο να φθάσουν σε ένα καλό πρότυπο [6].
Το πρότυπο φίλτρου (filter pattern) περιγράφει πώς να οργανώσει υπολογισμούς σε ροή δεδομένων με έναν εύκαμπτο τρόπο που να επιτρέπει να αναμιχθούν και να ταιριάσουν διαφορετικοί υπολογισμοί στο ίδιο ρεύμα δεδομένων. Είναι μια εξειδικευμένη έκδοση του προτύπου διακοσμητή (decorator pattern) που εστιάζει στο χειρισμό ενός ρεύματος δεδομένων [3].
Το πρότυπο read only interface περιγράφει πώς να χωριστούν οι κλάσεις που χρησιμοποιούν ένα αντικείμενο έτσι ώστε εκείνες που πρέπει να επιτρέπεται να το τροποποιούν αυτό μπορούν να το τροποποιούν και εκείνες που δεν πρέπει να το τροποποιούν, δεν μπορούν να το τροποποιούν.

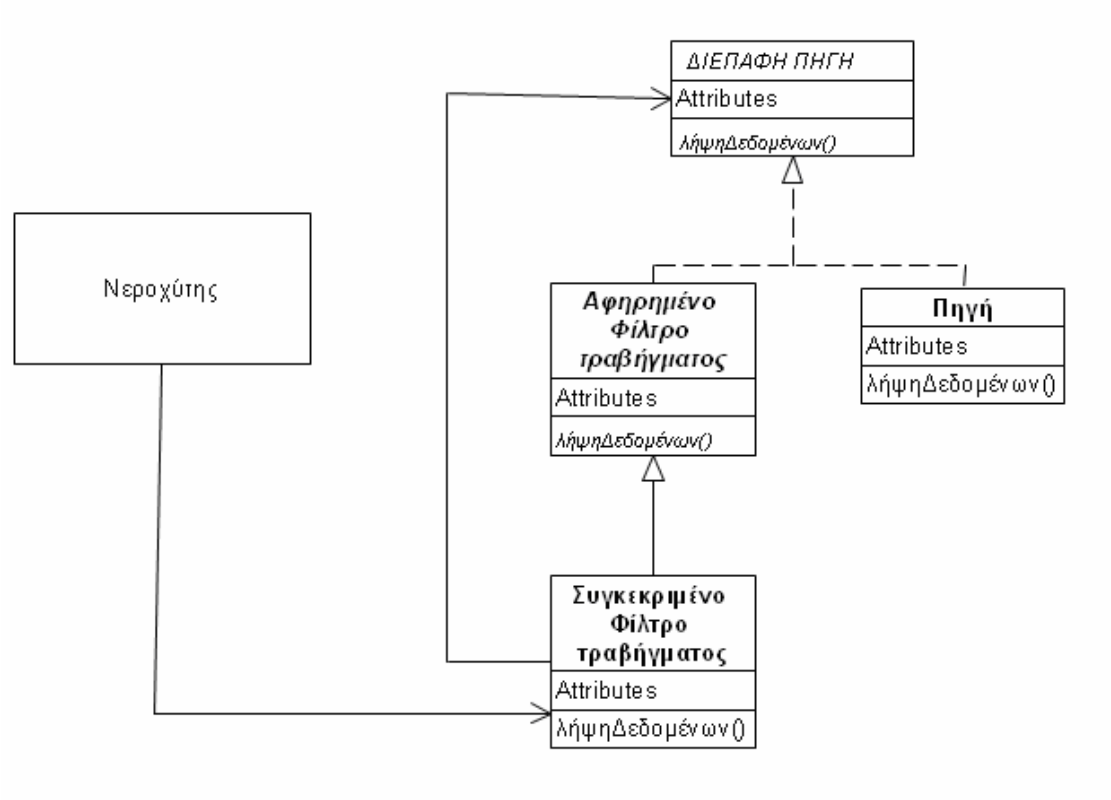
6.2 Πρότυπο Φίλτρου (Filter pattern)

Τα αντικείμενα που έχουν συμβατές διεπαφές αλλά εκτελούν διαφορετικούς μετασχηματισμούς και υπολογισμούς στα ρεύματα στοιχείων μπορούν να συνδεθούν δυναμικά για να εκτελέσουν αυθαίρετες διαδικασίες.
Ο σκοπός πολλών προγραμμάτων είναι να εκτελέσουν υπολογισμούς σε ένα ρεύμα δεδομένων. Δεδομένου ότι πολλά προγράμματα εκτελούν μετασχηματισμούς και αναλύσεις στα ρεύματα δεδομένων, θα ήταν ευεργετικό να καθοριστούν κλάσεις που εκτελούν τους πιο κοινούς μετασχηματισμούς και αναλύσεις. Τέτοιες κλάσεις θα επαναχρησιμοποιούνταν πολύ.
Οι κλάσεις που εκτελούν απλούς μετασχηματισμούς και αναλύσεις στα ρεύματα δεδομένων τείνουν να είναι πολύ γενικής φύσης. Όταν δημιουργούνται τέτοιες κλάσεις δεν μπορούν να είναι γνωστοί όλοι οι τρόποι που θα χρησιμοποιηθούν. Μερικές εφαρμογές θα θέλουν να εφαρμόσουν μετασχηματισμούς και αναλύσεις μόνο σε επιλεγμένα μέρη ενός ρεύματος δεδομένων. Σαφώς, αυτές οι κλάσεις πρέπει να γραφτούν με τέτοιο τρόπο, ώστε να επιτρέπεται μεγάλη ευελιξία στο πώς τα στιγμιότυπα τους μπορούν να συνδεθούν. Ένας τρόπος για να ολοκληρώσει αυτήν την ευελιξία πρόκειται να καθορίσει μια κοινή διεπαφή για όλες αυτές τις κλάσεις, έτσι ένα στιγμιότυπο, να μπορεί να χρησιμοποιήσει στιγμιότυπο μιας άλλης, χωρίς να πρέπει να ληφθεί υπόψη ποιας κλάσεις είναι το αντικείμενο [2].

Το πρότυπο φίλτρου οργανώνει τις κλάσεις που συμμετέχουν σε αυτό ως πηγές δεδομένων, νεροχύτες δεδομένων και φίλτρα δεδομένων. Τα δεδομένα κλάσεων φίλτρων εκτελούν τις διαδικασίες μετασχηματισμού και ανάλυσης. Υπάρχουν δύο βασικές μορφές του προτύπου φίλτρου. Στην πρώτη μορφή, τα δεδομένα ρέουν ως συνέπεια από ένα νεροχύτη δεδομένων που καλεί μια μέθοδο σε ένα αντικείμενο πηγής δεδομένων. Στη δεύτερη, δεδομένα ρέουν όταν περνά ένα αντικείμενο πηγής δεδομένων τα δεδομένα σε μια μέθοδο ενός αντικειμένου νεροχύτη δεδομένων.

Το σχήμα 6.2.1 είναι ένα διάγραμμα κλάσεων για τη μορφή φίλτρου, όπου τα αντικείμενα νεροχυτών δεδομένων παίρνουν τα δεδομένα καλώντας μεθόδους στις

πηγές δεδομένων. Αυτή η μορφή φίλτρου καλείται φίλτρο τραβήγματος (pull filter) [1].



Σχήμα 6.2.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου φίλτρου, όπου το φίλτρο ονομάζεται φίλτρο τραβήγματος.

Εδώ είναι οι περιγραφές για το πώς συμμετέχουν οι κλάσεις στο σχήμα 6.2.1 στη μορφή τραβήγματος του προτύπου φίλτρου.

1). Διεπαφή Πηγή.
Μια διεπαφή σε αυτόν τον ρόλο δηλώνει μια ή περισσότερες μεθόδους που επιστρέφουν δεδομένα όταν καλείται. Στο σχήμα 6.2.1, μια τέτοια μέθοδος είναι η λήψηΔεδομένων.

2). Πηγή.
Μια κλάση σε αυτόν τον ρόλο είναι αρμόδια πρώτα για την παροχή των δεδομένων παρά για να μετασχηματίζουν ή να αναλύουν τα δεδομένα. Οι κλάσεις σε αυτόν τον ρόλο απαιτούνται επίσης για να εφαρμόσουν τη διεπαφή Πηγή.

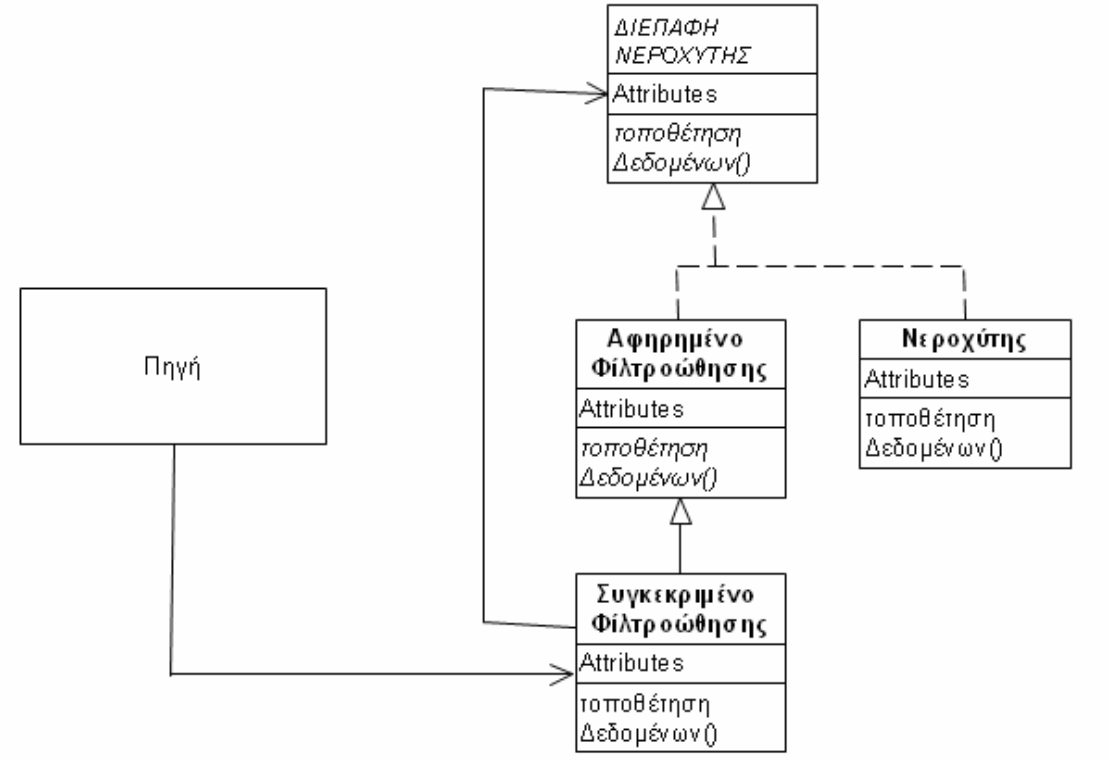
3). Αφηρημένο Φίλτρο τραβήγματος.
Μια κλάση σε αυτόν τον ρόλο είναι μια αφηρημένη υπέρ-κλάση από κλάσεις που μετασχηματίζουν και αναλύουν τα δεδομένα. Τα αντικείμενα αυτής της κλάσης εκπροσωπούν την προσκόμιση των δεδομένων στο αντικείμενο της διεπαφής Πηγή. Επειδή οι υποκλάσεις αυτής της κλάσης κληρονομούν το γεγονός ότι εφαρμόζει τη διεπαφή Πηγή, τα στιγμιότυπα τους μπορούν να παρέχουν τα δεδομένα σε άλλα αντικείμενα

Οι κλάσεις Αφηρημένο Φίλτρο τραβήγματος έχουν χαρακτηριστικά ένα μεταβλητό στιγμιότυπο που τίθεται από τον κατασκευαστή τους και αναφέρεται στο αντικείμενο Διεπαφή Πηγής που περνούν στον κατασκευαστή τους. Οι κλάσεις Αφηρημένο Φίλτρο τραβήγματος καθορίζουν χαρακτηριστικά μια μέθοδο λήψηΔεδομένων που καλεί απλά τη μέθοδο λήψηΔεδομένων του αντικειμένου Διεπαφή Πηγής που αναφέρεται από το μεταβλητό στιγμιότυπο.

4). Συγκεκριμένο Φίλτρο τραβήγματος.
Οι κλάσεις σε αυτόν τον ρόλο είναι μια συγκεκριμένη υποκλάση μιας κλάσης Αφηρημένο Φίλτρο τραβήγματος. Αγνοούν τη μέθοδο λήψηΔεδομένων που κληρονομούν για να εκτελέσουν τις κατάλληλες διαδικασίες μετασχηματισμού ή ανάλυσης.

5). Νεροχύτης.
Στιγμιότυπα κλάσεων σε αυτόν τον ρόλο καλούν τη μέθοδο λήψηΔεδομένων ενός αντικειμένου Διεπαφή Πηγής. Αντίθετα από τα αντικείμενα Συγκεκριμένο Φίλτρο τραβήγματος, τα στιγμιότυπα κλάσεων Νεροχύτη, χρησιμοποιούν τα δεδομένα χωρίς μεταφορά του ενός προς ένα άλλο αντικείμενο Αφηρημένο Φίλτρο τραβήγματος.

Το σχήμα 6.2.2 είναι ένα διάγραμμα κλάσεων για τη μορφή φίλτρου όπου τα αντικείμενα πηγής δεδομένων περνούν τα δεδομένα σε μεθόδους αντικειμένων δεδομένων νεροχύτη. Αυτή η μορφή φίλτρου καλείται φίλτρο ώθησης (push filter) [1].



Σχήμα 6.2.2 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου φίλτρου, όπου το φίλτρο ονομάζεται φίλτρο ώθησης.
Εδώ είναι οι περιγραφές για το πώς συμμετέχουν οι κλάσεις στο σχήμα 6.2.2 στη μορφή ώθησης του προτύπου φίλτρου.

1). Διεπαφή Νεροχύτη.
Μια διεπαφή σε αυτόν τον ρόλο δηλώνει μια ή περισσότερες μεθόδους που παίρνουν τα δεδομένα μέσω μιας από τις παραμέτρους της. Στο σχήμα 6.2.2 μια τέτοια μέθοδος είναι η τοποθέτησηΔεδομένων.

2). Νεροχύτης.
Μια κλάση σε αυτόν τον ρόλο είναι αρμόδια για τη λήψη και την επεξεργασία των δεδομένων, παρά να μετασχηματίζουν ή να αναλύουν τα δεδομένα. Οι κλάσεις σε αυτόν τον ρόλο απαιτούνται επίσης για να εφαρμόσουν τη διεπαφή Νεροχύτης. Τα δεδομένα περνούν στα αντικείμενα νεροχύτη περνώντας τα δεδομένα στη μέθοδο τοποθέτησηΔεδομένων του αντικειμένου Νεροχύτη.

3). Αφηρημένο Φίλτρο ώθησης.
Μια κλάση σε αυτόν τον ρόλο είναι μια αφηρημένη υπέρ-κλάση των κλάσεων που μετασχηματίζουν και αναλύουν τα δεδομένα. Έχει έναν κατασκευαστή που παίρνει ένα όρισμα (argument) που είναι ένα αντικείμενο Διεπαφή Νεροχύτη. Στιγμιότυπα αυτής της κλάσης περνούν τα δεδομένα στο αντικείμενο Διεπαφής Νεροχύτη που είχαν περάσει στον κατασκευαστή τους. Επειδή οι υποκλάσεις αυτής της κλάσης κληρονομούν το γεγονός ότι εφαρμόζει τη διεπαφή Νεροχύτη, τα στιγμιότυπα τους μπορούν να δεχτούν τα δεδομένα από άλλα αντικείμενα που περνούν τα δεδομένα στα αντικείμενα Διεπαφής Νεροχύτη.

Οι κλάσεις Αφηρημένο Φίλτρο ώθησης έχουν χαρακτηριστικά ένα μεταβλητό στιγμιότυπο που τίθεται από τον κατασκευαστή τους και αναφέρεται στο αντικείμενο Διεπαφής Νεροχύτη που περνούν στον κατασκευαστή τους. Οι κλάσεις Αφηρημένο Φίλτρο ώθησης καθορίζουν μια μέθοδο τοποθέτησηΔεδομένων που καλεί τη μέθοδο τοποθέτησηΔεδομένων του αντικειμένου Διεπαφής νεροχύτη που αναφέρεται από το μεταβλητό στιγμιότυπο.

4). Συγκεκριμένο Φίλτρο ώθησης.
Οι κλάσεις σε αυτόν τον ρόλο είναι μια συγκεκριμένη υποκλάση μιας κλάσης Αφηρημένο Φίλτρο ώθησης. Αγνοούν τη μέθοδο τοποθέτησηΔεδομένων που κληρονομούν για να εκτελέσουν τις κατάλληλες διαδικασίες μετασχηματισμού ή ανάλυσης.

5). Πηγή.
Τα στιγμιότυπα των κλάσεων σε αυτόν τον ρόλο καλούν τη μέθοδο τοποθέτησηΔεδομένων ενός αντικειμένου Διεπαφής Νεροχύτη.

6.3 Πρότυπο Διεπαφής μόνο ανάγνωσης (Read-Only Interface pattern)

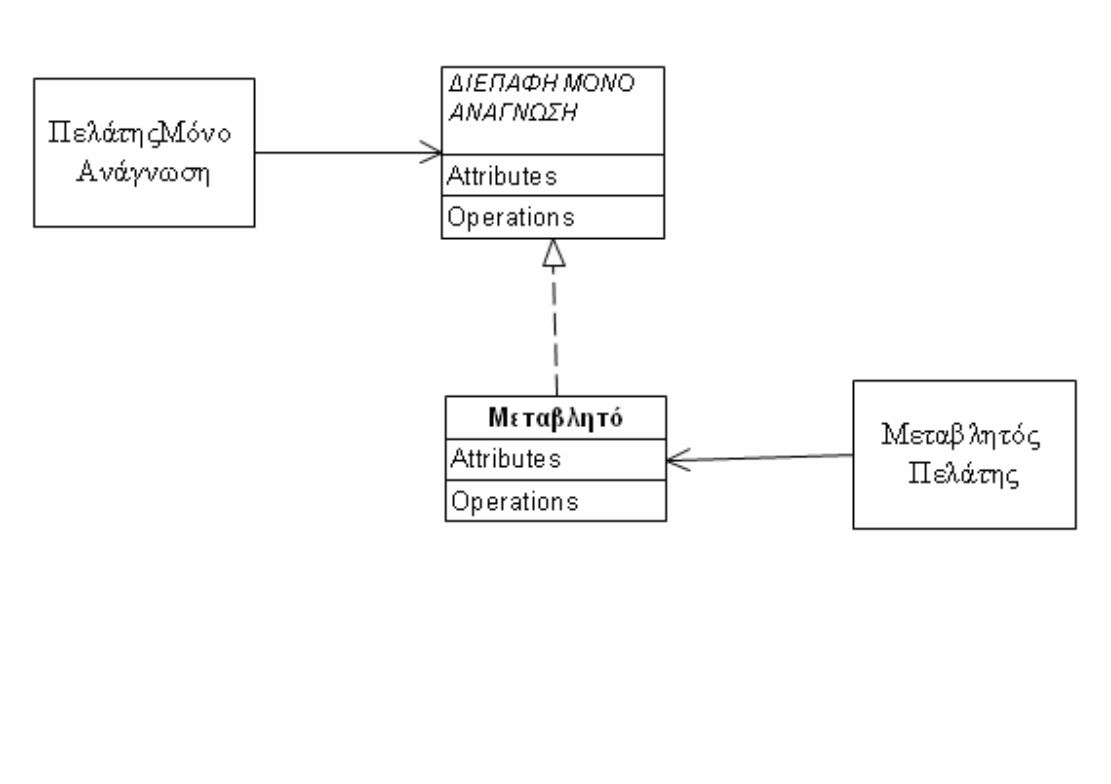
Ένα αντικείμενο πρέπει να τροποποιείται από ορισμένους από τους πελάτες του και όχι από άλλους. Οι πελάτες που δεν χρειάζεται να τροποποιήσουν ένα αντικείμενο δεν το τροποποιούν με καταναγκασμό, για να έχουν πρόσβαση στο αντικείμενο μέσω μιας διεπαφής που δεν περιλαμβάνει μεθόδους που τροποποιούν το αντικείμενο.

Υπάρχει μια κλάση της οποίας τα στιγμιότυπα, πρέπει να τροποποιηθούν από ορισμένες κλάσεις και όχι από άλλες κλάσεις. Πρέπει μέσω μιας διεπαφής που καθορίζεται να αναγκαστούν οι πελάτες μιας κλάσης να έχουν πρόσβαση στην

κλάση. Όμως μπορεί να μην είναι πρακτικό σε διάφορες καταστάσεις. Οι κλάσεις πελατών μπορεί να είναι λογισμικό τρίτων (third party software) ή μπορεί υπάρχει μια κατάσταση συντήρησης, όπου δεν είναι επιθυμητό να αλλάξει η διεπαφή που χρησιμοποιούν οι κλάσεις πελατών.

Αυτό το πρότυπο είναι χρήσιμο, επειδή προστατεύει από λάθη προγραμματισμού. Εντούτοις, δεν είναι χρήσιμο στην προστασία από τις κακόβουλες πρακτικές προγραμματισμού [5].

Παρέχεται τη (read- only) πρόσβαση μόνο ανάγνωσης σε ένα μεταβλητό αντικείμενο απαιτώντας αντικείμενα να προσεγγίσουν το μεταβλητό αντικείμενο μέσω μιας διεπαφής, που δεν περιλαμβάνει μεθόδους για το αντικείμενο. Αυτή η οργάνωση παρουσιάζεται στο σχήμα 6.3.1 [1] [4].



Σχήμα 6.3.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου διεπαφής μόνο ανάγνωσης.

Εδώ είναι οι περιγραφές των ρόλων που παίζουν οι κλάσεις και οι διεπαφές στο πρότυπο διεπαφών μόνο ανάγνωσης [1] [4].

1). Μεταβλητό (Mutable).
Μια κλάση σε αυτόν τον ρόλο έχει μεθόδους που παίρνουν και βάζουν τις τιμές των ιδιοτήτων της. Εφαρμόζει επίσης τη διεπαφή ΜόνοΑνάγνωση.

2). Διεπαφή ΜόνοΑνάγνωση.
Μια διεπαφή σε αυτόν τον ρόλο έχει τις ίδιες μεθόδους λήψης (get methods), όπως η Μεταβλητή κλάση, η οποία εφαρμόζει τη διεπαφή. Εντούτοις, αυτή η διεπαφή δεν περιλαμβάνει μεθόδους που θα ανάγκαζαν ένα Μεταβλητό αντικείμενο να τροποποιήσει το περιεχόμενό του.

3). ΜεταβλητόςΠελάτης.
Οι κλάσεις σε αυτόν τον ρόλο χρησιμοποιούν τη Μεταβλητή κλάση άμεσα και μπορούν να καλέσουν τις μεθόδους που τροποποιούν την κατάσταση ενός Μεταβλητού αντικειμένου.

4). ΠελάτηςΜόνοΑνάγνωση.
Οι κλάσεις σε αυτόν τον ρόλο έχουν πρόσβαση στη μεταβλητή κλάση μέσω της διεπαφής ΜόνοΑνάγνωση. Οι κλάσεις που έχουν πρόσβαση στη Μεταβλητή κλάση μέσω της διεπαφής ΜόνοΑνάγνωση δεν είναι ικανές να έχουν πρόσβαση στις μεθόδους που τροποποιούν τα Μεταβλητά αντικείμενα.

Μερικές συνέπειες του προτύπου αυτού είναι ότι χρησιμοποιώντας το πρότυπο διεπαφής μόνο ανάγνωσης, αποφεύγονται λάθη προγραμματισμού, τα οποία θα μπορούσαν να έχουν ως αποτέλεσμα ορισμένες κλάσεις να μπορούν να τροποποιήσουν αντικείμενα που δεν θα έπρεπε να τροποποιήσουν. Επίσης το πρότυπο αυτό δεν αποτρέπει τα αντικείμενα από το να τροποποιηθούν εσφαλμένα, ως αποτέλεσμα κακόβουλου προγραμματισμού.

Το πρότυπο Διεπαφής μόνο ανάγνωσης δεν πρέπει να μπερδεύεται με το Αμετάβλητο πρότυπο (Immutable Pattern). Η διαφορά μεταξύ τους είναι σημαντική. Το Αμετάβλητο αντιμετωπίζει ζητήματα σχετικά με συγχρονισμό για τα αντικείμενα η τιμή των οποίων αξία αλλά όχι η ταυτότητα είναι σημαντική. Η διεπαφή μόνο ανάγνωσης διευθύνει χωρισμό μιας κλάσης διεπαφής σε χωριστές διεπαφές βασισμένες στις παρενέργειες μεθόδων. Όχι μόνο είναι διαφορετική η πρόθεσή τους, αλλά και το κίνητρο και η δυνατότητα εφαρμογής τους [7].

Αναφορές:

[1]: Mark Grand, (2002), *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML, Second Edition*, John Wiley and Sons 2002.
[2]: Mark Grand and Brad Merrill, (2005), *Visual Basic design Patterns*, Wiley Publishing.
[3]: http://www.mindspring.com/~mgrand/pattern_synopses.htm#Filter
[4]: Timothy Lethbridge, Robert Laganiere, (2001), *Object-Oriented Software Engineering, Practical Software Development using UML and Java Chapter 6: Using Design Patterns*, Mc Graw Hill.
[5]: Dhamayanthi N., Thangavel P., (2006),*Structural Design Patterns and .NET Framework 2.0*, Journal of Object Technology Published by ETH Zurich, Chair of Software Engineering Vol.5, No.8, November-December 2006.
[6]: Gamma, Erich, et. al. (AKA, The Gang of Four), (1994), *Design Patterns: Elements of Reusable Object – Oriented Software*. Upper Saddle River, NJ: Addison-Wesley, 1994
[7]: http://www.bcs-spa.org/resources/BCSOOPSNL/Issue34Summer1998/Articles/Henney.html

7. ΠΡΟΤΥΠΑ ΤΑΥΤΟΧΡΟΝΙΣΜΟΥ
Concurrency Patterns

7.1 Σύντομη αναφορά των προτύπων ταυτοχρονισμού (συγχρονισμού).

Αυτά τα πρότυπα παρουσιάζουν γενικές λύσεις σε συχνά προβλήματα συγχρονισμού που βρίσκονται σε ταυτόχρονα ή και διανεμημένα συστήματα. Εστιάζουν σε δύο είδη προβλημάτων, στη διανομή των πόρων, εστιάζοντας στη διαχείριση αδιεξόδου (deadlock management), και την ταυτόχρονη εκτέλεση των διαδικασιών. Αυτές οι διαδικασίες πρέπει να ακολουθήσουν μια σωστή σειρά. Για παράδειγμα, η εισαγωγή ενός στοιχείου δεδομένων σε μια δομή δεδομένων πρέπει να συμβεί πριν από την αφαίρεσή του [1].

Η επιλογή της αρχιτεκτονικής συγχρονισμού έχει έναν σημαντικό αντίκτυπο στο σχεδιασμό και στην απόδοση του πολύπλοκων πολυνηματικών εφαρμογών. Καμία αρχιτεκτονική συγχρονισμού δεν είναι κατάλληλη για όλες τις περιπτώσεις και όλες τις πλατφόρμες υλικού και λογισμικού. Στην τεχνολογία λογισμικού, τα πρότυπα αυτά είναι τύποι σχεδιαστικών προτύπων που εξετάζουν το παράδειγμα πολυνηματικού προγραμματισμού. Τα παραδείγματα αυτής της κατηγορίας προτύπων περιλαμβάνουν [3]:

Ενεργό Αντικείμενο (Active Object)
Πρότυπο Εμποδισμού (Balking Pattern)
Πρότυπο Διπλοελεγμένου κλειδώματος (Double checked locking Pattern)
Φρουρούμενη Αναβολή-αναστολή (Guarded suspension)
Πρότυπο Ηγετών/ακολούθων (Leaders/Followers Pattern)
Αντικείμενο Οργάνου Ελέγχου (Monitor Object)
Πρότυπο Ανάγνωσης-γραφής κλειδαριάς- κλειδώματος (Read write lock Pattern)
Πρότυπο Χρονοπρογραμματιστή (Scheduler Pattern)
Πρότυπο Αποθέματος Νημάτων (Thread pool Pattern)
Πρότυπο Αποθήκευσης Συγκεκριμένου-Νήματος (Thread-Specific Storage Pattern)
Πρότυπο Αντιδραστήρα (Reactor Pattern)

Δύο πρότυπα που διευκρινίζουν σχεδιασμούς για τη διανομή των πόρων μεταξύ των πολλών νημάτων ή των διαδικασιών είναι το Ενεργό αντικείμενο (Active Object) και το Αντικείμενο οργάνου ελέγχου (Monitor Object). Το σχεδιαστικό πρότυπο ενεργού αντικειμένου αποσυνδέει την εκτέλεση μεθόδου από την επίκληση μεθόδου. Ο σκοπός του είναι να ενισχύσει το συγχρονισμό και να απλοποιήσει τη συγχρονισμένη πρόσβαση στα αντικείμενα που βρίσκονται στα νήματα ελέγχου τους. Το πρότυπο αντικείμενου οργάνου ελέγχου (Monitor Object) συγχρονίζει την ταυτόχρονη εκτέλεση μεθόδου για να εξασφαλίσει ότι μόνο μια μέθοδος τη φορά τρέχει μέσα σε ένα αντικείμενο. Επιτρέπει επίσης στις μεθόδους ενός αντικειμένου να σχεδιάσουν μαζί- από κοινού την ακολουθία εκτέλεσής τους. Και τα δύο πρότυπα μπορούν να συγχρονίσουν και να σχεδιάσουν τις μεθόδους που επικαλούνται ταυτόχρονα στα αντικείμενα. Η κύρια διαφορά είναι ότι ένα ενεργό αντικείμενο εκτελεί τις μεθόδους του σε ένα διαφορετικό νήμα από τους πελάτες του, ενώ ένα αντικείμενο οργάνου ελέγχου εκτελεί τις μεθόδους του δανείζοντας το νήμα των πελατών του. Κατά συνέπεια τα ενεργά αντικείμενα μπορούν να εκτελέσουν πιο περίπλοκο, αν και ακριβό, σχεδιασμό για να καθορίσει τη σειρά με την οποία οι μέθοδοί τους εκτελούνται.

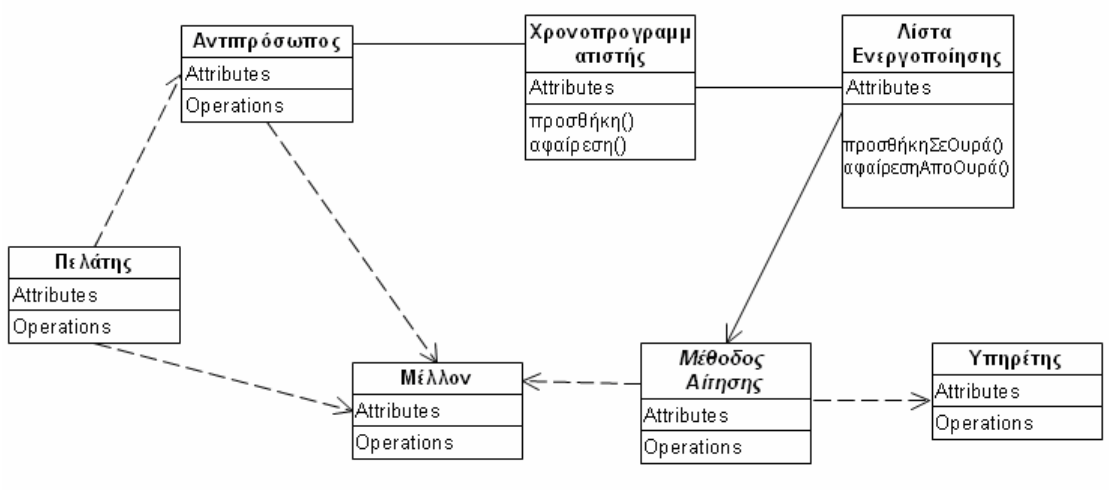
Το πρότυπο ηγετών/ακολούθων (Leaders/Followers Pattern) παρέχει ένα αποδοτικό μοντέλο συγχρονισμού, όπου πολλά νήματα αναλαμβάνουν με τη σειρά να μοιραστούν ένα σύνολο πηγών γεγονότος που ανιχνεύουν, αποπολυπλέκουν, αποστέλλουν, και επεξεργάζονται τα αιτήματα υπηρεσιών που εμφανίζονται στις

πηγές γεγονότος. Το πρότυπο αυτό μπορεί να χρησιμοποιηθεί για να βελτιώσει την απόδοση όταν δεν υπάρχει κανένας συγχρονισμός ή σύνολο περιορισμών στην επεξεργασία των αιτημάτων από τα συγκεντρωμένα νήματα.
Το Πρότυπο Αποθήκευσης Συγκεκριμένου-Νήματος (Thread-Specific Storage Pattern) προσφέρει μια διαφορετική στρατηγική για ορισμένες έμφυτες περιπλοκές του συγχρονισμού. Το συγκεκριμένο σχεδιαστικό πρότυπο επιτρέπει σε πολλά νήματα να χρησιμοποιήσουν ένα σημείο πρόσβασης 'λογικά σφαιρικό' για να ανακτήσουν ένα αντικείμενο που είναι τοπικό σε ένα νήμα, χωρίς να επιβαρυνθεί με έξοδα κλειδώματος σε κάθε πρόσβαση στο αντικείμενο. Ως ένα ορισμένο βαθμό αυτό το πρότυπο μπορεί να εμφανιστεί ως ''αντίθεση'' άλλων προτύπων, επειδή εξετάζει διάφορες περιπλοκές του συγχρονισμού, εμποδίζοντας τη διανομή των πόρων μεταξύ των νημάτων. Το πρότυπο αντιδραστήρα (Reactor Pattern) αποπολυπλέκει και αποστέλλει τα αιτήματα που παραδίδονται ταυτόχρονα σε μια εφαρμογή από έναν ή περισσότερους πελάτες [2].
Το Πρότυπο Αποθέματος Νημάτων (Thread pool Pattern) χρησιμοποιείται όταν προκύπτουν καταστάσεις σχεδιασμού, που θα μπορούσαν να ωφεληθούν με τη χρησιμοποίηση πολλών νημάτων μικρής διάρκειας (short lived) νήματα [21]. Αντί να δημιουργηθεί ένα νέο νήμα για κάθε εργασία, μπορεί να χρησιμοποιηθεί ένα από τα νήματα από το απόθεμα νημάτων και να οριστεί για την εργασία. Όταν το νήμα τελειώσει με την εργασία , προστίθεται πίσω στο απόθεμα και περιμένει μια άλλη ανάθεση.
Το πρότυπο διπλοελεγμένου κλειδώματος μειώνει τα έξοδα σύγκρουσης και συγχρονισμού όποτε κρίσιμα τμήματα του κώδικα χρειάζονται να αποκτήσουν κλειδαριά μια φορά, αλλά πρέπει να είναι νήμα-ασφαλές, όταν αποκτούν τη κλειδαριά [36].

7.2 Πρότυπο ενεργού αντικειμένου (Active Object Pattern)

Το πρότυπο ενεργού αντικειμένου, αποσυνδέει την εκτέλεση μεθόδου από την επίκληση μεθόδου προκειμένου να απλοποιήσει τη συγχρονισμένη πρόσβαση σε ένα αντικείμενο που βρίσκεται στο νήμα ελέγχου. Επίσης επιτρέπει σε ένα ή περισσότερα ανεξάρτητα νήματα της εκτέλεσης να προσπαθήσουν να έχουν πρόσβαση σε δεδομένα, ως ένα ενιαίο αντικείμενο. Αυτό το πρότυπο χρησιμοποιείται συνήθως στα διανεμημένα συστήματα που απαιτούν πολύπλοκους κεντρικούς υπολογιστές. Επιπλέον, σε εφαρμογές πελατών, όπως οι μηχανές αναζήτησης δικτύων, χρησιμοποιούν τα ενεργά αντικείμενα για να απλοποιήσουν τις ταυτόχρονες, ασύγχρονες διαδικασίες δικτύων.[37]
Το πρότυπο χρησιμοποιείται στα πλαίσια των αρχιτεκτονικών πελάτη-εξυπηρετητή (κεντρικό υπολογιστή), όταν ο χρόνος επεξεργασίας αιτήματος του εξυπηρετητή είναι αρκετά μεγάλος, και επομένως ο χρήστης μπορεί να χρησιμοποιήσει αυτό το χρόνο για να εκτελέσει άλλες στοιχειώδεις εργασίες ασύγχρονα με τον κεντρικό υπολογιστή. Στην ασύγχρονη επεξεργασία, ένας χρήστης ζητά τις υπηρεσίες ενός κεντρικού υπολογιστή εκδίδοντας ασύγχρονα αιτήματα. Ο κεντρικός υπολογιστής ενημερώνει το χρήστη πότε η υπηρεσία είναι πλήρης. Γράφοντας τον κώδικα για την εφαρμογή των ασύγχρονων αιτημάτων, κάποιος πρέπει να εξετάσει έναν μεγάλο αριθμό χαμηλού επιπέδου λεπτομερειών. Αυτό καθιστά τον κώδικα δυσνόητο και πιο δύσκολα επαναχρησιμοποιήσιμο και μπορεί να οδηγήσει σε κρυμμένα σφάλματα προγραμματισμού [31].

Το διάγραμμα κλάσεων του προτύπου ενεργού αντικειμένου απεικονίζεται στο επόμενο σχήμα [2] [4].



Σχήμα 7.2.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου ενεργού αντικειμένου.

Οι κλάσεις ή και τα αντικείμενα που συμμετέχουν στο διάγραμμα κλάσεων είναι:

1). Αντιπρόσωπος.
Ένας αντιπρόσωπος παρέχει μια διεπαφή που επιτρέπει στον πελάτη να επικαλεστεί δημόσια τις προσιτές μεθόδους σε ένα ενεργό αντικείμενο χρησιμοποιώντας τα τυποποιημένα, χαρακτηριστικά γνωρίσματα γλώσσας προγραμματισμού, αντί τη διαβίβαση μηνυμάτων μεταξύ των νημάτων. Όταν ένας πελάτης επικαλείται μια μέθοδο που ορίζεται από τον αντιπρόσωπο, αυτό προκαλεί την κατασκευή και την τοποθέτηση σε ουρά-σειρά ενός αντικειμένου ΜέθοδοςΑίτησης επάνω στη ΛίσταΕνεργοποίησης του Χρονοπρογραμματιστή.

2) ΜέθοδοςΑίτησης.
Χρησιμοποιείται για να περάσει τις πληροφορίες για μια συγκεκριμένη επίκληση μεθόδου σε έναν αντιπρόσωπο, όπως οι παράμετροι μεθόδου και ο κώδικας, από τον αντιπρόσωπο σε έναν Χρονοπρογραμματιστή που (τρέχει) σε ένα χωριστό νήμα. Μια αφηρημένη κλάση ΜέθοδοςΑίτησης καθορίζει μια διεπαφή για την εκτέλεση των μεθόδων ενός ενεργού αντικειμένου. Η διεπαφή περιέχει επίσης μεθόδους (φρουράς) που μπορούν να χρησιμοποιηθούν για να καθορίσουν πότε ισχύουν οι περιορισμοί συγχρονισμού μιας ΜεθόδουΑίτησης. Για κάθε μέθοδο Ενεργού Αντικειμένου που προσφέρεται από τον αντιπρόσωπο που απαιτεί τη συγχρονισμένη πρόσβαση στον Υπηρέτη του, η αφηρημένη κλάση ΜεθόδοςΑίτησης δημιουργείται σαν υποκλάση, για να δημιουργήσει μια συγκεκριμένη κλάση ΜεθόδοςΑίτησης. Στιγμιότυπα αυτών των κλάσεων δημιουργούνται από τον αντιπρόσωπο όταν καλούνται οι μέθοδοί της και περιέχουν τις συγκεκριμένες πληροφορίες πλαισίου απαραίτητες να εκτελέσουν αυτές τις επικλήσεις μεθόδων και να επιστρέψουν οποιαδήποτε αποτελέσματα πίσω στους πελάτες.

3). ΛίσταΕνεργοποίησης.
Διατηρεί τις εκκρεμής ΜεθόδουςΑίτησης που δημιουργούνται από τον Αντιπρόσωπο. Αυτή η σειρά αναμονής κρατά τη διαδρομή της ΜεθόδουΑίτησης που ζητά

να εκτελέσει. Αποσυνδέει επίσης το νήμα πελάτη από το νήμα υπαλλήλου, έτσι τα δύο νήματα μπορούν να τρέξουν ταυτόχρονα.

4). Χρονοπρογραμματιστής.
Ένας Χρονοπρογραμματιστής τρέχει σε ένα διαφορετικό νήμα από τους πελάτες του, διαχειρίζοντας μια ΛίσταΕνεργοποίησης των ΜεθόδωνΑίτησης, που εκκρεμή η εκτέλεση. Ένας Χρονοπρογραμματιστής αποφασίζει ποιά ΜέθοδοςΑίτησης να αφαιρέσει από την ουρά μετά και εκτελεί τον Υπηρέτη που εφαρμόζει αυτήν την μέθοδο. Αυτή η απόφαση βασίζεται σε διάφορα κριτήρια, όπως η σειρά με την οποία οι μέθοδοι παρεμβάλλονται στη ΛίσταΕνεργοποίησης και τους περιορισμούς συγχρονισμού.. Ένας χρονοπρογραμματιστής αξιολογεί τους περιορισμούς συγχρονισμού χρησιμοποιώντας μεθόδους φρουράς.

5). Υπηρέτης
Ένας Υπηρέτης καθορίζει τη συμπεριφορά και την κατάσταση που διαμορφώνεται ως ενεργό αντικείμενο. Οι υπηρέτες εφαρμόζουν τις μεθόδους που καθορίζονται από τον αντιπρόσωπο και τις αντίστοιχες ΜεθόδουςΑίτησης. Μια μέθοδος υπηρέτη καλείται όταν εκτελείται η αντίστοιχη ΜέθοδοςΑίτησης από έναν Χρονοπρογραμματιστή. Επομένως, οι υπηρέτες εκτελούν στο νήμα ελέγχου του Χρονοπρογραμματιστή.

6). Μέλλον.
Το Μέλλον επιτρέπει σε έναν πελάτη να επιτύχει τα αποτελέσματα των κλήσεων μεθόδων αφού τελειώσει ο Υπηρέτης να εκτελεί τη μέθοδο. Όταν ένας πελάτης επικαλείται μεθόδους μέσω ενός αντιπροσώπου, το Μέλλον επιστρέφεται αμέσως στον πελάτη. Το Μέλλον κρατά χώρο για να αποθηκεύσει τα αποτελέσματά της η μέθοδος που έχει καλεστεί.

Τα πλεονεκτήματα του προτύπου ενεργού αντικειμένου είναι τα εξής:
1). Ένα ενεργό αντικείμενο χωρίζει την επίκληση μεθόδου από τη μέθοδο που καλεί, και επιτρέπει να χρησιμοποιηθούν διαφορετικές στρατηγικές για την επίκληση μεθόδου χωρίς να πρέπει να τροποποιηθεί η κλήση του κώδικα [32].
2). Ενισχύει τον συγχρονισμό εφαρμογής και απλοποιεί την πολυπλοκότητα συγχρονισμού. Ο συγχρονισμός ενισχύεται, αφήνοντας τα νήματα πελατών και τις ασύγχρονες μεθόδους εκτέλεσης να τρέχουν ταυτόχρονα. Η πολυπλοκότητα συγχρονισμού απλοποιείται από το χρονοπρογραμματιστή, ο οποίος αξιολογεί τους περιορισμούς συγχρονισμού για να εγγυηθεί την σε σειρά πρόσβαση στους υπαλλήλους, που εξαρτώνται από την κατάστασή τους.
3). Η σειρά εκτέλεσης μεθόδου μπορεί να διαφέρει από τη σειρά επίκλησης μεθόδου. Οι μέθοδοι που επικαλούνται ασύγχρονα, εκτελούνται με βάση τους περιορισμούς συγχρονισμού τους, οι οποίοι μπορούν να διαφέρουν από τη σειρά κλήσης τους.

Ορισμένα μειονεκτήματα του ενεργού αντικειμένου είναι τα εξής:
1). Το κύριο μειονέκτημα είναι ότι όλες οι αλληλεπιδράσεις με ένα ενεργό αντικείμενο συμβαίνουν ασύγχρονα, και αυτό το ύφος της κλήσης μεθόδου δεν είναι το πιο φυσικό. Οποιαδήποτε μέθοδος που επιστρέφει μια τιμή, δεν μπορεί να χρησιμοποιηθεί με το ενεργό σχέδιο αντικειμένου. Αντ' αυτού, ο κώδικας πρέπει να γραφτεί σε ένα ασύγχρονο ύφος, όπου οι ενεργές μέθοδοι παίρνουν τα αντικείμενα επανάκλησης ως παραμέτρους και ειδοποιούν τους επισκέπτες τους πότε η εκτέλεση έχει τελειώσει. Ενώ αυτό το ύφος του προγραμματισμού δεν είναι φυσικό, συνήθως

δεν χρειάζεται πολλή προσπάθεια να μετατραπεί μια αλληλεπίδραση, που γράφεται σύγχρονη, να είναι ασύγχρονη [32].
2). Ένα άλλο μειονέκτημα των ενεργών αντικειμένων είναι ότι εισάγουν ένα επίπεδο πολυπλοκότητας που καθιστά τη διόρθωση δύσκολη. Η χρησιμοποίηση ενός διορθωτή μέσω μιας κλήσης σε μια ενεργό μέθοδο δεν είναι τρομερά χρήσιμη, δεδομένου ότι η κλήση επιστρέφει αμέσως και η πραγματική εργασία της μεθόδου πραγματοποιείται σε ένα άλλο πλαίσιο εκτέλεσης. Επιπλέον, πολλοί διορθωτές (debuggers) δεν υποστηρίζουν επαρκώς τις ταυτόχρονες εφαρμογές.

7.3 Πρότυπο εμποδισμού (Balking Pattern)

Εάν καλείται μια από τις πολλές μεθόδους ενός αντικειμένου, όταν δεν είναι το αντικείμενο σε μια κατάλληλη κατάσταση για να εκτελέσει τη μέθοδο, η μέθοδος return, δεν κάνει τίποτα [33]. Δηλαδή λέμε ότι η μέθοδος εμποδίζεται (balking).
Για παράδειγμα, έστω ένα πρόγραμμα για να ελέγχει το ηλεκτρονικό καζανάκι τουαλετών [33]. Τέτοιες συσκευές προορίζονται για τις δημόσιες τουαλέτες. Έχουν έναν αισθητήρα φωτός και όταν ο αισθητήρας ανιχνεύει ένα αυξημένο επίπεδο φωτός, υποθέτει ότι ένας άνθρωπος έχει αφήσει την τουαλέτα και προκαλεί μια εκροή. Ηλεκτρονικά καζανάκια τουαλετών έχουν επίσης ένα κουμπί για να προκαλέσουν με το χέρι μια εκροή. Η flush μέθοδος αρχίζει μια εκροή και επιστρέφει (return) μόλις αρχίσει η εκροή. Θα πρέπει να αποφασιστεί τι συμβαίνει όταν η μέθοδος flush καλείται ,ενώ μια εκροή είναι ήδη υπό εξέλιξη. Θα πρέπει επίσης να αποφασιστεί τι συμβαίνει όταν και το αντικείμενο αισθητήρας και το αντικείμενο που τραβά με το χέρι το καζανάκι καλούν την flush μέθοδο του αντικειμένου Καζανάκι. Αυτές είναι οι τρεις προφανέστερες επιλογές για το πώς να χειριστεί μια κλήση στην flush μέθοδο, ενώ υπάρχει εκροή ένα υπό εξέλιξη.

1). Αρχίζει μια νέα εκροή αμέσως. Η έναρξη μιας νέας εκροής ενώ μια εκροή είναι ήδη υπό εξέλιξη έχει την ίδια επίδραση, με το να κάνει την εκροή υπό εξέλιξη να διαρκέσει πιο πολύ από μια κανονική εκροή. Το βέλτιστο μήκος μιας κανονικής εκροής έχει καθοριστεί μέσω της εμπειρίας. Μια πιο μακροχρόνια εκροή θα σπαταλήσει το νερό, έτσι αυτό δεν είναι μια καλή επιλογή.

2). Αναμονή έως ότου τελειώσει η τρέχουσα εκροή και έναρξη αμέσως μιας άλλης. Αυτή η επιλογή διπλασιάζει το μήκος μιας εκροής. Ξοδεύεται περισσότερο νερό, από την πρώτη επιλογή.

3). Να μη γίνει τίποτα. Αυτή η επιλογή δεν σπαταλά καθόλου νερό, έτσι είναι η καλύτερη επιλογή.
Όταν υπάρχουν δύο ταυτόχρονες κλήσεις στην flush μέθοδο, επιτρέπεται σε μια να εκτελεστεί και αγνοείται η άλλη.

Ένα αντικείμενο Πελάτη καλεί μια μέθοδο ενός αντικειμένου. Εάν η μέθοδος του αντικειμένου καλείται όταν είναι το αντικείμενο σε μια κατάσταση ακατάλληλη για την εκτέλεση μιας κλήσης στη μέθοδο, τότε η μέθοδος επιστρέφει χωρίς εκτέλεση των συνηθισμένων λειτουργιών της [34]. Η μέθοδος επιστρέφει ένα αποτέλεσμα. Το αποτέλεσμα είναι είτε αληθινό (true) είτε ψεύτικο (false), δείχνοντας εάν η μέθοδος εκτέλεσε τις κανονικές λειτουργίες της ή τις εμπόδισε.

Οι συνέπειες του προτύπου εμποδισμού είναι οι εξής:

1). Ένα αντικείμενο μπορεί να είναι σε μια κατάσταση στην οποία είναι ακατάλληλο να εκτελεστεί μια κλήση μεθόδου.

2). Η αναβολή της εκτέλεσης μιας κλήσης μεθόδου δεν είναι μια καλή πολιτική για το πρόβλημα. Η κλήση μεθόδου πρέπει να εκτελεστεί αμέσως για να παραγάγει το σωστό αποτέλεσμα.

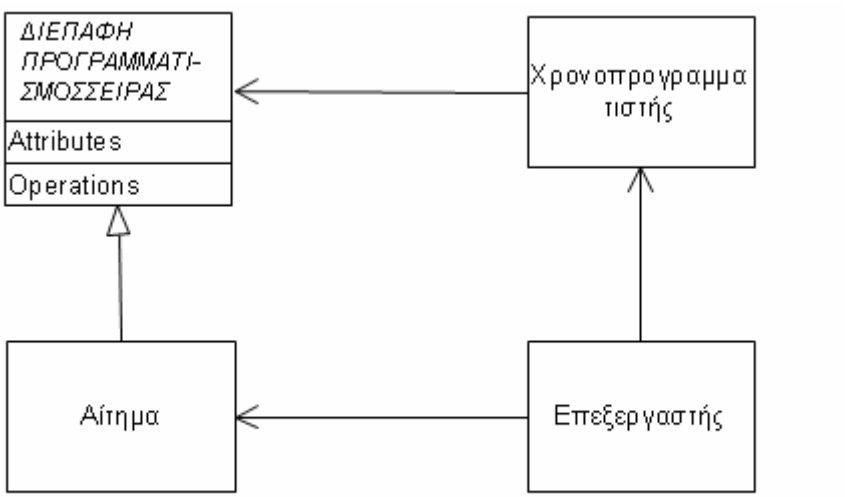
3). Οι κλήσεις που γίνονται στη μέθοδο ενός αντικειμένου όταν δεν είναι το αντικείμενο σε μια κατάλληλη κατάσταση για να εκτελέσει τη μέθοδο ,μπορούν να αγνοηθούν.

Για το πρότυπο αυτό δεν υπάρχει κάποιο συγκεκριμένο διάγραμμα κλάσεων.

7.4 Πρότυπο Χρονοπρογραμματιστή (Scheduler Pattern)

Στον προγραμματισμό υπολογιστών το πρότυπο Χρονοπρογραμματιστή είναι ένα σχεδιαστικό πρότυπο λογισμικού. Είναι ένα πρότυπο συγχρονισμού που χρησιμοποιείται για να ελέγξει, όταν τα νήματα μπορούν να εκτελέσουν απλού νήματος κώδικα. Το πρότυπο Χρονοπρογραμματιστή χρησιμοποιεί ένα αντικείμενο που βάζει σε σειρά νήματα που περιμένουν. Παρέχει έναν μηχανισμό για να εφαρμόσει μια πολιτική σχεδιασμού, αλλά είναι ανεξάρτητο από οποιαδήποτε συγκεκριμένη πολιτική. Η πολιτική είναι ενθυλακωμένη στην κλάση της και είναι επαναχρησιμοποιήσιμη [5].
Το πρότυπο Χρονοπρογραμματιστή χρησιμοποιεί ένα αντικείμενο για να σχεδιάσει τα ταυτόχρονα αιτήματα από τα νήματα για τη μη συγχρονισμένη επεξεργασία. Κάθε νήμα που προσπαθεί να εκτελέσει μια χρονοπρογραμματισμένη μέθοδο εμποδίζεται έως ότου την ξυπνούν. Εξ ορισμού, ο σχεδιασμός της διάταξης- σειράς είναι FIFO (First In- First Out). Η σειρά μπορεί να προσδιοριστεί ή μπορεί να εφαρμοστεί μια άλλη πολιτική σχεδιασμού [6].

Στο επόμενο σχήμα απεικονίζεται το διάγραμμα κλάσεων του προτύπου Χρονοπρογραμματιστή [5] [33].



Σχήμα 7.4.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Χρονοπρογραμματιστή.

Οι συμμετέχοντες στο διάγραμμα και οι ρόλοι που έχουν είναι οι εξής:

1). Αίτημα.
Οι κλάσεις σε αυτόν τον ρόλο πρέπει να εφαρμόσουν τη διεπαφή στο ρόλο ΠρογραμματισμόςΣειράς. Τα αντικείμενα Αιτήματος ενθυλακώνουν ένα αίτημα για ένα αντικείμενο Επεξεργαστή.

2). Επεξεργαστής.
Στιγμιότυπα κλάσεων σε αυτόν τον ρόλο εκτελούν έναν υπολογισμό που περιγράφεται από ένα αντικείμενο Αιτήματος. Μπορούν να αναπαρασταθούν με περισσότερα από ένα αντικείμενα Αιτήματος που επεξεργάζονται σε μια στιγμή, αλλά μπορούν να επεξεργαστούν μόνο ένα σε κάθε στιγμή. Ένα αντικείμενο Επεξεργαστή εκπροσωπεί ένα αντικείμενο Χρονοπρογραμματιστή την ευθύνη για το χρονοπρογραμματισμό αντικείμενων Αιτήματος, ένα κάθε φορά.

3). Χρονοπρογραμματιστής.
Στιγμιότυπα κλάσεων σε αυτό το ρόλο χρονοπρογραμματίζουν τα αντικείμενα Αιτήματος για την επεξεργασία από ένα αντικείμενο Επεξεργαστή. Για να προωθήσει την ικανότητα επαναχρησιμοποίησης, μια κλάση Χρονοπρογραμματιστή δεν γνωρίζει τίποτα για την κλάση Αίτηση που σχεδιάζει. Όμως, έχει πρόσβαση στα αντικείμενα Αίτησης μέσω της διεπαφής ΠρογραμματισμόςΣειράς που εφαρμόζουν. Μια κλάση σε αυτόν τον ρόλο είναι αρμόδια για να αποφασίσει, πότε θα τρέξει το επόμενο αίτημα. Δεν είναι αρμόδιο για τη σειρά στην οποία τα αιτήματα θα τρέξουν. Εκπροσωπεί την ευθύνη σε μια διεπαφή ΠρογραμματισμόςΣειράς.

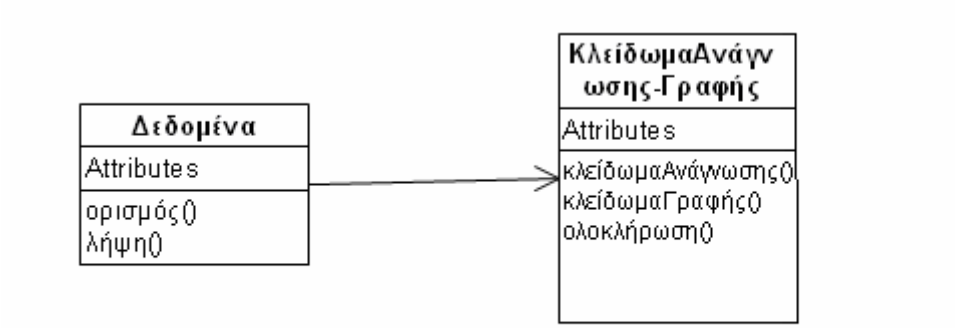
4). ΠρογραμματισμόςΣειράς.
Τα αντικείμενα Αιτήματος εφαρμόζουν τη διεπαφή που είναι σε αυτόν τον ρόλο. Οι διεπαφές σε αυτόν τον ρόλο εξυπηρετούν δύο σκοπούς. Πρώτον, αναφέροντας σε μια διεπαφή ΠρογραμματισμόςΣειράς, οι κλάσεις Επεξεργαστή αποφεύγουν μια εξάρτηση σε μια κλάση Αίτησης. Δεύτερον, με την κλήση των μεθόδων που καθορίζονται από τη διεπαφή ΠρογραμματισμόςΣειράς, οι κλάσεις Χρονοπρογραμματιστή είναι σε θέση να παίρνουν την απόφαση για το ποιο

αντικείμενο Αίτησης θα υποβληθεί σε επεξεργασία μετά. Αυτό αυξάνει την ικανότητα επαναχρησιμοποίησης των κλάσεων Χρονοπρογραμματιστή.

Δυο θετικές συνέπειες του προτύπου Χρονοπρογραμματιστή είναι οι εξής. Η πρώτη είναι ότι το πρότυπο αυτό παρέχει ένα τρόπο να ελέγξει, όταν τα νήματα μπορεί να εκτελούν ένα κομμάτι του κώδικα. Η δεύτερη είναι ότι η πολιτική χρονοπρογραμματισμού ενθυλακώθηκε στην ίδια της την κλάση και είναι επαναχρησιμοποιήσιμη.
Μια αρνητική συνέπεια είναι ότι η χρησιμοποίηση του προτύπου προσθέτει σημαντικά έξοδα πέρα από αυτό που πρέπει να κάνει μια απλή κλήση σε μια συγχρονισμένη μέθοδο [33].

7.5 Πρότυπο Ανάγνωσης-γραφής κλειδαριάς- κλειδώματος (Read-Write Lock Pattern)

Ένα πρότυπο ανάγνωσης-γραφής κλειδώματος ή απλά ένα RWL (Read-Write Lock) είναι ένα σχεδιαστικό πρότυπο λογισμικού που επιτρέπει την ταυτόχρονη πρόσβαση ανάγνωσης σε ένα αντικείμενο αλλά απαιτεί αποκλειστική πρόσβαση για τις μεθόδους γραφής. Σε αυτό το πρότυπο, οι πολλοί αναγνώστες μπορούν να διαβάσουν τα στοιχεία παράλληλα, αλλά ένα αποκλειστικό κλείδωμα απαιτείται όταν γράφονται τα στοιχεία. Όταν ένας γράφει τα στοιχεία, οι αναγνώστες θα εμποδιστούν έως ότου τελειώσει το γράψιμο [7] [9].
Το κλείδωμα ανάγνωσης-γραφής είναι μια βελτίωση απόδοσης για τις περιπτώσεις όπου η ανάγνωση ξεπερνά αριθμητικώς το γράψιμο. Η ιδέα είναι να χωριστούν τα κλειδώματα σε δύο κλάσεις. Στην κλάση ανάγνωσης και κλάση γραφής. Τα πολλαπλά ταυτόχρονα κλειδώματα ανάγνωσης μπορούν να κρατηθούν, αλλά γράφονται ότι τα κλειδώματα κρατούνται αποκλειστικά. Το αποκλειστικό κλείδωμα γραψίματος εξασφαλίζει ότι δεν εμφανίζεται ανταγωνισμός, εάν ένας χρήστης γράφει στα στοιχεία κανένας άλλος χρήστης δεν μπορεί να διαβάσει ή να γράψει. Επίσης, η άδεια για πολλά ταυτόχρονα κλειδώματα ανάγνωσης μειώνει τη διαμάχη των πόρων, δεδομένου ότι οι πολλοί αναγνώστες μπορούν ακίνδυνα να χρησιμοποιήσουν τα κοινά στοιχεία [8].
Δηλαδή, η λογική για το συντονισμό των διαδικασιών γραφής και ανάγνωσης πρέπει να είναι επαναχρησιμοποιήσιμη. Υπάρχει μια ανάγκη για πρόσβαση ανάγνωσης και γραφής στην πληροφορία κατάστασης ενός αντικειμένου. Οποιοσδήποτε αριθμός διαδικασιών ανάγνωσης μπορεί να εκτελεστεί στις πληροφορίες κατάστασης του αντικειμένου ταυτόχρονα. Όμως, οι διαδικασίες ανάγνωσης εγγυώνται ότι θα επιστρέψουν τη σωστή τιμή, μόνο εάν δεν υπάρχει διαδικασία γραφής, που εκτελείται ταυτόχρονα με μια λειτουργία ανάγνωσης. Οι διαδικασίες γραφής στις πληροφορίες κατάστασης του αντικειμένου πρέπει να εκτελούνται μια κάθε φορά, για να εξασφαλιστεί η ακρίβειά τους.
Το επόμενο διάγραμμα κλάσεων παρουσιάζει τους ρόλους που παίζουν οι κλάσεις στο Πρότυπο Ανάγνωσης-γραφής κλειδαριάς- κλειδώματος (Read/Write Lock pattern).



Σχήμα 7.5.1 Απεικονίζεται το διάγραμμα κλάσεων του Πρότυπου Ανάγνωσης-γραφής κλειδαριάς- κλειδώματος (Read-Write Lock Pattern).

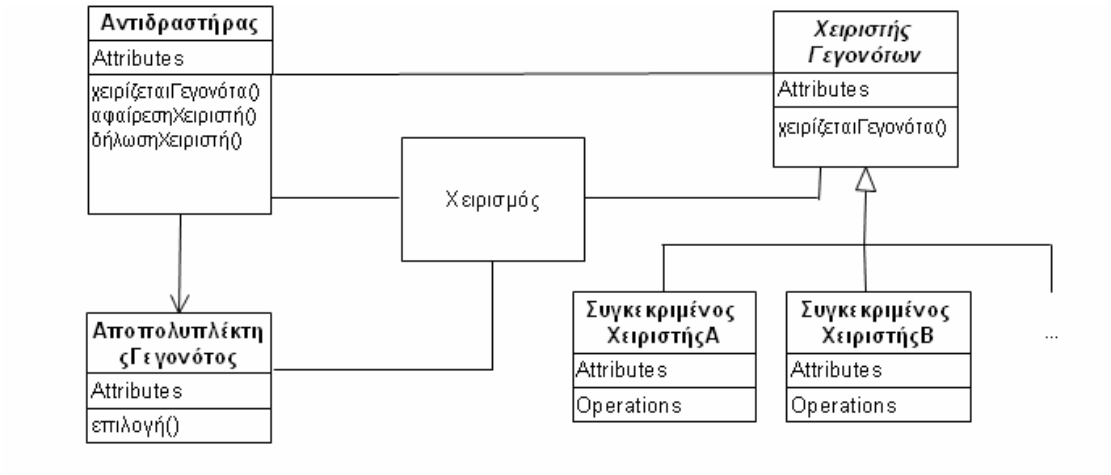
Μια κλάση στο ρόλο Δεδομένα έχει μεθόδους που παίρνουν-λαμβάνουν και μεθόδους που ορίζουν τις πληροφορίες των στιγμιότυπων της. Οποιοσδήποτε αριθμός νημάτων μπορεί ταυτόχρονα να πάρει τις πληροφορίες ενός αντικειμένου Δεδομένων, εφ' όσον κανένα νήμα δεν ορίζει πληροφορίες, την ίδια στιγμή. Οι μέθοδοι ορισμού της, πρέπει να εμφανιστούν μια κάθε στιγμή, ενώ καμία λειτουργία δεν εκτελείται. Τα αντικείμενα Δεδομένων πρέπει να συντονίσουν τις μεθόδους λήψης και ορισμού, έτσι υπακούνε στους περιορισμούς.
Τα αντικείμενα Δεδομένα χρησιμοποιούν μια αφαίρεση που συντονίζει τις διαδικασίες λήψης. Η αφαίρεση αυτή είναι μια κλειδαριά ανάγνωσης. Οι μέθοδοι λήψης ενός αντικειμένου Δεδομένου, δεν προσκομίζουν πληροφορίες έως ότου πάρουν μια κλειδαριά ανάγνωσης. Ένα αντικείμενο ΚλείδωμαΑνάγνωσης-Γραφής είναι συνδεμένο με κάθε αντικείμενο Δεδομένου. Προτού μια από τις μεθόδους λήψης πάρει οτιδήποτε, καλεί τη μέθοδο κλείδωμαΑνάγνωσης του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής, η οποία διανέμει ένα κλείδωμα ανάγνωσης στο τρέχον νήμα. Ενώ το νήμα έχει ένα κλείδωμαΑνάγνωσης, η μέθοδος λήψης μπορεί να είναι βέβαια ότι είναι ασφαλές να αποκτηθούν τα στοιχεία από το αντικείμενο. Αυτό γίνεται, επειδή ενώ υπάρχουν οποιεσδήποτε σημαντικά κλειδώματα ανάγνωσης, το αντικείμενο ΚλείδωμαΑνάγνωσης-Γραφής δεν θα εκδώσει κλείδωμα εγγραφής. Εάν υπάρχει οποιαδήποτε εκκρεμότητα με κλείδωμα εγγραφής, όταν καλείται η μέθοδος κλείδωμαΑνάγνωσης του αντικείμενου ΚλείδωμαΑνάγνωσης-Γραφής, δεν επιστρέφει έως ότου όλες οι μέθοδοι ΚλείδωμαΕγγραφής που εκκρεμούν έχουν σταματήσει, από κλήσεις στη ολοκλήρωση μέθοδο του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής. Διαφορετικά, κλήσεις στη μέθοδο κλείδωμαΑνάγνωσης του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής επιστρέφουν αμέσως [33].
Όταν μια μέθοδος λήψης ενός αντικειμένου Δεδομένων έχει τελειώσει παίρνοντας τα δεδομένα από το αντικείμενο, καλεί τη μέθοδο ολοκλήρωση του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής. Μια κλήση σε εκείνη την μέθοδο αναγκάζει το τρέχον νήμα να σταματήσει το κλείδωμα ανάγνωσης [1].
Ομοίως, τα αντικείμενα δεδομένων χρησιμοποιούν μια αφαίρεση κλειδώματος εγγραφής . Οι μέθοδοι ορισμού ενός αντικειμένου Δεδομένων δεν αποθηκεύουν πληροφορίες, έως ότου πάρουν κλείδωμα εγγραφής. Προτού να αποθηκεύσει οποιεσδήποτε πληροφορίες από τις μεθόδους ορισμού ενός αντικειμένου Δεδομένου, καλεί τη μέθοδο κλείδωμαΕγγραφής του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής, η οποία εκδίδει την μέθοδο κλείδωμαΕγγραφής στο τρέχον νήμα. Ενώ το νήμα έχει κλείδωμα εγγραφής, η μέθοδος ορισμού μπορεί να είναι σίγουρη ότι είναι ασφαλής η αποθήκευση δεδομένων στο αντικείμενο. Αυτό γίνεται, επειδή τα αντικείμενα

ΚλείδωμαΑνάγνωσης-Γραφής χρησιμοποιούν κλείδωμα εγγραφής, μόνο όταν δεν υπάρχει καμία εκκρεμής κλείδωμαΑνάγνωσης και καμία εκκρεμής κλείδωμαΕγγραφής. Εάν υπάρχουν οποιεσδήποτε εκκρεμείς κλειδαριές ,τότε όταν καλείται η μέθοδος κλείδωμαΕγγραφής του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής, δεν επιστρέφει έως ότου έχουν σταματήσει όλες τις εκκρεμείς κλειδαριές από κλήσεις στη μέθοδο ολοκλήρωση του αντικειμένου ΚλείδωμαΑνάγνωσης-Γραφής.
Οι προηγούμενοι περιορισμοί που ισχύουν όταν εκδίδονται οι κλειδαριές ανάγνωσης και εγγραφής, δεν εξετάζουν τη σειρά με την οποία εκδίδονται οι κλειδαριές ανάγνωσης και εγγραφής. Η σειρά με την οποία εκδίδονται οι κλειδαριές ανάγνωσης και εγγραφής δεν παίζει ρόλο, εφ' όσον οι διαδικασίες λήψης μπορούν να εκτελεστούν ταυτόχρονα. Δεδομένου ότι οι διαδικασίες εγγραφής εκτελούνται μια κάθε φορά, η σειρά με την οποία εκδίδονται οι κλειδαριές ανάγνωσης και εγγραφής πρέπει να είναι η σειρά στην οποία ζητούνται οι κλειδαριές ανάγνωσης και εγγραφής [33].

Το πρότυπο αυτό έχει ορισμένες συνέπειες. Το σχεδιαστικό πρότυπο ανάγνωσης-γραφής κλειδώματος αυξάνει το συγχρονισμό των μεθόδων ανάγνωσης και επιτυγχάνει τον αμοιβαίο αποκλεισμό [10]. Επιτρέπει επίσης την επαναχρησιμοποίηση της λογικής ελέγχου συγχρονισμού, αυξάνοντας το συγχρονισμό όποτε υπάρχουν περισσότερες διεργασίες ανάγνωσης από τις διεργασίες γραφής. Εντούτοις, αυτό το πρότυπο δεν αποδίδει καλά όταν ο αριθμός των διαδικασιών γραφής είναι μεγαλύτερος από τον αριθμό των διαδικασιών ανάγνωσης [1].

7.6 Πρότυπο αντιδραστήρα (Reactor Pattern)

Η δυνατότητα να αντιμετωπιστούν και να αποσταλούν ταυτόχρονα εμφανιζόμενα γεγονότα, αποτελεσματικά χωρίς οποιαδήποτε έξοδα, είναι ένα αναπόσπαστο τμήμα του υλικολογισμικού που χρησιμοποιείται σε πραγματικό χρόνο, σε κρίσιμο σε απόδοση περιβάλλον. Για τη σωστή λειτουργία του ολόκληρου συστήματος, η απόδοση του υλικολογισμικού για αυτόν τον ιδιαίτερο στόχο πρέπει να είναι βέλτιστη. Το Πρότυπο Αντιδραστήρα αναστρέφει τη ροή του ελέγχου σε ένα σύστημα κατά τη διάρκεια του χειρισμού γεγονότος [36].
Το πρότυπο αντιδραστήρα επιτρέπει τις γεγονός-οδηγημένες (data driven) εφαρμογές να αποπολυπλεχτούν (demultiplex) και να αποσταλούν τα αιτήματα υπηρεσιών που παραδίδονται σε μια εφαρμογή από έναν ή περισσότερους χρήστες. [11].
Λαμβάνει τα εισερχόμενα μηνύματα ή τα αιτήματα από τους πολλούς ταυτόχρονους πελάτες. Επεξεργάζεται αυτά τα μηνύματα χρησιμοποιώντας τους χειριστές γεγονότος, έναν κάθε φορά. Είναι χρήσιμο στους κεντρικούς υπολογιστές, συστήματα γραφικής παράστασης, και αλλού [35].
Το διάγραμμα κλάσεων του προτύπου αντιδραστήρα φαίνεται στο επόμενο σχήμα [9] [12] [13] [35] [36].



Σχήμα 7.6.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου αντιδραστήρα.

Το πρότυπο αποτελείται από τους εξής συμμετέχοντες [12] [36]:

1). Χειρισμός.
Λαμβάνει τα γεγονότα. Προσδιορίζει μεμονωμένα τις πηγές γεγονότος όπως οι συνδέσεις δικτύων ή τα ανοικτά αρχεία. Όποτε ένα γεγονός παράγεται από μια πηγή γεγονότος, περιμένει στη σειρά επάνω στη χειριστή.

2). Σύγχρονος αποπολυπλέκτης γεγονότος (Synchronous Event Demultiplexer).
Αυτή η οντότητα εφαρμόζεται πραγματικά ως κλήση μεθόδου, όπως η επιλογή. Περιμένει ένα ή περισσότερα γεγονότα να εμφανιστούν και διαδίδει έπειτα αυτά τα γεγονότα στον Αντιδραστήρα.

3). Αντιδραστήρας.
Ο Αντιδραστήρας είναι ο μηχανισμός αποστολής του προτύπου αντιδραστήρα. Σε απάντηση σε ένα γεγονός, αποστέλλει τον αντίστοιχο χειριστή γεγονότος για εκείνο το γεγονός.

4). Χειριστής γεγονότων.
Οι Χειριστές γεγονότων είναι οι οντότητες που επεξεργάζονται πραγματικά το γεγονός. Αυτοί καταχωρούνται με τον αντιδραστήρα και αποστέλλονται από τον αντιδραστήρα όταν εμφανίζεται το γεγονός για το οποίο καταχωρούνται.

5). Συγκεκριμένοι χειριστές (ΣυγκεκριμένοςΧειριστήςΑ, ΣυγκεκριμένοςΧειριστήςΒ, κτλ).
Οι συγκεκριμένοι χειριστές ειδικεύουν το γενικό ΧειριστήΓεγονότων. Είναι αρμόδιοι για την επεξεργασία συγκεκριμένων τύπων γεγονότων.
Αρχικά η εφαρμογή δηλώνει τους χειριστές γεγονότος της, καθώς επίσης και το Χειρισμό ότι προετοιμάζονται να αποδεχθούν, με τον Αντιδραστήρα. Ο Αντιδραστήρας συλλέγει όλες τις καταχωρημένες Χειρισμούς. Στη συνέχεια, όποτε ένα γεγονός εμφανίζεται, ο Σύγχρονος αποπολυπλέκτης γεγονότος το διοχετεύει στον αντιδραστήρα. Με βάση το Χειρισμό για εκείνο το γεγονός, ο Αντιδραστήρας αποστέλλει τον αντίστοιχο δηλωμένο χειριστή γεγονότος. Ο χειριστής γεγονότος επεξεργάζεται έπειτα το γεγονός. Κατ' αυτό τον τρόπο, η εφαρμογή δεν ξοδεύει πολύ

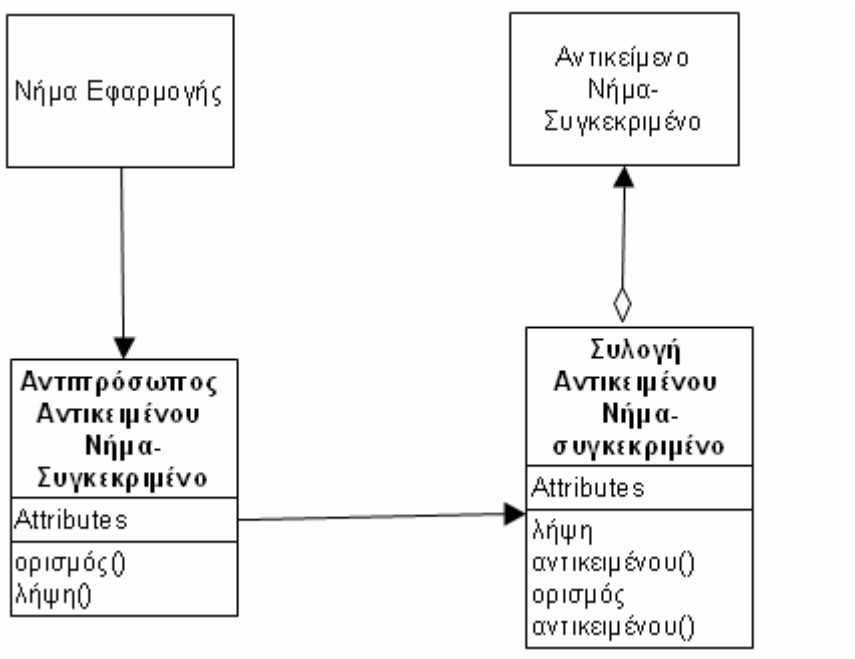
χρόνο για ένα γεγονός, ούτε πρέπει να εξετάσει την αποπολύπλεξη του γεγονότος στον αρμόδιο χειριστή.
Το πρότυπο αντιδραστήρα έχει τα ακόλουθα οφέλη [12]:
1). Βελτιώνει τη διαμόρφωση, την ικανότητα επαναχρησιμοποίησης, και επεξεργασία των γεγονός-οδηγημένων εφαρμογών. Αυτό γίνεται με την αποσύζευξη-αποπολύπλεξη των ανεξάρτητων από εφαρμογή μηχανισμών από τις πολιτικές επεξεργασίας της εφαρμογής [14].
2). Βελτιώνει τη φορητότητα εφαρμογής επιτρέποντας στην διεπαφή της να επαναχρησιμοποιηθεί ανεξάρτητα από τις κλήσεις λειτουργικών συστημάτων (OS) που εκτελούν την αποδιαύλωση γεγονότος.
3). Παρέχει στις εφαρμογές συγχρονισμένο έλεγχο που ελαχιστοποιεί την ανάγκη για τον περίπλοκο συγχρονισμό ή το κλείδωμα μέσα σε μια διαδικασία εφαρμογής.

Το πρότυπο αντιδραστήρα έχει το εξής μειονέκτημα [12]:
1). Οι εφαρμογές που έχουν γραφτεί χρησιμοποιώντας το πρότυπο αντιδραστήρα είναι δύσκολο να διορθωθούν, επειδή η ροή ελέγχου τους ταλαντεύεται μεταξύ του κώδικα χαμηλού επιπέδου και των υψηλότερου επιπέδου μεθόδων που παρέχονται από τους υπεύθυνους για την ανάπτυξη εφαρμογής.

7.7 Πρότυπο Αποθήκευσης Συγκεκριμένου-Νήματος (Thread Specific Storage Pattern)

Το Πρότυπο Αποθήκευσης Συγκεκριμένου-Νήματος (Thread-Specific Storage Pattern) βελτιώνει την απόδοση και απλοποιεί τις πολύπλοκες εφαρμογές επιτρέποντας πολλά νήματα να χρησιμοποιούν ένα ‘λογικά σφαιρικό’ (global) σημείο πρόσβασης, για να ανακτήσει ένα αντικείμενο, που είναι τοπικό σε ένα νήμα χωρίς να υποστεί έξοδα για κάθε πρόσβαση [38] [17].

Το πρότυπο χρησιμοποιείται, όταν μια εφαρμογή έχει τα ακόλουθα χαρακτηριστικά:
1). Περιέχει πολλά νήματα ελέγχου που μπορούν να εκτελέσουν ταυτόχρονα σε μια αυθαίρετη σειρά σχεδιασμού.
2). Κάθε νήμα ελέγχου επικαλείται ακολουθίες μεθόδων που μοιράζονται κοινά δεδομένα μόνο για εκείνο το νήμα.
3). Τα δεδομένα που μοιράζονται από αντικείμενα μέσα σε κάθε νήμα πρέπει να προσεγγιστούν μέσω ενός συνολικά ορατού σημείου πρόσβασης που ‘λογικά’ μοιράζεται με άλλα νήματα, αλλά ‘φυσικά’ μοναδικό για κάθε νήμα.
4). Τα δεδομένα περνούν σιωπηρά μεταξύ των μεθόδων, αντί να περαστούν ρητά μέσω των παραμέτρων.
Στο επόμενο σχήμα φαίνεται το διάγραμμα κλάσεων του προτύπου [15] [16] [17] [18].



Σχήμα 7.7.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Αποθήκευσης Συγκεκριμένου-Νήματος (Thread Specific Storage Pattern).

Το παραπάνω σχήμα επεξηγεί τη δομή των συμμετεχόντων στο προτύπου Αποθήκευσης Συγκεκριμένου-Νήματος [15].

1). Νήμα εφαρμογής.
Χρησιμοποιεί την ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο για να έχει πρόσβαση στο ΑντικείμενοΝήμα-Συγκεκριμένο.

2). ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο.
Το ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο ορίζει τη διεπαφή του ΑντικειμένουΝήμα-Συγκεκριμένο. Είναι αρμόδιο για την παροχή πρόσβασης σε ένα μοναδικό αντικείμενο, για κάθε νήμα εφαρμογής, μέσω των μεθόδων λήψη και ορισμός. Ένα ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο είναι αρμόδιο για ένα τύπο αντικειμένου δηλαδή αυτό δίνει την πρόσβαση σε ένα ΑντικείμενοΝήμα-Συγκεκριμένο για κάθε νήμα που έχει πρόσβαση στον αντιπρόσωπο.

3). ΑντικείμενοΝήμα-Συγκεκριμένο.
Ένα ΑντικείμενοΝήμα-Συγκεκριμένο είναι το στιγμιότυπο ενός ιδιαίτερου νήματος ενός ΑντικείμενοΝήμα-Συγκεκριμένο. Χειρίζεται από τη ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο και γίνεται προσβάσιμο μόνο μέσω ενός ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο.

4). ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο.
Σε σύνθετες εφαρμογές, ένα νήμα για να ανακτήσει τα λάθος δεδομένα Νήμα-Συγκεκριμένα, πρέπει να χρησιμοποιήσει ένα κλειδί. Η ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο περιέχει ένα σύνολο όλων των ΑντικείμενοΝήμα-Συγκεκριμένο που συνδέονται με ένα συγκεκριμένο νήμα . Δηλαδή κάθε νήμα έχει μια μοναδική ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο. Η ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο σχεδιάζει κλειδιά για ΑντικείμενοΝήμα-Συγκεκριμένο. Ένα

ΑντιπρόσωποςΑντικειμένουΝήμα-Συγκεκριμένο χρησιμοποιεί το κλειδί για να ανακτήσει ένα ΑντικείμενοΝήμα-Συγκεκριμένο από την ΣυλογήΑντικειμένουΝήμα-Συγκεκριμένο μέσω των μεθόδων λήψηΑντικειμένου και ορισμόςΑντικειμένου.

Τα οφέλη του Πρότυπου Αποθήκευσης Συγκεκριμένου-Νήματος (Thread Specific Storage Pattern) είναι η αποδοτικότητα, η ευκολία χρήσης και η φορητότητα.
Όσο αφορά την αποδοτικότητα, το πρότυπο αυτό μπορεί να εφαρμοστεί, έτσι ώστε κανένα κλείδωμα να μην απαιτείται στα νήματα-συγκεκριμένα δεδομένα.
Επίσης το Πρότυπο Αποθήκευσης Συγκεκριμένου-Νήματος είναι απλό για προγραμματιστές εφαρμογής, επειδή οι υπεύθυνοι για την ανάπτυξη συστημάτων μπορούν να χρησιμοποιήσουν την αποθήκευση Συγκεκριμένου-Νήματος που είναι απολύτως διαφανή στο επίπεδο του κώδικα (source code) μέσω της αφαίρεσης δεδομένων.
Με την φορητότητα εννοείται ότι η νήμα-συγκεκριμένη αποθήκευση είναι διαθέσιμη στις περισσότερες πολύπλοκες πλατφόρμες λειτουργικών συστημάτων (OS).

Υπάρχουν επίσης τα ακόλουθα δυο μειονεκτήματα στη χρησιμοποίηση του Πρότυπου Αποθήκευσης Συγκεκριμένου-Νήματος.
Πολλές εφαρμογές δεν απαιτούν πολλά νήματα να έχουν πρόσβαση στα νήματα-συγκεκριμένα δεδομένα, μέσω ενός κοινού σημείου πρόσβασης. Έτσι, τα δεδομένα πρέπει να αποθηκευτούν έτσι ώστε μόνο το νήμα στο οποίο ανήκουν τα δεδομένα μπορεί να έχει πρόσβαση σε αυτά.
Ένα άλλο μειονέκτημα είναι ότι το πρότυπο κρύβει τη δομή του συστήματος. Η χρήση της αποθήκευσης συγκεκριμένου νήματος κρύβει τις σχέσεις μεταξύ των αντικειμένων σε μια εφαρμογή, καθιστώντας την εφαρμογή δύσκολο να γίνει κατανοητή.

Δεν είναι πολύ καλό να χρησιμοποιηθεί το πρότυπο σε μια εφαρμογή, όπου πολλά νήματα συνεργάζονται σε μια ενιαία εργασία που απαιτεί την ταυτόχρονη πρόσβαση σε κοινά δεδομένα. Για παράδειγμα, μια πολύπλοκη εφαρμογή μπορεί να εκτελέσει ανάγνωση και γραφή ταυτόχρονα σε μια βάση δεδομένων-μνήμης. Σε αυτήν την περίπτωση, τα νήματα πρέπει να μοιραστούν τα αρχεία και τους πίνακες που δεν είναι νήμα-συγκεκριμένοι. Εάν η νήμα-συγκεκριμένη αποθήκευση χρησιμοποιήθηκε για να αποθηκεύσει τη βάση δεδομένων, τα νήματα δεν θα μπορούσαν να μοιραστούν τα δεδομένα.

7.8 Πρότυπο Αποθέματος Νημάτων (Thread Pool Pattern)

Στον προγραμματισμό του προτύπου αποθέματος νημάτων, δημιουργούνται διάφορα νήματα για να εκτελέσουν διάφορες εργασίες, οι οποίες οργανώνονται συνήθως σε μια σειρά αναμονής (ουρά). Συνήθως, υπάρχουν πολλές περισσότερες εργασίες από τα νήματα. Μόλις ένα νήμα ολοκληρώνει την υποχρέωσή του, θα ζητήσει την επόμενη εργασία από τη σειρά αναμονής έως ότου έχουν ολοκληρωθεί όλες οι υποχρεώσεις. Το νήμα μπορεί τότε να τερματιστεί ή να περιμένει έως ότου υπάρχουν νέες εργασίες διαθέσιμες.
Ο αριθμός χρησιμοποιούμενων νημάτων είναι μια παράμετρος που μπορεί να συντονιστεί για να παρέχει την καλύτερη απόδοση. Επιπλέον, ο αριθμός νημάτων μπορεί να είναι δυναμικός βασισμένος στον αριθμό των εργασιών που περιμένουν. Για παράδειγμα, ένας κεντρικός υπολογιστής διαδικτύου μπορεί να προσθέσει τα νήματα εάν υπάρχουν πολυάριθμα αιτήματα ιστοσελίδας και μπορεί να αφαιρέσει τα

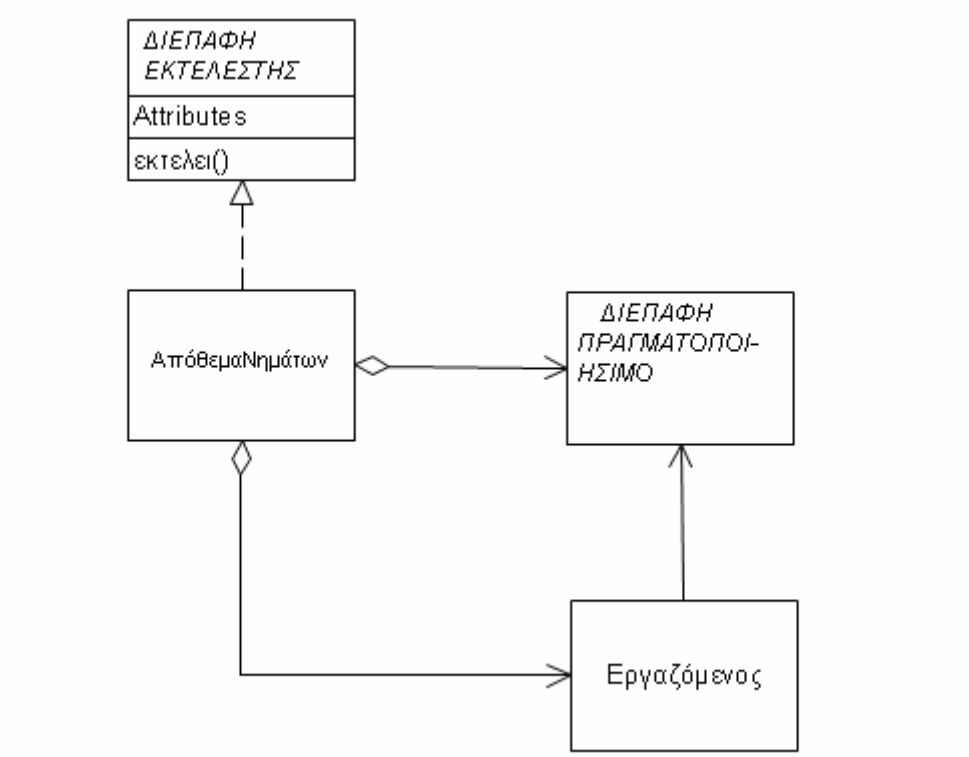
νήματα, όταν δεν υπάρχουν τόσες πολλές αιτήσεις. Το κόστος να υπάρχει ένα μεγαλύτερο απόθεμα νημάτων είναι αυξανόμενη χρήση των πόρων. Ο αλγόριθμος που καθορίζει πότε θα δημιουργούνται ή θα καταστρέφονται τα νήματα ασκεί επίδραση στη γενική απόδοση [20].

Αν δημιουργηθούν πάρα πολλά νήματα και οι πόροι σπαταλιούνται, θα υπάρχει και χρόνος που σπαταλιέται δημιουργώντας τα αχρησιμοποίητα νήματα [22]. Αν καταστραφούν πάρα πολλά νήματα και θα ξοδευτεί αργότερα περισσότερος χρόνος δημιουργώντας τα πάλι. Η δημιουργία των νημάτων πάρα πολύ αργά θα οδηγήσει στην κακή απόδοση χρηστών (θα περιμένουν πολύ χρόνο). Η καταστροφή των νημάτων πάρα πολύ αργά μπορεί να αργοπορήσει άλλες διαδικασίες των πόρων. Επομένως ο αλγόριθμος που επιλέγεται θα εξαρτηθεί από το πρόβλημα και τα αναμενόμενα πρότυπα χρήσης.

Το πλεονέκτημα με ένα απόθεμα νημάτων πέρα από τη δημιουργία ενός νέου νήματος για κάθε στοιχειώδη εργασία, είναι ότι τα γενικά έξοδα δημιουργίας νημάτων και καταστροφής μειώνονται, τα οποία μπορούν να οδηγήσουν στην καλύτερη απόδοση και την καλύτερη σταθερότητα συστημάτων. Επίσης, εάν ο αριθμός στοιχειωδών εργασιών είναι πολύ μεγάλος, η δημιουργία ενός νήματος για κάθε μια μπορεί να μην είναι πρακτική [20] [23].

Κατά εφαρμογή αυτού του προτύπου, ο προγραμματιστής πρέπει να εξασφαλίσει την ασφάλεια νήματος της σειράς αναμονής. Η ασφάλεια νημάτων είναι μια έννοια προγραμματισμού υπολογιστών που εφαρμόζεται στα πλαίσια των πολύπλοκων προγραμμάτων. Ένα κομμάτι του κώδικα είναι νήμα-ασφαλές εάν λειτουργεί σωστά κατά τη διάρκεια της ταυτόχρονης εκτέλεσης από τα πολλά νήματα. Ειδικότερα, πρέπει να ικανοποιήσει την ανάγκη για τα πολλά νήματα να προσεγγίσουν το ίδιο δεδομένο και την ανάγκη για ένα δεδομένο να προσεγγίζεται από ένα νήμα οποιαδήποτε στιγμή [19].

Το επόμενο σχήμα εμφανίζει τους ρόλους που παίζουν οι κλάσεις και οι διεπαφές στο πρότυπο ΑπόθεμαΝημάτων [23] [39].



Σχήμα 7.8.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Απόθεμα Νημάτων.

Εδώ περιγράφονται οι ρόλοι αυτοί [23]:

1). Εκτελεστής.
Μια διεπαφή σε αυτόν τον ρόλο ορίζει μια μέθοδο με την οποία περνά ένα πραγματοποιήσιμο αντικείμενο με σκοπό την εκτέλεση του. Η διεπαφή Εκτελεστή εφαρμόζεται από τις κλάσεις που είναι αρμόδιες για τον έλεγχο της εκτέλεσης των εργασιών.

2). ΑπόθεμαΝημάτων.
Οι κλάσεις σε αυτόν τον ρόλο εφαρμόζουν τη διεπαφή Εκτελεστή. Διαχειρίζονται ένα απόθεμα των νημάτων που χρησιμοποιούνται για να εκτελέσουν τις εργασίες που περνούν σε αυτές.

3). Πραγματοποιήσιμο.
Οι κλάσεις εφαρμόζουν τη διεπαφή Πραγματοποιήσιμο. Τα αντικείμενα Πραγματοποιήσιμα περνούν σε μια μέθοδο εκτέλεση του αντικειμένου ΑπόθεμαΝημάτων που τα τοποθετεί σε μια σειρά αναμονής ή παρόμοια δομή δεδομένων. Τα πραγματοποιήσιμα αντικείμενα μένουν στη σειρά αναμονής έως ότου τα παίρνει ένα μη απασχολημένο αντικείμενο Εργαζομένου από τη σειρά αναμονής και τα εκτελεί.

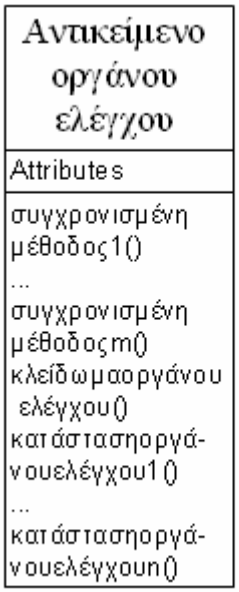
4). Εργαζόμενος.
Τα αντικείμενα Εργαζομένου συνδέονται με ένα νήμα. Ο σκοπός τους είναι να τρέξουν τις εργασίες που ενθυλακώνονται από τα πραγματοποιήσιμα αντικείμενα που περνούν στη ΑπόθεμαΝημάτων. Όταν ένα αντικείμενο Εργαζομένου δεν τρέχει μια

στοιχειώδη εργασία, περιμένει να πάρει το επόμενο Πραγματοποιήσιμο αντικείμενό του από το ΑπόθεμαΝημάτων.
Τα αντικείμενα ΑπόθεμαΝημάτων περιορίζουν γενικά τον αριθμό των αντικειμένων Εργαζομένων που υπάρχουν σε οποιοδήποτε στιγμή. Με τον περιορισμό του αριθμού αντικειμένων Εργαζομένων, περιορίζουν τον αριθμό νημάτων που χρησιμοποιούν.

Ενώ το απόθεμα νημάτων (pool treads) είναι ένας ισχυρός μηχανισμός για πολύπλοκες εφαρμογές, περιέχει ορισμένους κινδύνους. Οι εφαρμογές που χτίζονται με απόθεμα νημάτων μπορεί να εμφανίσουν κινδύνους, όπως τα λάθη συγχρονισμού, αδιέξοδο και διαρροή νημάτων [40].
Ένα σύνολο διαδικασιών ή νημάτων λέγεται ότι είναι σε αδιέξοδο, όταν περιμένει κάθε μια ένα γεγονός, που μόνο μια άλλη διαδικασία μπορεί να προκαλέσει. Η απλούστερη περίπτωση του αδιεξόδου είναι όπου το νήμα Α κρατά μια αποκλειστική κλειδαριά στο αντικείμενο Χ και περιμένει μια κλειδαριά στο αντικείμενο Υ, ενώ το νήμα Β κρατά μια αποκλειστική κλειδαριά στο αντικείμενο Υ και περιμένει την κλειδαριά στο αντικείμενο Χ. Αν δεν υπάρχει κάποιος τρόπος για να τερματιστεί το κλείδωμα, τότε τα νήματα θα περιμένουν για πάντα σε αυτήν την κατάσταση.
Εάν ο κώδικας δεν γραφτεί σωστά (λάθη συγχρονισμού), είναι δυνατό τα νήματα να παραμένουν σε κατάσταση αδράνειας, ακόμα κι αν υπάρχει εργασία στη σειρά αναμονής που υποβάλλεται σε επεξεργασία.
Ένας σημαντικός κίνδυνος σε όλα τα είδη αποθεμάτων νημάτων είναι η διαρροή, που εμφανίζεται όταν αφαιρείται ένα νήμα από το απόθεμα για να εκτελέσει έναν στόχο, αλλά δεν επιστρέφεται στο απόθεμα όταν ολοκληρωθεί ο στόχος. Όταν αυτό συμβαίνει πολλές φορές, το απόθεμα θα είναι τελικά κενό, και το σύστημα θα έχει πρόβλημα, επειδή κανένα νήμα δεν είναι διαθέσιμο για τις εργασίες.

7.9 Πρότυπο Αντικείμενο οργάνου ελέγχου (Monitor Object Pattern)

Ένα όργανο ελέγχου είναι ένα αντικείμενο που προορίζεται για να χρησιμοποιηθεί ακίνδυνα από περισσότερα από ένα νήματα. Το χαρακτηριστικό καθορισμού ενός οργάνου ελέγχου είναι ότι οι μέθοδοί του εκτελούνται με τον αμοιβαίο αποκλεισμό. Δηλαδή σε κάθε χρονική στιγμή, το πολύ ένα νήμα μπορεί να εκτελεί οποιαδήποτε από τις μεθόδους του.
Τα όργανα ελέγχου παρέχουν επίσης έναν μηχανισμό για τα νήματα για να σταματήσουν προσωρινά την αποκλειστική πρόσβαση, προκειμένου να περιμένουν κάποιο όρο που ικανοποιείται, πριν επανακτήσουν την αποκλειστική πρόσβαση και συνεχίσουν την εργασία τους.
Τα όργανα ελέγχου εφευρέθηκαν από τους C.A.R. Hoare και PerBrinch Hansen, και πρώτη φορά εφαρμόστηκαν στην γλώσσα Concurrent PASCAL του Brinch Hansen [43].
Το Πρότυπο Αντικείμενο οργάνου ελέγχου συγχρονίζει την εκτέλεση μεθόδου για να εξασφαλίσει το τρέξιμο μιας μεθόδου μόνο μέσα σε ένα αντικείμενο σε μια στιγμή. Επιτρέπει επίσης στις μεθόδους ενός αντικειμένου να σχεδιάσουν συνεταιριστικά τις ακολουθίες εκτέλεσής τους.
Η δομή του Προτύπου Αντικείμενο οργάνου ελέγχου φαίνεται στο παρακάτω σχήμα:
[41] [42]



Σχήμα 7.9.1 Απεικονίζεται το διάγραμμα κλάσεων του Προτύπου Αντικείμενο οργάνου ελέγχου.

Εδώ φαίνονται οι συμμετάσχοντες στο πρότυπο: [41]

Αντικείμενο οργάνου ελέγχου.
Ένα Αντικείμενο οργάνου ελέγχου εξάγει μια ή περισσότερες μεθόδους σε πελάτες. Για να προστατεύσουν την εσωτερική κατάσταση του αντικειμένου του οργάνου ελέγχου από ανεξέλεγκτες αλλαγές, όλοι οι πελάτες πρέπει να έχουν πρόσβαση στο Αντικείμενο οργάνου ελέγχου, μόνο μέσω αυτών των μεθόδων. Κάθε μέθοδος εκτελεί στο νήμα του πελάτη, που το επικαλείται επειδή ένα Αντικείμενο οργάνου ελέγχου δεν έχει δικό του νήμα ελέγχου.

Συγχρονισμένη μέθοδος.
Οι συγχρονισμένες μέθοδοι εφαρμόζουν τις νήματα-ασφαλείς υπηρεσίες που εξάγονται από ένα Αντικείμενο οργάνου ελέγχου. Μόνο μια συγχρονισμένη μέθοδος μπορεί να εκτελεστεί μέσα σε ένα όργανο ελέγχου σε οποιοδήποτε χρονική στιγμή, ανεξάρτητα από τον αριθμό των νημάτων, που επικαλούνται τις συγχρονισμένες μεθόδους του αντικειμένου ή τον αριθμό των συγχρονισμένων μεθόδων στην κλάση του αντικειμένου.

Κλείδωμα οργάνου ελέγχου.
Κάθε Αντικείμενο οργάνου ελέγχου περιέχει την δικιά του κλειδαριά οργάνου ελέγχου. Οι συγχρονισμένες μέθοδοι χρησιμοποιούν αυτήν την κλειδαριά για να χρησιμοποιηθούν οι επικλήσεις μεθόδων για κάθε αντικείμενο. Κάθε συγχρονισμένη μέθοδος πρέπει να αποκτήσει/απελευθερώσει την κλειδαριά οργάνου ελέγχου, όταν η μέθοδος εισάγει /βγάζει το αντικείμενο, αντίστοιχα. Αυτό το πρωτόκολλο εξασφαλίζει ότι η κλειδαριά οργάνου ελέγχου διατηρείται, όποτε μια μέθοδος εκτελεί διαδικασίες που έχουν πρόσβαση ή τροποποιούν την κατάσταση του αντικειμένου της.

Κατάσταση οργάνου ελέγχου.

Πολλές συγχρονισμένες μέθοδοι που τρέχουν σε χωριστά νήματα μπορούν να χρονοπρογραμματίσουν με τη μεταξύ τους συνεργασία, να σχεδιάσουν τις ακολουθίες εκτέλεσής τους, με την αναμονή και την ειδοποίηση της μιας μέσω της άλλης, μέσω όρων οργάνου ελέγχου, που συνδέονται με το Αντικείμενο οργάνου ελέγχου.

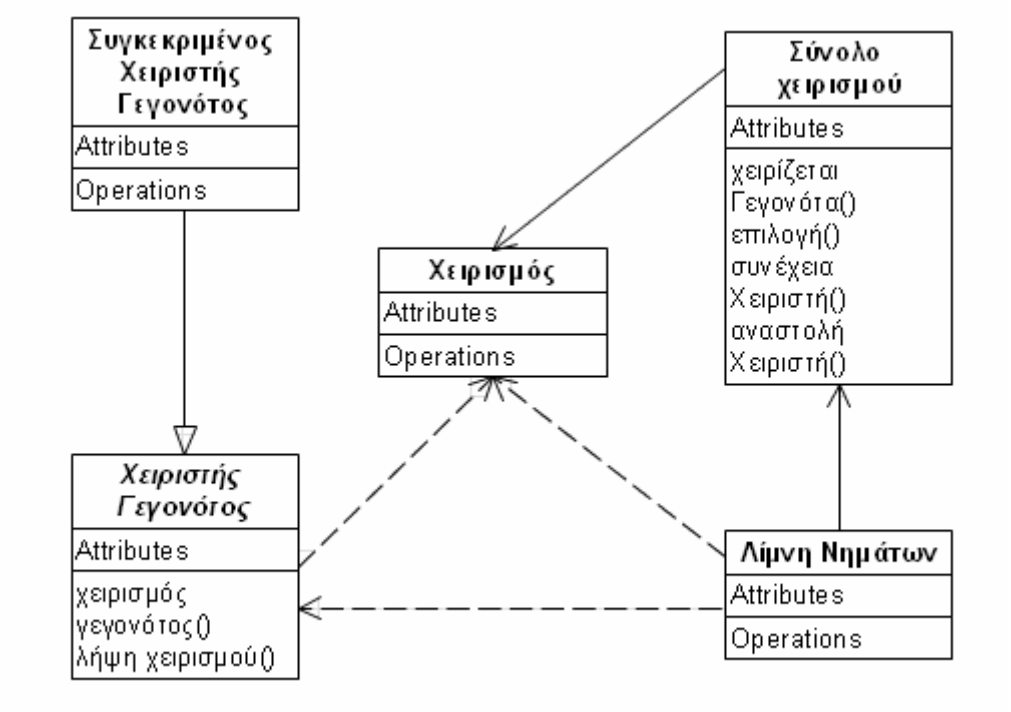
Τα πλεονεκτήματα του αντικειμένου όργανο ελέγχου είναι :
1). Απλοποιεί το συγχρονισμό των μεθόδων που επικαλούνται ταυτόχρονα σε ένα αντικείμενο. Οι πελάτες δεν χρειάζεται να ενδιαφερθούν για τον έλεγχο συγχρονισμού, όταν επικαλούνται οι μέθοδοι σε ένα αντικείμενο όργανο ελέγχου.
2). Οι συγχρονισμένες μέθοδοι μπορούν να σχεδιάσουν μεταξύ τους τη σειρά της εκτέλεσης. Οι συγχρονισμένες μέθοδοι χρησιμοποιούν τους όρους όργανο ελέγχου, για να καθορίσουν τις συνθήκες κάτω από τις οποίες πρέπει να αναστείλουν ή να επαναλάβουν την εκτέλεσή τους.

Το πρότυπο όργανο ελέγχου έχει επίσης μειονεκτήματα [42]:
1). Υπάρχει περιορισμένο ποσοστό ελέγχου.
2). Ο μηχανισμός κλειδώματος και η εφαρμογή είναι στενά συνδεδεμένα
3). Μπορεί να εμφανιστεί πρόβλημα κλειδώματος ενός αντικείμενο όργανο ελέγχου, όταν το αντικείμενο όργανο ελέγχου τοποθετείται μέσα σε ένα άλλο αντικείμενο όργανο ελέγχου.

7.10 Πρότυπο Ηγέτη-Ακολούθων (Leaders-Followers Pattern)

Το Πρότυπο Ηγέτη-Ακολούθων παρέχει ένα αποτελεσματικό πρότυπο συγχρονισμού. Η δομή του περιέχει ένα απόθεμα νημάτων-λίμνη νημάτων που μοιράζεται ένα σύνολο γεγονότων, αποπολυπλέκοντας με τη σειρά τα γεγονότα που φθάνουν και αποστέλλοντας συγχρονισμένα τα γεγονότα στις υπηρεσίες εφαρμογής, οι οποίες τα επεξεργάζονται [25].

Το παρακάτω σχήμα δείχνει την δομή των συμμετεχόντων στο Πρότυπο Ηγέτη-Ακολούθων. [24] [44].



Σχήμα 7.10.1 Απεικόνιση του διαγράμματος κλάσεων του Πρότυπου Ηγέτη-Ακολούθων.

Οι συμμετέχοντες στο πρότυπο αυτό είναι:

1). Σύνολο Χειρισμού.
Ένα σύνολο Χειρισμού είναι μια συλλογή των Χειρισμών που μπορεί να χρησιμοποιηθεί για να περιμένει να εμφανιστούν τα γεγονότα στο σύνολο. Ένα σύνολο Χειριστών επιστρέφει σε αυτόν που το κάλεσε, όταν είναι δυνατό να αρχίσει μια λειτουργία σε ένα Χειρισμό στο σύνολο χωρίς να φραχτεί η λειτουργία.

2). Χειρισμός.
Προσδιορίζει τους πόρους, [45] όπως συνδέσεις υποδοχών (socket connections) ή τα ανοικτά αρχεία (open files), που εφαρμόζονται συχνά και ρυθμίζονται από ένα λειτουργικό σύστημα.

3). Χειριστής γεγονότος.
Ένας χειριστής γεγονότος προσδιορίζει μια διεπαφή που αποτελείται από μια ή περισσότερες μεθόδους γάντζων-hook [26] [27]. Αυτές οι μέθοδοι αναπαριστούν το σύνολο διαθέσιμων διαδικασιών ή συγκεκριμένα γεγονότα που εμφανίζονται στο Χειρισμό που σχετίζονται με έναν χειριστή γεγονότος.

4). Συγκεκριμένος χειριστής γεγονότος.
Οι συγκεκριμένοι χειριστές γεγονότος εξειδικεύονται από τον χειριστή γεγονότος και εφαρμόζουν μια συγκεκριμένη υπηρεσία που προσφέρει η εφαρμογή. Κάθε συγκεκριμένος χειριστής γεγονότος συνδέεται με ένα Χειρισμό, που προσδιορίζει αυτήν την υπηρεσία μέσα στην εφαρμογή. Επιπλέον, οι συγκεκριμένοι χειριστές γεγονότος εφαρμόζουν τη μέθοδο hook, που είναι αρμόδια για την επεξεργασία γεγονότων που λαμβάνονται μέσω του Χειρισμού τους.

5). ΛίμνηΝημάτων [45].
Τα νήματα διαδραματίζουν διάφορους ρόλους στο πρότυπο. Ένα ή περισσότερα νήματα ακολούθων περιμένουν στη σειρά έναν συγχρονιστή, περιμένοντας να γίνουν το νήμα ηγέτη. Ένα από αυτά τα νήματα επιλέγεται να είναι ο ηγέτης, το οποίο περιμένει ένα γεγονός να εμφανιστεί σε οποιαδήποτε Χειρισμό στο σύνολο Χειρισμού του. Όταν ένα γεγονός εμφανίζεται, το παρόν νήμα ηγέτη προάγει ένα νήμα οπαδών για να γίνει ο νέος ηγέτης. Ο αρχικός ηγέτης τότε ταυτόχρονα έχει το ρόλο ενός νήματος επεξεργασίας, το οποίο αποπολυπλέκει το γεγονός από Σύνολο Χειρισμού στο συσχετιζόμενο Χειριστή γεγονότος και χειρίζεται το γεγονός. Αφού τελειώσει ένα νήμα επεξεργασίας, επιστρέφει παίζοντας το ρόλο ενός νήματος οπαδού και περιμένει πάλι να γίνει νήμα ηγέτη.

Το Πρότυπο Ηγέτη-Ακολούθων έχει τα εξής πλεονεκτήματα:
1). Ελαχιστοποιεί το σφάλμα κλειδώματος με την μη ανταλλαγή των δεδομένων μεταξύ των νημάτων. Επομένως αυξάνει η απόδοση.
2). Το πρότυπο απλοποιεί τον προγραμματισμό συγχρονισμένων μοντέλων, όπου πολλά νήματα μπορούν να λάβουν αιτήσεις, απαντήσεις επεξεργασίας, και να αποπολυπλέξουν τις συνδέσεις χρησιμοποιώντας ένα κοινό σύνολο Χειρισμού.

7.11 Πρότυπο Διπλοελεγμένου κλειδώματος (Double Checked Locking Pattern)

Στην τεχνολογία λογισμικού, το διπλοελεγγμένο κλείδωμα είναι ένα σχεδιαστικό πρότυπο λογισμικού γνωστό επίσης ως "διπλο-ελεγμένο κλείδωμα οπτικοποίησης (double-checked locking optimization) ". Το πρότυπο σχεδιάζεται για να μειώσει τα έξοδα της απόκτησης ενός κλειδώματος, εξετάζοντας πρώτα το κριτήριο κλειδώματος. Μόνο εάν αυτό πετύχει, το πραγματικό κλείδωμα προχωρά. Το πρότυπο, όταν εφαρμόζεται σε μερικούς συνδυασμούς γλωσσών/υλικού (language/hardware), μπορεί να είναι ασφαλές [29]. Μπορεί επομένως να θεωρηθεί μερικές φορές ως αντι-πρότυπο (anti-pattern). Χρησιμοποιείται χαρακτηριστικά για να μειώσει τα έξοδα κλειδώματος κατά την εφαρμογή της "lazy έναρξης" (lazy initialization) σε ένα πολυνηματικό περιβάλλον, ειδικά ως τμήμα του προτύπου Singleton [28].

Το Πρότυπο Διπλοελεγμένου κλειδώματος χρησιμοποιείται, όταν μια εφαρμογή έχει τα ακόλουθα χαρακτηριστικά:
1). Η εφαρμογή έχει ένα ή περισσότερα κρίσιμα τμήματα του κώδικα που πρέπει να εκτελεστούν διαδοχικά.

2). Πολλά νήματα μπορεί να προσπαθήσουν να εκτελέσουν το κρίσιμο τμήμα ταυτόχρονα .

3). Το κρίσιμο τμήμα του κώδικα εκτελείται ακριβώς μια φορά.

4). Η απόκτηση μιας κλειδαριάς σε κάθε πρόσβαση στο κρίσιμο τμήμα προκαλεί υπερβολικά έξοδα.
.
5). Είναι δυνατό να χρησιμοποιηθεί μια ελαφριά, όμως αξιόπιστη, υποθετική δοκιμή (conditional test) αντί μιας κλειδαριάς.

Η δομή και οι συμμετέχοντες στο πρότυπο μπορούν να αναπαρασταθούν πιο εύκολα χρησιμοποιώντας ψευδοκώδικα. Στο επόμενο κομμάτι κώδικα φαίνονται οι συμμετέχοντες στο πρότυπο [46].

```
if (Flag == FALSE)
  {
    Mutex.acquire ();
    if (Flag == FALSE)
      {
        critical section;
        Flag = TRUE;
      }
    Mutex.release ();
  }
```

1). Το κρίσιμο τμήμα (critical section) περιέχει τον κώδικα που εκτελείται ακριβώς μια φορά.

2). Mutex είναι μια κλειδαριά που κάνει διαθέσιμη την πρόσβαση στο κρίσιμο τμήμα του κώδικα.

3). Η Σημαία (Flag), δείχνει εάν το κρίσιμο τμήμα έχει εκτελεστεί ήδη.

4). Νήμα Εφαρμογής (Application thread) είναι το νήμα που προσπαθεί να εκτελέσει το κρίσιμο τμήμα.

Το Νήμα Εφαρμογής ελέγχει αρχικά, εάν η σημαία έχει τεθεί ως βελτιστοποίηση για την συγκεκριμένη περίπτωση. Εάν δεν έχει τεθεί, το Mutex αποκτιέται. Κρατώντας την κλειδαριά, το Νήμα Εφαρμογής ελέγχει πάλι αν η σημαία έχει τεθεί, εκτελώντας το κρίσιμο τμήμα μια φορά και βάζει την σημαία σε αληθινό (true). Τέλος, το Νήμα Εφαρμογής απελευθερώνει την κλειδαριά.
Με την προσθήκη ενός Mutex και ενός δεύτερου ελέγχου, η κανονική εφαρμογή Singleton μπορεί να είναι νήμα-ασφαλές (threadsafe) χωρίς να υποστεί οποιαδήποτε έξοδα κλειδώματος, αφότου έχει εμφανιστεί η έναρξη.

Το πρότυπο αυτό έχει ορισμένα πλεονεκτήματα:
1). Ελαχιστοποιημένο κλείδωμα: Εκτελούνται δύο έλεγχοι σημαίας. Μόλις τεθεί η σημαία, ο πρώτος έλεγχος εξασφαλίζει ότι οι επόμενες προσβάσεις δεν απαιτούν κανένα κλείδωμα.

2). Ο δεύτερος έλεγχος της σημαίας εξασφαλίζει ότι το γεγονός εκτελείται μόνο μια φορά.

Υπάρχει επίσης ένα μειονέκτημα [46]. Υπάρχει ένα λεπτό ζήτημα φορητότητας (portability) που μπορεί να οδηγήσει σε ολέθρια σφάλματα, εάν το πρότυπο χρησιμοποιείται στο λογισμικό σε πλατφόρμες υλικού (hardware), που έχουν

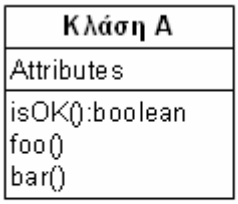
nonatomic pointer ή ακέραια σημασιολογία ανάθεσης (integral assignment semantics).

7.12 Φρουρημένη αναβολή-αναστολή (Guarded Suspension Pattern)

Στον συγχρονισμένο προγραμματισμό, η φρουρημένη αναστολή είναι ένα σχεδιαστικό πρότυπο λογισμικού για τη διαχείριση διαδικασιών που απαιτούν και ένα κλείδωμα να αποκτηθεί και μια προϋπόθεση να ικανοποιηθεί προτού να μπορέσει να εκτελεστεί η λειτουργία. Το πρότυπο φρουρημένης αναστολής εφαρμόζεται χαρακτηριστικά στις κλήσεις μεθόδου στα αντικειμενοστρεφή προγράμματα, και περιλαμβάνει την αναστολή της κλήσης μεθόδου, και το καλώντας νήμα, έως ότου ικανοποιηθεί η προϋπόθεση (που ενεργεί ως φρουρά) [30].
Λόγω στη φύση του, το πρότυπο φρουρημένης αναστολής πρέπει μόνο να χρησιμοποιηθεί όταν ξέρει ο υπεύθυνος ανάπτυξης ότι μια κλήση μεθόδου θα ανασταλεί για μια πεπερασμένη χρονική περίοδο. Εάν μια κλήση μεθόδου αναστέλλεται για υπερβολικά μεγάλο χρονικό διάστημα, κατόπιν το γενικό πρόγραμμα που σχετίζεται με την κλήση μεθόδου θα επιβραδυνθεί. Εάν οι υπεύθυνοι για την ανάπτυξη ξέρουν ότι η αναστολή κλήσης μεθόδου θα είναι αόριστη, τότε θα ήταν καλύτερα να χρησιμοποιηθεί το πρότυπο εμπόδισης (Balking Pattern) [30].

Οι μέθοδοι μιας κλάσης πρέπει να συγχρονιστούν για να επιτρέψουν ασφαλείς συγχρονισμένες κλήσεις. Ένα αντικείμενο μπορεί να είναι σε μια κατάσταση που το καθιστά αδύνατο για μια από τις συγχρονισμένες μεθόδους του να εκτελεστεί. Για να φύγει το αντικείμενο από αυτήν την κατάσταση, πρέπει να εκτελεστεί μια κλήση σε μια από τις άλλες συγχρονισμένες μεθόδους του αντικειμένου. Εάν μια κλήση στην πρώτη μέθοδο επιτρέπεται να προχωρήσει ενώ το αντικείμενο είναι σε εκείνη τη κατάσταση, δημιουργείται αδιέξοδο και δεν θα ολοκληρώσει ποτέ. Οι κλήσεις στην μεταβαλλόμενης κατάστασης μέθοδο, που επιτρέπει στην πρώτη μέθοδο να ολοκληρωθεί, θα πρέπει να περιμένουν έως ότου ολοκληρωθεί, το οποίο δεν θα συμβεί ποτέ.

Στο παρακάτω σχήμα παρουσιάζεται μια κλάση που ονομάζεται Κλάση Α. Έχει δύο συγχρονισμένες μεθόδους τις foo και bar. Υπάρχει μια κατάσταση (exceptional), στην οποία μπορούν να εισέλθουν τα αντικείμενα Α. Ενώ ένα αντικείμενο Α είναι σε αυτήν την κατάσταση, η μέθοδος isOK της επιστρέφει false, αλλιώς επιστρέφει true. Όταν ένα αντικείμενο Α είναι σε αυτήν την κατάσταση, μια κλήση στη μέθοδο bar, μπορεί να το βγάλει από εκείνη την κατάσταση. Δεν υπάρχει κανένας τρόπος να βγει από την κατάσταση (exceptional), εκτός από μια κλήση στην bar. Η λήψη ενός αντικειμένου Α από την (exceptional) κατάστασή της, είναι μια παρενέργεια του κύριου σκοπού της μεθόδου bar, έτσι δεν είναι αποδεκτό να κληθεί η μέθοδος bar, για να πάρει μόνο ένα αντικείμενο Α από την (exceptional) κατάστασή του. Παρακάτω φαίνεται η κλάση Μη Φρουρημένης αναστολής. [33]



Σχήμα 7.12.1 Απεικόνιση της κλάσης μη φρουρημένης αναστολής.

Μια κλήση στη μέθοδο foo ενός αντικειμένου Α δεν μπορεί να ολοκληρωθεί, εάν το αντικείμενο Α είναι στην (exceptional) κατάστασή του. Εάν αυτό συμβεί, επειδή οι μέθοδοι foo και bar είναι συγχρονισμένες, οι επόμενες κλήσεις στις μεθόδους foo και bar του αντικειμένου Α δεν θα εκτελεστούν, έως ότου η εκκρεμής κλήση επιστρέψει στο foo. Η κλήση στο foo δεν θα επιστρέψει, έως ότου μια κλήση στην bar πάρει το αντικείμενο Α από την (exceptional) κατάστασή της.
Ο σκοπός του προτύπου είναι να αποφευχθεί η κατάσταση αδιεξόδου, που μπορεί να εμφανιστεί, όταν ένα νήμα είναι έτοιμο να εκτελέσει τη συγχρονισμένη μέθοδο ενός αντικειμένου και η κατάσταση του αντικειμένου αποτρέπει τη μέθοδο να ολοκληρωθεί. Εάν μια κλήση μεθόδου εμφανίζεται όταν ένα αντικείμενο είναι σε μια κατάσταση που αποτρέπει τη μέθοδο να ολοκληρωθεί, το πρότυπο αναστέλλει το νήμα έως ότου το αντικείμενο είναι σε μια κατάσταση που επιτρέπει στη μέθοδο να ολοκληρωθεί.
Το πρότυπο φρουρημένης αναστολής εφαρμόζεται χρησιμοποιώντας τις μεθόδους αναμονή (wait) και ειδοποίηση (notify) όπως φαίνεται στο σχήμα 7.12.2.

```
class Widget {
    synchronized void foo(){
        while (!isOK()) {
            wait();
        }
        ...
    }

    synchronized void bar(x int) {
        ...
        notify();
    }
}
```

Σχήμα 7.12.2 Απεικονίζεται πως χρησιμοποιούνται οι μέθοδοι αναμονή (wait) και ειδοποίηση (notify).

Μια μέθοδος όπως η foo πρέπει να ικανοποιήσει τις προϋποθέσεις προτού να αρχίσει. Το πρώτο πράγμα που κάνει μια τέτοια μέθοδος είναι να δοκιμάσει τις προϋποθέσεις της σε έναν βρόχο. Αν οι προϋποθέσεις είναι ψεύτικες (false), καλεί τη μέθοδο αναμονής.
Κάθε κλάση κληρονομεί τη μέθοδο αναμονής από την κλάση αντικειμένου. Όταν ένα νήμα καλεί τη μέθοδο αναμονής του αντικειμένου, η μέθοδος αναμονής προκαλεί το νήμα να ελευθερώσει τη κλειδαριά συγχρονισμού που κρατά στο αντικείμενο. Η

μέθοδος περιμένει έως ότου ειδοποιηθεί ότι μπορεί να επιστρέψει. Τότε, μόλις είναι σε θέση ένα νήμα να συλλάβει εκ νέου το κλείδωμα, η μέθοδος αναμονής επιστρέφει.
Όταν η μέθοδος αναμονής επιστρέφει, ο έλεγχος επιστρέφει στην κορυφή του βρόχου, ο οποίος εξετάζει τις προϋποθέσεις πάλι. Ο λόγος για να ελεχθούν οι προϋποθέσεις σε έναν βρόχο, είναι ότι μεταξύ του χρόνου που το νήμα προσπαθεί αρχικά να συλλάβει εκ νέου το κλείδωμα συγχρονισμού και του χρόνου που το συλλαμβάνει, ένα άλλο νήμα μπορεί να είχε καταστήσει τις προϋποθέσεις ψεύτικες-ψευδείς (false).
Η κλήση στη μέθοδο αναμονής δηλώνεται ότι πρέπει να επιστρέψει όταν καλεί μια άλλη μέθοδος, όπως η bar, τη μέθοδο ειδοποίηση του αντικειμένου. Τέτοιες μέθοδοι καλούν τη μέθοδο ειδοποίηση αφότου έχουν αλλάξει την κατάσταση του αντικειμένου με έναν τρόπο που μπορεί να ικανοποιήσει τις προϋποθέσεις. Η μέθοδος ειδοποίηση είναι μια άλλη μέθοδος που όλες οι κλάσεις κληρονομούν από την κλάση αντικειμένου. Αυτό που κάνει η μέθοδος ειδοποίηση είναι να ειδοποιεί ένα άλλο νήμα που περιμένει για τη μέθοδο αναμονής να επιστρέψει, ότι πρέπει να επιστρέψει.
Εάν περισσότερα από ένα νήματα περιμένουν, η μέθοδος ειδοποίηση επιλέγει να ειδοποιήσει ένα αυθαίρετα. Η αυθαίρετη επιλογή λειτουργεί καλά στις περισσότερες καταστάσεις. Δεν λειτουργεί καλά για τα αντικείμενα που έχουν μεθόδους με διαφορετικές προϋποθέσεις.

Αναφορές:

[1]: Cleidson R. B. de Souza and Roberto S. Silva Filho, *Checking Java Concurrency Design Patterns Using Bandera*, CiteSeer Computer and Information Science Publications collection.
[2]: Douglas C. Schmidt, Stephen D. Huston, (2001), *C++ Network Programming, Volume I: Mastering Complexity with ACE and Patterns*, Addison-Wesley Professional.
[3]: http://en.wikipedia.org/wiki/Concurrency_pattern
[4]: http://en.wikipedia.org/wiki/Active_Object
[5]: http://en.wikipedia.org/wiki/Scheduler_pattern
[6]: Carlos A. Cunha, João L. Sobral and Miguel P. Monteiro, (2006), *Reusable Aspect-Oriented Implementations of Concurrency Patterns and Mechanisms*, Aspect-oriented software development, Proceedings of the 5th international conference on Aspect-oriented software development, Bonn, Germany, SESSION: Evaluation and metrics,2006, pp. 134 – 145.
[7]: http://en.wikipedia.org/wiki/Read/write_lock_pattern
[8]: http://paulbridger.net/read_write_lock
[9]: Marek Biskup, (2006), *Design Patterns for Concurrent Objects*, Warsaw University, CERN School of Computing.
[10]: http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/locks/ReadWriteLock.html
[11]: D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, (2000), *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects.* Addison Wesley,2000.
[12]: Douglas C. Schmidt, (1995), *Using Design Patterns to Develop Reusable Object-Oriented Communication Software* , COMMUNICATIONS OF THE ACM October 1995/Vol. 38, No. 10.
[13]: Christopher Gill, Guandong Wang, Zhenning Hu, Zhenghui Xie, Slides: *The Reactor Pattern*, E61 CS 342S: Object-Oriented Software Development Laboratory Department of Computer Science and Engineering Washington University, St. Louis.
[14]: Lee Provoost, (2005), Slides: *The Reactor Pattern*, Department of Information & Computing Sciences University of Utrecht 28 September 2005.
[15]: *More C++ Gems*, (2000), Robert Martin, Cambridge University Press, 2000
[16]: Venkita Subramonian & Christopher D. Gill, (2006), Slides: *Thread-Specific Storage (TSS)* E81 CSE 532S: Advanced Multi-Paradigm Software Development.
[17]: Slides: *POSA2: Thread-Specific Storage Design Pattern*, Distributed Real-Time Systems (TI-DRTS), Version: 11.11.2008.
[18]: Slides: Lecture 10: Java Threads Programming 3 NCCU Fall 2005 Nov. 29,2005
[19]: http://en.wikipedia.org/wiki/Thread-safe

[20]: http://en.wikipedia.org/wiki/Thread_pool_pattern
[21]: http://orshalic.wikispaces.com/Thread+pool+pattern
[22]: http://www.ibm.com/developerworks/library/j-jtp0730.html
[23]: Mark Grand, (2002), *PATTERNS IN JAVA VOLUME 3, Java Enterprise Design Patterns*, Wiley Computer Publishing.
[24]: David F P ZHOU, (2005), Slides: *Leader/Follower Pattern*, CSI 5380, System and Architecture for Electronic Commerce, February 9, 2005.
[25]: Nikolai Cieslak and Dennis Eder, *Architecture Pattern The Reactor*, hasso-Plattner-Instityte for Software Systems Engineering.
[26] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns, (1995), *Elements of Reusable Object-Oriented Software*. Reading, MA: Addison- Wesley, 1995.
[27] W. Pree, (1995), *Design Patterns for Object-oriented Software Development.* Reading, MA: Addison-Wesley, 1995.
[28]: Scott Meyers and Andrei Alexandrescu, (2004), *C++ and the Perils of Double-Checked Locking*, September 2004. *Dr. Dobbs Journal* in the July (Part I) and August (Part II), 2004, issues.
[29]: http://en.wikipedia.org/wiki/Double-checked_locking
[30]: http://en.wikipedia.org/wiki/Guarded_suspension
[31].http://www.titu.jyu.fi/modpa/Patterns/pattern-ActiveObject.html
[32]. http://benpryor.com/blog/2006/03/08/implementing-active-objects-with-java-dynamic-proxies/
[33]. Mark Grand, (1998), *Patterns in Java, Volume 1—A Catalog of Reusable Design Patterns Illustrated with UML, Second Edition*. John Wiley and Sons.
[34]. Mark Grand, (2001), Slides: *Multi- Threading Design Patterns*, ClickBlocks LLC 1998, 2002, 2003
[35]. Jan Lönnberg, (2005), Slides: *Reactor pattern T-106.402 Concurrent programming*, Department of Computer Science, Helsinki University of Technology 15th November 2005
[36]. Arundhati Kogekar, Aniruddha Gokhale, (2006), *Performance Evaluation of the Reactor Pattern Using the OMNeT++ Simulator*, 44th ACM Southeast Conference Melbourne, Florida USA March 10-12, 2006.
[37]. R. Greg Lavender Douglas C. Schmidt, (1996), *Active Object, An Object Behavioral Pattern for Concurrent Programming*, Pattern languages of program design 2 ,1996, pp.483 – 499.
[38]. Douglas C. Schmidt, Nat Pryce, and Timothy H. Harrison, (1997), "*ThreadSpecific Storage for C/C++ - An Object Behavioral Pattern for Accessing perThread State Efficiently" Collected papers from the PLoP '97 and EuroPLoP '97 Conference*, Technical Report #wucs-97-34, Dept. of Computer Science, Washington University Department of Computer Science, September 1997.
[39].http://www.codeproject.com/KB/threads/threadlibrary.aspx
[40]. Douglas C. Schmidt, *Monitor Object An Object Behavioral Pattern for Concurrent Programming*, Department of Computer Science Washington University, St. Louis.
[41].
[42]. Huib van den Brink, (2005), *Monitor Object Concurrency Pattern*, Center for Software Technology, Utrecht University October 14, 2005. http://www.cs.uu.nl/groups/ST/
[43]. http://en.wikipedia.org/wiki/Monitor_(synchronization)
[44]. Lennart Kats, (2005), *Scalable Webserver Threading Using The Leader/Followers Pattern*
[45]. Douglas C. Schmidt and Carlos O'Ryan, (2000), *Leader/Followers A Design Pattern for Efficient Multi-threaded I/O Demultiplexing and Dispatching*, University of Washington. Proceedings of the 7th Pattern Languages of Programs, Conference in Allerton Park, Illinois, August 2000.
[46]. Douglas C. Schmidt and Tim Harrison, *Double-Checked Locking An Optimization Pattern for Efficiently Initializing and Accessing Thread-safe Objects* . This paper appeared in a chapter in the book "Pattern Languages of Program Design 3" ISBN, edited by Robert Martin, Frank Buschmann, and Dirke Riehle published by Addison-Wesley, 1997.

8. ΑΡΧΙΤΕΚΤΟΝΙΚΑ ΠΡΟΤΥΠΑ
ARCHITECTURAL PATTERNS

8.1 Σύντομη αναφορά των αρχιτεκτονικών προτύπων

Τα αρχιτεκτονικά πρότυπα είναι πρότυπα λογισμικού που προσφέρουν λύσεις στα αρχιτεκτονικά προβλήματα στην τεχνολογία λογισμικού. Ένα αρχιτεκτονικό πρότυπο εκφράζει ένα θεμελιώδες δομικό σχήμα οργάνωσης για ένα σύστημα λογισμικού, το οποίο αποτελείται από υποσυστήματα, τις ευθύνες και τις αμοιβαίες σχέσεις τους. Σε σύγκριση με τα σχεδιαστικά πρότυπα (design patterns), τα αρχιτεκτονικά πρότυπα είναι μεγαλύτερα στην κλίμακα. Ακόμα κι αν ένα αρχιτεκτονικό πρότυπο μεταβιβάζει μια εικόνα ενός συστήματος, δεν είναι μια αρχιτεκτονική υπό αυτήν τη μορφή. Ένα αρχιτεκτονικό πρότυπο είναι μια έννοια που συλλαμβάνει τα απαραίτητα στοιχεία μιας αρχιτεκτονικής λογισμικού. Οι αμέτρητες διαφορετικές αρχιτεκτονικές μπορούν να εφαρμόσουν το ίδιο πρότυπο και με αυτόν τον τρόπο να έχουν τα ίδια χαρακτηριστικά.
Μια από τις σημαντικότερες πτυχές των αρχιτεκτονικών προτύπων είναι ότι ενσωματώνουν διαφορετικές ποιοτικές ιδιότητες. Παραδείγματος χάριν, μερικά πρότυπα αντιπροσωπεύουν τις λύσεις στα προβλήματα απόδοσης και άλλα μπορούν να χρησιμοποιηθούν επιτυχώς στα υψηλής διαθεσιμότητας (high-availability) συστήματα. Στην πρώρη φάση προτύπου, ένας αρχιτέκτονας λογισμικού κάνει μια επιλογή, ποιο αρχιτεκτονικό πρότυπο ή πρότυπα παρέχουν καλύτερα τις επιθυμητές ιδιότητες του συστήματος [1] [2] [3].
Ένα αρχιτεκτονικό πρότυπο εκφράζει ένα θεμελιώδες δομικό σχήμα οργάνωσης για το σύστημα λογισμικού. Παρέχει ένα σύνολο προκαθορισμένων υποσυστημάτων, προσδιορίζει τις ευθύνες τους και περιλαμβάνει τους κανόνες και τις οδηγίες για την οργάνωση της σχέσης μεταξύ τους.

Τα παραδείγματα των αρχιτεκτονικών προτύπων περιλαμβάνουν τα εξής:
- Παρουσίαση- Αφαίρεση -Έλεγχος (Presentation- Abstraction- Control)
- Σωλήνωση (Pipeline)
- Υπονοούμενη επίκληση (Implicit invocation)
- Πρότυπο Πίνακα (Blackboard Pattern)
- (Peer to peer)
- Προσανατολισμένη προς τις υπηρεσίες αρχιτεκτονική (Service-oriented architecture)
- Γυμνά Αντικείμενα (Naked objects)
- Μοντέλο-Όψη-Ελεγκτής (Model- View-Controller)

Το πρότυπο παρουσίαση-αφαίρεση-ελέγχου (PAC pattern) καθορίζει μια δομή για τα διαλογικά συστήματα λογισμικού (interactive software systems) υπό μορφή ιεραρχίας των συνεργαζόμενων πρακτόρων. Κάθε πράκτορας είναι αρμόδιος για μια συγκεκριμένη πτυχή της λειτουργίας της εφαρμογής και αποτελείται από τρία συστατικά: παρουσίαση, αφαίρεση και έλεγχος. Αυτή η υποδιαίρεση χωρίζει τις πτυχές αλληλεπίδρασης ανθρώπου-υπολογιστή του πράκτορα από το λειτουργικό πυρήνα της και της επικοινωνίας της με άλλους πράκτορες.
Το πρότυπο μοντέλο-όψη-ελεγκτής (MVC pattern) διαιρεί μια διαλογική εφαρμογή σε τρία συστατικά. Το μοντέλο περιέχει τη λειτουργία πυρήνα και τα δεδομένα. Η όψη παρουσιάζει τις πληροφορίες στο χρήστη. Οι ελεγκτές χειρίζονται την είσοδο

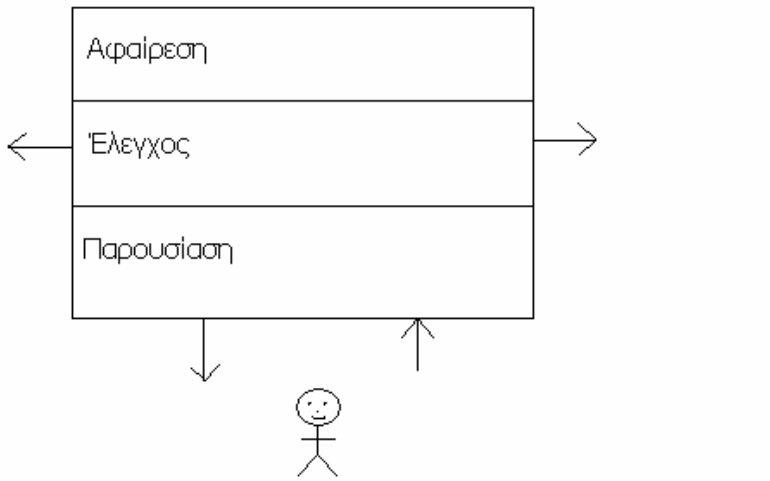
δεδομένων των χρηστών. Οι όψεις και οι ελεγκτές περιλαμβάνουν μαζί το ενδιάμεσο με τον χρήστη. Ένας μηχανισμός αλλαγή-διάδοσης (change propagation mechanism) εξασφαλίζει συνέπεια μεταξύ του ενδιάμεσου με τον χρήστη και του μοντέλου.
Το πρότυπο σωλήνων και φίλτρων παρέχει μια δομή για τα συστήματα που επεξεργάζονται μια ροή των δεδομένων. Κάθε βήμα επεξεργασίας ενθυλακώνεται σε ένα τμήμα φίλτρων. Τα δεδομένα περνούν μέσω των σωλήνων μεταξύ των παρακείμενων φίλτρων. Ο επανασυνδιασμός των φίλτρων επιτρέπει να χτιστούν οικογένειες των σχετικών συστημάτων.
Το πρότυπο πινάκων (Blackboard pattern) είναι χρήσιμο για προβλήματα για τα οποία καμία αιτιοκρατική-ντετερμινιστική στρατηγική λύσης δεν είναι γνωστή. Στον πίνακα διάφορα εξειδικευμένα υποσυστήματα συγκεντρώνουν τη γνώση τους για να χτίσουν μια ενδεχομένως μερική ή κατά προσέγγιση λύση [53] [54] [59].

Η επιλογή ενός αρχιτεκτονικού προτύπου πρέπει να οδηγηθεί από τις ιδιότητες της εφαρμογής που υπάρχει. Η επιλογή προτύπων πρέπει να επηρεάζεται από τις μη λειτουργικές απαιτήσεις των εφαρμογών, όπως η δυνατότητα να αλλάζει η εφαρμογή ή η αξιοπιστία. Είναι επίσης χρήσιμο να ερευνηθούν διάφορες εναλλακτικές λύσεις πριν αποφασιστεί σχετικά με ένα συγκεκριμένο αρχιτεκτονικό πρότυπο. Παραδείγματος χάριν, το πρότυπο παρουσίαση-αφαίρεση-έλεγχος (PAC) και το πρότυπο μοντέλο-όψη-ελεγκτής (MVC) και τα δύο ανήκουν στις διαλογικές εφαρμογές [1].
Διαφορετικά αρχιτεκτονικά πρότυπα επιφέρουν διαφορετικές συνέπειες, ακόμα κι αν εξετάζουν τα ίδια ή πολύ παρόμοια προβλήματα. Παραδείγματος χάριν, μια MVC αρχιτεκτονική είναι συνήθως αποδοτικότερη από μια αρχιτεκτονική PAC. Όμως, η PAC υποστηρίζει τα πολλαπλά καθήκοντα (multitasking) και task-specific ενδιάμεσα με τον χρήστη καλύτερα από την MVC.
Τα περισσότερα συστήματα λογισμικού, εντούτοις, δεν μπορούν να κατασκευαστούν σύμφωνα με ένα αρχιτεκτονικό πρότυπο. Πρέπει να υποστηρίξουν διάφορες απαιτήσεις συστημάτων που μπορούν μόνο να καλυφτούν από τα διαφορετικά αρχιτεκτονικά πρότυπα [1].

8.2 Πρότυπο Παρουσίαση-αφαίρεση-έλεγχος (Presentation-Abstraction-Control Pattern)

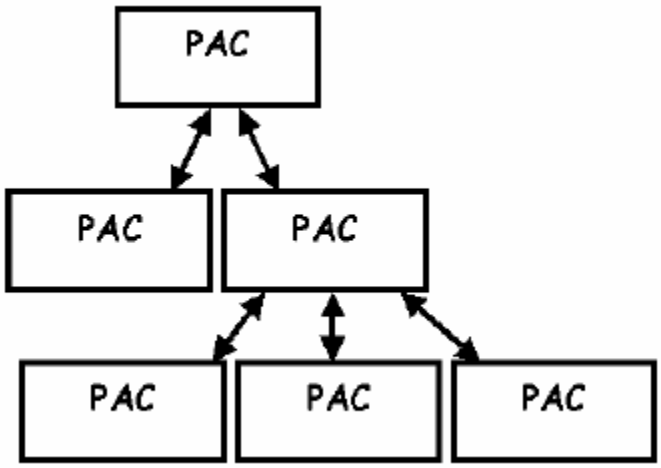
Το πρότυπο Παρουσίαση-αφαίρεση-έλεγχος (PAC) είναι ένα αντικειμενοστρεφές σχεδιαστικό πρότυπο που διαχωρίζει τις ανησυχίες ενός συστήματος, σπάζοντας το σε χαλαρά συνδεδεμένους πράκτορες, κάθε ένας αρμόδιος για μια στοιχειώδη εργασία (σχήμα 8.2.1). Το αρχιτεκτονικό πρότυπο PAC καθορίζει μια δομή για τα διαλογικά συστήματα λογισμικού (interactive software systems) υπό μορφή ιεραρχίας των συνεργαζόμενων πρακτόρων [11].
Ένας πράκτορας δείχνει ένα τμήμα επεξεργασίας πληροφοριών που περιλαμβάνει δέκτες και αποστολείς γεγονότων, δομές δεδομένων για να διατηρήσει τη κατάσταση, και έναν επεξεργαστή που επεξεργάζεται τα εισερχόμενα γεγονότα, ενημερώνει την κατάστασή του και μπορεί να παράγει νέα γεγονότα .
Κάθε πράκτορας έχει εσωτερικά τα συστατικά που εξυπηρετούν τη μια από τρεις στοιχειώδεις εργασίες. Εκείνα τα συστατικά που αφαιρούν τη λειτουργία πυρήνα και τα δεδομένα που χρησιμοποιούνται από τον πράκτορα (αφαίρεση), εκείνα τα συστατικά που παρέχουν την πρόσβαση στον πράκτορα (παρουσίαση) και εκείνα τα συστατικά που ελέγχουν τις αλληλεπιδράσεις μεταξύ των στρωμάτων αφαίρεσης και

παρουσίασης (έλεγχος) [6] [8] [9]. Πρέπει να σημειωθεί ότι τα πρότυπα PAC είναι παρόμοια με το πρότυπο μοντέλο- όψη-ελεγκτής (MVC) δεδομένου ότι κρύβει την εσωτερική εφαρμογή των λογικών λειτουργιών του συστήματος από το ενδιάμεσο με τον χρήστη. Στο PAC, ο διαχωρισμός μεταξύ του ενδιάμεσου με τον χρήστη και τη λειτουργία των εσωτερικών της εφαρμογής γίνεται με τη χρησιμοποίηση του τμήματος ελέγχου για να περάσουν τα μηνύματα μεταξύ των δύο στρωμάτων [5].



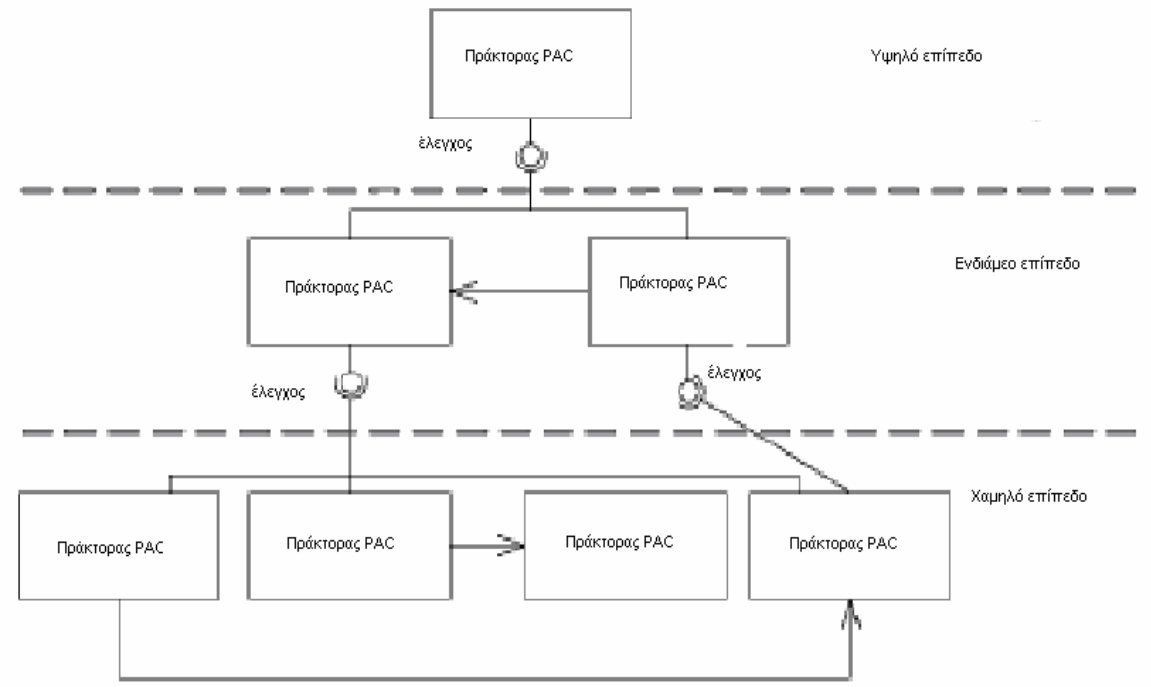
Σχήμα 8.2.1 Απεικονίζεται το πρότυπο παρουσίαση-αφαίρεση-έλεγχος.

Όπως αναφέρθηκε και προηγουμένως καθορίζει μια δομή για τα διαλογικά (interactive) συστήματα λογισμικού υπό μορφή ιεραρχίας των συνεργαζόμενων πρακτόρων. Κάθε πράκτορας είναι αρμόδιος για μια συγκεκριμένη πτυχή της λειτουργίας της εφαρμογής, και αποτελείται από τρία συστατικά: παρουσίαση, αφαίρεση, και έλεγχο. Αυτή η υποδιαίρεση χωρίζει τις πτυχές αλληλεπίδρασης ανθρώπου-υπολογιστή του πράκτορα από το λειτουργικό πυρήνα της και την επικοινωνία της με άλλους πράκτορες. Τα διαλογικά συστήματα μπορούν συχνά να αντιμετωπισθούν ως σύνολο συνεργαζόμενων πρακτόρων [5] [6] [7].



Σχήμα 8.2.2 Απεικονίζεται η επικοινωνία και η συνεργασία μεταξύ των πρακτόρων.

Οι πράκτορες ειδικευμένοι στην αλληλεπίδραση ανθρώπου-υπολογιστή δέχονται την εισαγωγή από τους χρήστες και απεικονίζουν τα δεδομένα. Άλλοι πράκτορες διατηρούν το μοντέλο δεδομένων του συστήματος και προσφέρουν τη λειτουργικότητα που ενεργεί σε αυτά τα δεδομένα.
Πρόσθετοι πράκτορες είναι αρμόδιοι για διαφορετικούς στόχους, όπως ο χειρισμός λαθών ή την επικοινωνία με άλλα συστήματα λογισμικού. Εκτός από αυτήν την κάθετη αποσύνθεση της λειτουργίας συστημάτων, αντιμετωπίζουμε συχνά μια οριζόντια αποσύνθεση.
Σε μια τέτοια αρχιτεκτονική των συνεργαζόμενων πρακτόρων κάθε πράκτορας είναι ειδικευμένος για έναν συγκεκριμένο στόχο και όλοι οι πράκτορες παρέχουν μαζί τη λειτουργία του συστήματος. Μερικά βασικά χαρακτηριστικά είναι τα εξής:
1). Οι πράκτορες διατηρούν συχνά την κατάστασή και τα στοιχεία τους.
2). Οι διαλογικοί πράκτορες παρέχουν το ενδιάμεσο με τον χρήστη τους δεδομένου ότι οι αντίστοιχες ανθρώπου-υπολογιστή αλληλεπιδράσεις τους είναι συχνά διαφορετικές.
3). Οι συνεργαζόμενοι πράκτορες χρειάζονται έναν μηχανισμό για να ανταλλάξουν αποτελεσματικά τα στοιχεία, τα μηνύματα, και τα γεγονότα.
4). Πολλές μεγάλες εφαρμογές είναι συστήματα πολλών χρηστών (multi user systems). Κατά συνέπεια, το multi tasking είναι μια σημαντική ανησυχία όταν ο σχεδιάζονται τέτοια συστήματα λογισμικού.
5). Τα συστήματα εξελίσσονται με την πάροδο του χρόνου. Ειδικά η πτυχή της παρουσίασης πρέπει να μπορεί να αλλάζει. Η χρησιμοποίηση της γραφικής παράστασης, και πιο πρόσφατα, συμπεριλαμβανομένων των χαρακτηριστικών γνωρισμάτων πολυμέσων (multimedia features), είναι παραδείγματα των κυρίαρχων αλλαγών στα ενδιάμεσα με τον χρήστη. Αλλαγές στους μεμονωμένους πράκτορες ή την επέκταση του συστήματος με νέους πράκτορες, δεν πρέπει να έχει επιπτώσεις σε ολόκληρο το σύστημα.



Σχήμα 8.2.3 Απεικόνιση της δομής της ιεραρχίας των πρακτόρων στο πρότυπο Παρουσίαση-αφαίρεση-έλεγχος

Η δομή της διαλογικής εφαρμογής ως ιεραρχία των αποκαλούμενων πρακτόρων PAC, φαίνεται και στο παραπάνω σχήμα. Υπάρχει ένας υψηλού επιπέδου πράκτορας, γνωστός και ως κορυφαίος πράκτορας PAC, διάφοροι ενδιάμεσου επιπέδου πράκτορες PAC, και ακόμα περισσότεροι χαμηλού επιπέδου PAC πράκτορες. Κάθε πράκτορας είναι αρμόδιος για μια συγκεκριμένη πτυχή της λειτουργίας της εφαρμογής και αποτελείται από τρία συστατικά. Την παρουσίαση, την αφαίρεση, και τον έλεγχο.
1). Ο κορυφαίος πράκτορας PAC παρέχει το λειτουργικό πυρήνα του συστήματος. Οι περισσότεροι άλλοι πράκτορες PAC εξαρτώνται ή αναπτύσσουν δραστηριότητες σε αυτόν τον πυρήνα. Επιπλέον, ο κορυφαίος πράκτορας PAC περιλαμβάνει εκείνα τα μέρη του ενδιάμεσου με τον χρήστη που δεν μπορούν να οριστούν στις δευτερεύουσες δραστηριότητες.
2). Οι χαμηλού επιπέδου PAC πράκτορες αναπαριστούν σε ποιους χρήστες του συστήματος μπορούν να ενεργήσουν οι ανεξάρτητες έννοιες. Επίσης παρουσιάζουν αυτές τις έννοιες στο χρήστη και παρέχουν όλες τις λειτουργίες που οι χρήστες μπορούν να εκτελέσουν σε αυτούς τους πράκτορες, όπως μεγέθυνση ή η κίνηση ενός ιστογράμματος.
3). Οι ενδιάμεσου επιπέδου πράκτορες PAC αναπαριστούν είτε τους συνδυασμούς είτε τις σχέσεις μεταξύ των χαμηλότερων πρακτόρων.
Πιο αναλυτικά:
1). Η κύρια ευθύνη του κορυφαίου πράκτορα PAC είναι να παρέχει τα δεδομένα του μοντέλου του λογισμικού. Διατηρείται στο τμήμα αφαίρεσης αυτού του πράκτορα. Η διεπαφή της αφαίρεσης προσφέρει τις μεθόδους για να χειριστούν τα δεδομένα του μοντέλου και για να ανακτήσει τις πληροφορίες για αυτό.

Το κορυφαίο τμήμα παρουσίασης έχει λίγη ευθύνη. Μπορεί να περιλαμβάνει τα στοιχεία ενδιάμεσων με τον χρήστη που είναι κοινά για ολόκληρη την εφαρμογή. Σε μερικά συστήματα, δεν υπάρχει κορυφαία παρουσίαση.

Το τμήμα ελέγχου του κορυφαίου πράκτορα PAC έχει τρεις ευθύνες. Κατ' αρχάς, επιτρέπει στους χαμηλότερου επιπέδου πράκτορες να χρησιμοποιούν τις κορυφαίες υπηρεσίες, συνήθως να έχουν πρόσβαση και να χειριστούν τα δεδομένα του μοντέλου. Τα εισερχόμενα αιτήματα υπηρεσιών από τους χαμηλότερου επιπέδου πράκτορες διαβιβάζονται είτε στην αφαίρεση είτε στο τμήμα αναπαράστασης.
Δεύτερον, ο έλεγχος συντονίζει την ιεραρχία των πρακτόρων PAC. Διατηρεί τις πληροφορίες για τις συνδέσεις του κορυφαίου πράκτορα στους χαμηλότερου επιπέδου πράκτορες. Ο έλεγχος χρησιμοποιεί αυτές τις πληροφορίες που εξασφαλίζουν σωστή συνεργασία και ανταλλαγή δεδομένων μεταξύ των κορυφαίων πρακτόρων και πρακτόρων χαμηλού επιπέδου. Καλύπτει τις λειτουργίες για την ανταλλαγή των γεγονότων, των μηνυμάτων, και των δεδομένων με τους χαμηλότερου επιπέδου πράκτορες.
Τρίτον, ο έλεγχος διατηρεί τις πληροφορίες για την αλληλεπίδραση του χρήστη με το σύστημα.
2). Οι χαμηλού επιπέδου PAC πράκτορες αναπαριστούν τη συγκεκριμένη σημασιολογική έννοια της περιοχής εφαρμογής. Αυτή η σημασιολογική έννοια μπορεί να είναι τόσο χαμηλού επιπέδου όσο ένα απλό γραφικό αντικείμενο όπως ένας κύκλος ή τόσο σύνθετη όσο ένα ιστόγραμμα που συνοψίζει όλα τα δεδομένα στο σύστημα.
Το τμήμα παρουσίασης ενός χαμηλού επιπέδου PAC πράκτορα παρουσιάζει μια συγκεκριμένη όψη επάνω στην αντίστοιχη σημασιολογική έννοια και παρέχει

πρόσβαση σε όλους τους χρήστες τις λειτουργίες που μπορούν να χρησιμοποιήσουν. Εσωτερικά, το τμήμα παρουσίασης διατηρεί πληροφορίες για την όψη, παραδείγματος χάριν η θέση της στην οθόνη.
Το τμήμα αφαίρεσης ενός χαμηλού PAC πράκτορα έχει την παρόμοια ευθύνη όπως το τμήμα αφαίρεσης του κορυφαίου πράκτορα PAC. Διατηρεί τα συγκεκριμένα δεδομένα πρακτόρων. Σε αντίθεση με την κορυφαία αφαίρεση, εντούτοις, κανένας άλλος πράκτορας PAC δεν εξαρτάται από αυτά τα δεδομένα.

Το τμήμα ελέγχου ενός χαμηλού επιπέδου PAC πράκτορα διατηρεί τη σταθερότητα μεταξύ του τμήματος αφαίρεσης και παρουσίασης, αποφεύγοντας τις άμεσες εξαρτήσεις μεταξύ τους. Χρησιμεύει σαν μετασχηματιστής και εκτελεί και τη διεπαφή και τον μετασχηματισμό δεδομένων.
Επιπλέον, ο έλεγχος επικοινωνεί με τους υψηλότερου επιπέδου πράκτορες για να ανταλλάσει δεδομένα, γεγονότα, ή πληροφορίες κατάστασης. Τα εισερχόμενα γεγονότα, τα μηνύματα, ή τα δεδομένα από τους υψηλότερου επιπέδου πράκτορες PAC διαβιβάζονται είτε στην αφαίρεση είτε στο τμήμα παρουσίασης.
Οι έννοιες που αναπαρίστανται από τους χαμηλού επιπέδου PAC πράκτορες, όπως τα διαγράμματα, είναι ατομικά με την έννοια ότι είναι οι μικρότερες μονάδες που ένας χρήστης μπορεί να χειριστεί.
Οι χαμηλού επιπέδου PAC πράκτορες δεν είναι περιορισμένοι να παρέχουν τις σημασιολογικές έννοιες του πεδίου εφαρμογής. Μπορεί επίσης να διευκρινιστούν πράκτορες χαμηλού επιπέδου που εφαρμόζουν τις υπηρεσίες συστημάτων. Παραδείγματος χάριν, μπορεί να υπάρξει ένας πράκτορας επικοινωνίας που επιτρέπει στο σύστημα να συνεργαστεί με άλλες εφαρμογές και για να ελέγξει αυτήν την συνεργασία.
3). Οι μεσαίου επιπέδου πράκτορες PAC μπορούν να ενσωματώσουν δύο διαφορετικούς ρόλους. Σύνθεση και συντονισμό. Όταν, για παράδειγμα, κάθε αντικείμενο σε μια σύνθετη γραφική παράσταση αντιπροσωπεύεται από έναν ξεχωριστό πράκτορα PAC, ένας μεσαίου επιπέδου πράκτορας συνθέτει αυτά τα αντικείμενα σε ένα σύνθετο γραφικό αντικείμενο. Ο πράκτορας αυτός καθορίζει μια νέα αφαίρεση, της οποίας η συμπεριφορά καλύπτει και τη συμπεριφορά των συστατικών της και των νέων χαρακτηριστικών που προστίθενται στη σύνθεση. Ο δεύτερος ρόλος ενός μεσαίου επιπέδου πράκτορα είναι να διατηρήσει τη σταθερότητα μεταξύ των πρακτόρων χαμηλού επιπέδου, παραδείγματος χάριν όταν συντονίζονται πολλές όψεις επάνω στα ίδια δεδομένα.
Το τμήμα αφαίρεσης διατηρεί τα συγκεκριμένα δεδομένα του μεσαίου επιπέδου πράκτορα PAC. Το τμήμα παρουσίασης εφαρμόζει το ενδιάμεσο με τον χρήστη του. Το τμήμα ελέγχου έχει όλες τις ευθύνες των τμημάτων ελέγχου χαμηλού επιπέδου PAC των πρακτόρων και του κορυφαίου πράκτορα PAC.

Το αρχιτεκτονικό πρότυπο παρουσίαση-αφαίρεση-έλεγχος έχει διάφορα πλεονεκτήματα:
1). Χωρισμός των ανησυχιών:
Οι διαφορετικές σημασιολογικές έννοιες της εφαρμογής αναπαρίστανται από ξεχωριστούς πράκτορες. Κάθε πράκτορας διατηρεί την κατάσταση και τα δεδομένα του, που συντονίζεται με αυτά, αλλά είναι ανεξάρτητα από άλλους πράκτορες PAC. Οι μεμονωμένοι πράκτορες PAC παρέχουν επίσης την δικιά τους αλληλεπίδραση ανθρώπου-υπολογιστή. Αυτό επιτρέπει να αναπτυχθεί ένα μοντέλο δεδομένων και ένα ενδιάμεσο με τον χρήστη για κάθε σημασιολογική έννοια ή στόχο μέσα στην εφαρμογή, ανεξάρτητα από άλλες σημασιολογικές έννοιες ή στόχους.

2). Υποστήριξη για το πολλαπλό καθήκον (multitasking):
Οι πράκτορες PAC μπορούν να διανεμηθούν εύκολα σε διαφορετικά νήματα, τις διαδικασίες, ή τις μηχανές. Η επέκταση ενός πράκτορα PAC με την κατάλληλη λειτουργία IPC έχει επιπτώσεις ακριβώς στο τμήμα ελέγχου της.
3). Υποστήριξη για αλλαγή και επέκταση:
Οι αλλαγές μέσα στα τμήματα παρουσίασης ή αφαίρεσης ενός πράκτορα PAC δεν έχουν επιπτώσεις σε άλλους πράκτορες του συστήματος. Αυτό επιτρέπει να τροποποιηθούν χωριστά ή να συντονιστούν τα δεδομένα που βρίσκονται κάτω από έναν πράκτορα PAC ή να αλλαχτεί το ενδιάμεσο με τον χρήστη του.

Τα μειονεκτήματα αυτού του προτύπου είναι τα ακόλουθα:
1). Αυξανόμενη πολυπλοκότητα συστήματος:
Η εφαρμογή κάθε σημασιολογικής έννοιας μέσα σε μια εφαρμογή ως πράκτορα PAC της μπορεί να οδηγήσει σε μια δομή πολυσύνθετων συστημάτων. Παραδείγματος χάριν, εάν κάθε γραφικό αντικείμενο όπως ένας κύκλος ή ένα τετράγωνο μέσα σε έναν συντάκτη γραφικής παράστασης εφαρμόζεται ως πράκτορας PAC του, το σύστημα θα πνίγονταν σε μια θάλασσα από πράκτορες. Αυτοί οι πράκτορες πρέπει επίσης να συντονιστούν και να ελεγχθούν κάτι που απαιτεί επιπρόσθετους πράκτορες συντονισμού.
2). Σύνθετο τμήμα ελέγχου:
Σε ένα σύστημα PAC, τα τμήματα ελέγχου είναι οι μεσολαβητές επικοινωνίας μεταξύ των τμημάτων αφαίρεσης και παρουσίασης ενός πράκτορα και μεταξύ των διαφορετικών πρακτόρων PAC. Η ποιότητα των εφαρμογών ελέγχου είναι επομένως κρίσιμη για μια αποτελεσματική συνεργασία μεταξύ των πρακτόρων και επομένως για τη γενική ποιότητα της αρχιτεκτονικής συστημάτων. Όλοι οι μεμονωμένοι ρόλοι των τμημάτων ελέγχου πρέπει να διαχωριστούν έντονα μεταξύ τους. Η εφαρμογή αυτών των ρόλων δεν πρέπει να εξαρτηθεί από τις συγκεκριμένες λεπτομέρειες άλλων πρακτόρων, όπως το συγκεκριμένο όνομα ή η φυσική θέση τους σε ένα διανεμημένο σύστημα.
3). Αποδοτικότητα:
Τα έξοδα στην επικοινωνία μεταξύ των πρακτόρων PAC μπορούν να μειώσουν την αποδοτικότητα των συστημάτων. Παραδείγματος χάριν, εάν ένας χαμηλού επιπέδου πράκτορας ανακτά τα δεδομένα του από τον κορυφαίο πράκτορα, όλοι οι μεσαίου επιπέδου πράκτορες, κατά μήκος της πορείας, από το κατώτατο σημείο μέχρι την κορυφή της ιεραρχίας PAC θα περιλαμβάνονται σε αυτήν την ανταλλαγή στοιχείων.

8.3 Πρότυπο Σωλήνωσης (Pipeline Pattern)

Στην τεχνολογία λογισμικού, μια σωλήνωση αποτελείται από μια αλυσίδα των στοιχείων επεξεργασίας (διαδικασίες, νήματα, κ.λπ.), που τακτοποιούνται έτσι ώστε η έξοδος κάθε στοιχείου να είναι η είσοδος του επόμενου. Συνήθως κάποιο ποσό αποθήκευσης παρέχεται μεταξύ διαδοχικών στοιχείων. Οι πληροφορίες που ρέουν σε αυτές τις σωληνώσεις είναι συχνά ένα ρεύμα των αρχείων, των bit ή των Byte. Η έννοια λέγεται επίσης σχεδιαστικό πρότυπο σωλήνων και φίλτρων (pipes and filters design pattern).
Multiprocessed Σωληνώσεις. Οι σωληνώσεις εφαρμόζονται συχνά σε ένα λειτουργικό σύστημα πολλών καθηκόντων (multitasking OS), με την προώθηση όλων των στοιχείων ταυτόχρονα με τις διαδικασίες και εξυπηρετώντας τα αιτήματα που διαβάζουν τα δεδομένα. Κατ' αυτό τον τρόπο, ο επεξεργαστής (CPU) θα εναλλάσσεται μεταξύ των διαδικασιών από το χρονοπρογραμματιστή για να

ελαχιστοποιηθεί ο ανενεργός χρόνος (idle time). Σε άλλα μοντέλα τα στοιχεία εφαρμόζονται ως ελαφριά νήματα, για να μειώσουν την επιβάρυνση στο λειτουργικό σύστημα που περιλαμβάνεται συχνά με τις διαδικασίες. Ανάλογα με το λειτουργικό σύστημα, τα νήματα μπορούν να σχεδιαστούν άμεσα από το λειτουργικό σύστημα ή από έναν διευθυντή νημάτων.

Ψευδό-σωληνώσεις (Pseudo-pipelines). Σε λειτουργικά συστήματα απλής ανάθεσης (single-tasking OS), οι διαδικασίες μιας σωλήνωσης πρέπει να εκτελεστούν μια-μια σε σειρά. Κατά συνέπεια η έξοδος κάθε διαδικασίας πρέπει να σωθεί σε ένα προσωρινό αρχείο, το οποίο διαβάζεται έπειτα με την επόμενη διαδικασία. Δεδομένου ότι δεν υπάρχει παραλληλισμός ή εναλλαγή επεξεργαστή, αυτή η έκδοση καλείται "ψευδό-σωλήνωση " [12].

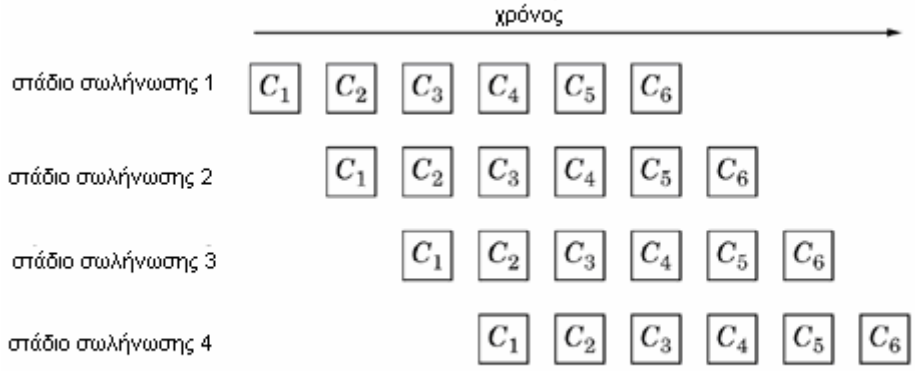
Σε μερικές περιπτώσεις ο γενικός υπολογισμός μιας λύσης ενός προβλήματος περιλαμβάνει την εκτέλεση ενός υπολογισμού σε πολλά σύνολα δεδομένων, όπου ο υπολογισμός μπορεί να αντιμετωπισθεί μέσα από μια σειρά βημάτων. Έστω ότι πρέπει να κατασκευαστούν διάφορα αυτοκίνητα. Η διαδικασία παραγωγής μπορεί να αναλυθεί σε μια ακολουθία διαδικασιών, όπου κάθε μια διαδικασία προσθέτει κάποιο συστατικό, για παράδειγμα μηχανή ή ανεμοθραύστη, στο αυτοκίνητο. Μια σωλήνωση ορίζει ένα συστατικό σε κάθε εργαζόμενο. Καθώς κάθε αυτοκίνητο κινείται κάτω στη γραμμή παραγωγής, κάθε εργαζόμενος εγκαθιστά το ίδιο συστατικό επανειλημμένως σε μια σειρά αυτοκινήτων. Σε κάποιο σημείο οι εργαζόμενοι μπορεί όλοι να είναι πολυάσχολοι ταυτόχρονα, όλοι εκτελούν τις διαδικασίες τους στα αυτοκίνητα που είναι αυτήν την περίοδο στους σταθμούς τους. Παραδείγματα από σωληνώσεις συναντιούνται σε πολλά επίπεδα των συστημάτων υπολογιστή (computer systems) [13] [87].

Η δυνατότητα της ταυτόχρονης εκτέλεσης διαφορετικών διαδικασιών στα διαφορετικά στοιχεία δεδομένων, είναι ο συγχρονισμός που εκμεταλλεύεται αυτό το πρότυπο [24]. Από την άποψη της ανάλυσης, κάθε στόχος αποτελείται από επανειλημμένη εφαρμογή μιας λειτουργίας σε ένα στοιχείο δεδομένου, και από τις εξαρτήσεις μεταξύ των στόχων, ορίζοντας περιορισμούς, επιβάλλοντας τη σειρά με την οποία οι διαδικασίες πρέπει να εκτελεστούν σε κάθε στοιχείο. Μια καλή λύση πρέπει να εκφράσει τους περιορισμούς σειράς-ακολουθίας. Σε μερικές εφαρμογές, είναι αναμενόμενες οι μελλοντικές προσθήκες, οι τροποποιήσεις, ή η αλλαγή της σειράς των σταδίων. Σε μερικές εφαρμογές, στοιχεία στην ακολουθία εισαγωγής μπορούν να περιέχουν τα λάθη που αποτρέπουν την επεξεργασία τους [13].

Στους όρους παράλληλου-προγραμματισμού, η ιδέα είναι να οριστεί κάθε στόχος (στάδιο της σωλήνωσης) και να υπάρξει ένας μηχανισμός, με τον οποίο κάθε στάδιο της σωλήνωσης μπορεί να στείλει τα δεδομένα στο επόμενο στάδιο. Αυτή η στρατηγική είναι πιθανώς ο απλούστερος τρόπος να εξεταστεί αυτός ο τύπος σειράς των περιορισμών. Επιτρέπει στην εφαρμογή να εκμεταλλευτεί το ειδικής χρήσης υλικό με την κατάλληλη χαρτογράφηση των σταδίων σωληνώσεων και παρέχει έναν λογικό μηχανισμό για το χειρισμό των λαθών. Είναι επίσης πιθανό να παραχθεί ένα πρότυπο που μπορεί αργότερα να επεκταθεί ή να τροποποιηθεί.

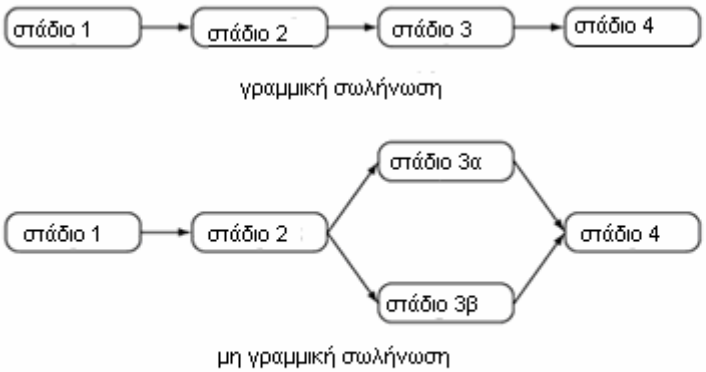
Η σωλήνωση πρέπει να λειτουργεί με τον εξής τρόπο. Το Ci αντιπροσωπεύει έναν πολλαπλών βημάτων υπολογισμό στο στοιχείο i. Ci(j) είναι το jth βήμα του υπολογισμού. Η ιδέα είναι να χαρτογραφηθούν τα βήματα υπολογισμού στα στάδια σωληνώσεων έτσι ώστε κάθε στάδιο της σωλήνωσης υπολογίζει ένα βήμα. Αρχικά, η πρώτη φάση της σωλήνωσης εκτελεί το C1 (1). Αφού ολοκληρωθεί, το δεύτερο στάδιο της σωλήνωσης λαμβάνει το πρώτο στοιχείο και υπολογίζει C1 (2), ενώ η πρώτη φάση υπολογίζει το πρώτο βήμα του δεύτερου στοιχείου, C2 (1). Έπειτα, το

τρίτο στάδιο υπολογίζει C1 (3), ενώ το δεύτερο στάδιο υπολογίζει C2 (2) και τη πρώτη φάση C3 (1). Το παρακάτω σχήμα 8.3.1 [13] [14], επεξηγεί την εργασία για μια σωλήνωση που αποτελείται από τέσσερα στάδια. Αρχικά ο συγχρονισμός είναι περιορισμένος και μερικοί πόροι παραμένουν μη απασχολημένοι έως ότου όλα τα στάδια ασχολούνται με την εργασία. Αυτό αναφέρεται ως πλήρωση της σωλήνωσης (filing the pipeline). Στο τέλος του υπολογισμού (draining the pipeline), πάλι υπάρχει περιορισμένος συγχρονισμός και μη απασχολημένοι πόροι, αφού μόνο το τελικό στοιχείο λειτουργεί μέσω της σωλήνωσης. Πρέπει ο χρόνος πλήρωσης ή στραγγίσματος της σωλήνωσης να είναι μικρός έναντι του συνολικού χρόνου του υπολογισμού. Αυτό θα συμβεί, εάν ο αριθμός σταδίων είναι μικρός συγκρινόμενος με τον αριθμό στοιχείων που υποβάλλονται σε επεξεργασία. Η αποδοτικότητα μεγιστοποιείται, εάν ο χρόνος που λαμβάνεται για να επεξεργαστεί ένα στοιχείο στοιχείων είναι κατά προσέγγιση ο ίδιος για κάθε στάδιο.



Σχήμα 8.3.1 Απεικονίζεται ο τρόπος λειτουργίας της Σωλήνωσης.

Αυτή η ιδέα μπορεί να επεκταθεί για να περιλάβει τις καταστάσεις γενικότερες από μια απολύτως γραμμική σωλήνωση. Παραδείγματος χάριν, το παρακάτω σχήμα 8.3.2 επεξηγεί δύο σωληνώσεις, κάθε μια με τέσσερα στάδια. Στη δεύτερη, το τρίτο στάδιο αποτελείται από δύο διαδικασίες που μπορούν να εκτελεστούν ταυτόχρονα [13] [14].



Σχήμα 8.3.2 Απεικονίζονται δυο σωληνώσεις. Η πρώτη είναι γραμμική ενώ η δεύτερη δεν είναι γραμμική.

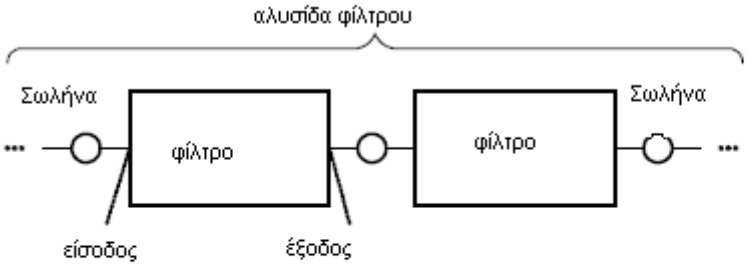
Το σχήμα 8.3.2 δείχνει ότι μπορεί να αναπαρασταθεί μια σωλήνωση ως κατευθυνόμενος γράφος, με κόμβους που αντιστοιχούν στα στοιχεία του υπολογισμού και των ακμές που δείχνουν τη ροή πληροφοριών. Για να συντηρηθεί η ιδέα της σωλήνωσης, είναι απαραίτητο ότι αυτός ο γράφος είναι ακυκλικός [14].

Παραδείγματα του προτύπου αυτού βρίσκονται στις εργασίες [25] [26] [27] [28] [29].

Σωλήνωση και φίλτρο (Pipeline and filter)

Οι εφαρμογές που εκτελούν την επεξεργασία των πολλών δεδομένων μέσα στα διάφορα διακριτά βήματα επεξεργασίας υφίστανται την ακαμψία στην επαναχρησιμοποίηση και τον επανασυνδυασμό των βημάτων επεξεργασίας, εάν αναμειγνύονται σε μια συγκέντρωση. Το αρχιτεκτονικό πρότυπο σωλήνων και φίλτρων προσεγγίζει αυτά τα προβλήματα μεταλλεύοντας τα οφέλη των ομοιόμορφων μεθόδων διασύνδεσης για να κατασκευαστούν αλυσίδες των μη-αλληλοεξαρτώμενων οντοτήτων επεξεργασίας [21].

Το αρχιτεκτονικό πρότυπο σωλήνων και φίλτρων παρέχει μια δομή για τα συστήματα που επεξεργάζονται ένα ρεύμα των δεδομένων. Κάθε βήμα επεξεργασίας ενθυλακώνεται σε ένα τμήμα φίλτρων. Τα δεδομένα περνούν μέσω των σωλήνων μεταξύ των παρακείμενων φίλτρων. Ο επανασυνδυασμός των φίλτρων επιτρέπει να χτιστούν οικογένειες των σχετικών συστημάτων [11] [20].
Το αρχιτεκτονικό πρότυπο σωλήνων και φίλτρων διαιρεί το στόχο ενός συστήματος σε διαδοχικά βήματα επεξεργασίας [19]. Αυτά τα βήματα συνδέονται με τη ροή δεδομένων μέσω του συστήματος. Το δεδομένο εξόδου ενός βήματος είναι η εισαγωγή στο επόμενο βήμα. Κάθε βήμα επεξεργασίας εφαρμόζεται από ένα τμήμα φίλτρων. Ένα φίλτρο καταναλώνει και παραδίδει τα δεδομένα αυξητικά. Η εισαγωγή στο σύστημα παρέχεται από μια πηγή δεδομένων όπως ένα αρχείο κειμένου. Η παραγωγή-έξοδος ρέει σε έναν νεροχύτη δεδομένων όπως ένα αρχείο, ένα πρόγραμμα και ούτω καθεξής. Η πηγή δεδομένων, τα φίλτρα και ο νεροχύτης δεδομένων συνδέονται διαδοχικά με τους σωλήνες. Κάθε σωλήνας εφαρμόζει τη ροή δεδομένων μεταξύ των παρακείμενων βημάτων επεξεργασίας. Η ακολουθία φίλτρων που συνδυάζεται από τους σωλήνες καλείται επεξεργασία σωλήνωσης.
Τα βήματα επεξεργασίας συνδέονται με σωλήνες για να δημιουργήσουν τις αλυσίδες φίλτρων (filter chain). Μια ακολουθία από γενικά ανεξάρτητα, βήματα επεξεργασίας που εκτελούνται από τα φίλτρα που συνδέονται με τα ομοιόμορφα κανάλια, επονομαζόμενα σωλήνες, χρησιμοποιούνται για να επεξεργαστούν αυξητικά τα δεδομένα, αποτελεί μια αλυσίδα φίλτρων [21].



Σχήμα 8.3.3 Απεικονίζεται μια αλυσίδα φίλτρου, όπου η ροή των δεδομένων γίνεται από αριστερά προς τα δεξιά.
Εφαρμόζοντας το πρότυπο Σωλήνων και φίλτρων, υπάρχουν ορισμένα πλεονεκτήματα και μειονεκτήματα. Μπορεί να εκτελεστεί σύνθετη επεξεργασία σε δεδομένα διατηρώντας και την ανεξαρτησία και την ευελιξία [18]. Όμως ο χειρισμός

των λαθών είναι δύσκολος και μερικές φορές τα φίλτρα καταναλώνουν όλα τα δεδομένα εισόδου (input) πριν να παράγουν κάποιο αποτέλεσμα (output). Επίσης μπορεί να υπάρχει και επιβάρυνση λόγω του μετασχηματισμού των δεδομένων που γίνεται μεταξύ των φίλτρων [4].
Πιο συγκεκριμένα τα πλεονεκτήματα είναι τα εξής [22] [23] [30].
1). Μια διεπαφή των τμημάτων των φίλτρων και σωλήνων επιτρέπει στις αλυσίδες φίλτρων να συνδυαστούν και συνδυαστούν ξανά πολύ εύκολα. Έτσι, πολλές αλυσίδες φίλτρων με διαφορετική συμπεριφορά μπορούν να παραχθούν πολύ γρήγορα. Επίσης, οι σωλήνες και τα φίλτρα απλοποιούν την επαναχρησιμοποίηση. Οι νέες αλυσίδες φίλτρων, ακόμη και με την απολύτως διαφορετική συμπεριφορά, μπορούν να σχεδιαστούν με την εκ νέου ρύθμιση των φίλτρων μιας υπάρχουσας αλυσίδας φίλτρων, ή την προσθήκη μερικών νέων φίλτρων. Επιπλέον, οι πλήρεις αλυσίδες φίλτρων μπορούν να χρησιμοποιηθούν ως ένα φίλτρο σε μια νέα αλυσίδα.
2). Μέσω της αυξητικής επεξεργασίας των δεδομένων, τα ενεργά φίλτρα που τρέχουν σε ένα σύστημα πολυεπεξεργαστών ή σε ένα δίκτυο μπορούν να εκτελέσουν τις λειτουργίες τους παράλληλα, επειδή όλα τα φίλτρα μπορούν ήδη να αρχίσουν να εργάζονται στα μερικά αποτελέσματα των προκατόχων τους αντί να πρέπει να περιμένουν την ολοκλήρωσή τους. Αυτό μπορεί να βελτιώσει την απόδοση του συστήματος χρησιμοποιώντας μια αλυσίδα φίλτρων.
Τα μειονεκτήματα του προτύπου πιο αναλυτικά στις επόμενες πέντε παραγράφους [21] [23] [30].
1). Ο χειρισμός σφάλματος είναι το μεγαλύτερο πρόβλημα των σωλήνων και των φίλτρων, επειδή τα τμήματα σωληνώσεων έχουν μόνο μια δυνατότητα της επικοινωνίας. Τουλάχιστον η υποβολή έκθεσης σφάλματος μπορεί να γίνει χρησιμοποιώντας ένα χωριστό κανάλι εξόδου, αλλά στις περισσότερες περιπτώσεις η μόνη δυνατότητα είναι να γίνει επανεκκίνηση της σωλήνωσης εάν ένα σφάλμα εμφανίστηκε και για να ελπίσει ότι θα ολοκληρωθεί χωρίς αποτυχία. Εάν ο χειρισμός σφάλματος είναι σημαντικός, πρέπει να εξεταστούν διαφορετικές αρχιτεκτονικές.

2). Η διανομή των πληροφοριών κατάστασης. Εάν τα διαφορετικά στάδια επεξεργασίας πρέπει να μοιραστούν ένα μεγάλο ποσό global δεδομένων, το πρότυπο σωλήνων και φίλτρων μπορεί να είναι ακατάλληλο. Αυτό επειδή εισάγει εξαρτήσεις μεταξύ των φίλτρων, που μειώνουν τη δυνατότητα να επανασυνδυαστούν οι αλυσίδες φίλτρων και να επεξεργαστούν παράλληλα τα δεδομένα.

3). Ο μηχανισμός σωλήνων είναι ακριβός. Εάν οι σωλήνες και τα φίλτρα έχουν διαφορετικές μορφές δεδομένων, τα γενικά έξοδα της μετατροπής των δεδομένων μπορεί να μειώσουν τα οφέλη των σωλήνων και των φίλτρων, ειδικά τις βελτιώσεις απόδοσης. Τα χαρακτηριστικά γνωρίσματα που προσφέρονται από τους σωλήνες, όπως η αποθήκευση, η αναμονή και η μεταφορά των δεδομένων, δεσμεύουν πόρους περιορίζοντας τον αριθμό σωλήνων.

4). Λόγω της γραμμικής αρχιτεκτονικής των σωλήνων και των φίλτρων, η ρυθμοαπόδοση μιας αλυσίδας φίλτρων περιορίζεται από τη ρυθμοαπόδοση του πιο αργού συστατικού της. Έτσι η εξελιξιμότητα μιας αλυσίδας φίλτρων περιορίζεται στην εξελιξιμότητα των συστατικών της.

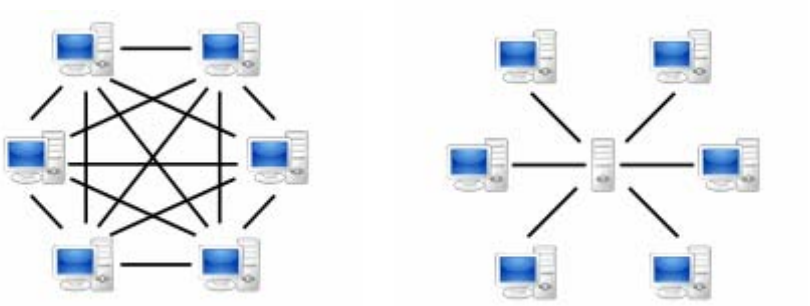
5). Το κόστος για τη μεταφορά των δεδομένων μεταξύ των φίλτρων μπορεί να είναι σχετικά υψηλό έναντι στο κόστος του υπολογισμού που εκτελεί ένα ενιαίο φίλτρο, έτσι το σύστημα θα ασχολείται κυρίως με τη μεταφορά δεδομένων αντί των

υπολογισμών. Εκτός αυτού, εάν το ποσό δεδομένων στη διαδικασία είναι πολύ μικρό, το κόστος μια αλυσίδα φίλτρων μπορεί να είναι πάρα πολύ υψηλό.

Το πρότυπο αυτό χρησιμοποιήθηκε στο σύστημα UNIX, το οποίο έκανε το πρότυπο διάσημο [15]. Η CMS Σωλήνωση επεκτείνει τα λειτουργικά συστήματα των κεντρικών υπολογιστών της IBM που υποστηρίζουν τη Σωλήνωση και τις αρχιτεκτονικές πλαισίων [16]. Το LASSPTools είναι ένα σύνολο εργαλείων για την αριθμητική ανάλυση και τη γραφική παράσταση. Το LASSPTools περιέχει τα προγράμματα φίλτρων που μπορούν να συνδυαστούν χρησιμοποιώντας τους σωλήνες UNIX [17].

8.4 Peer to Peer

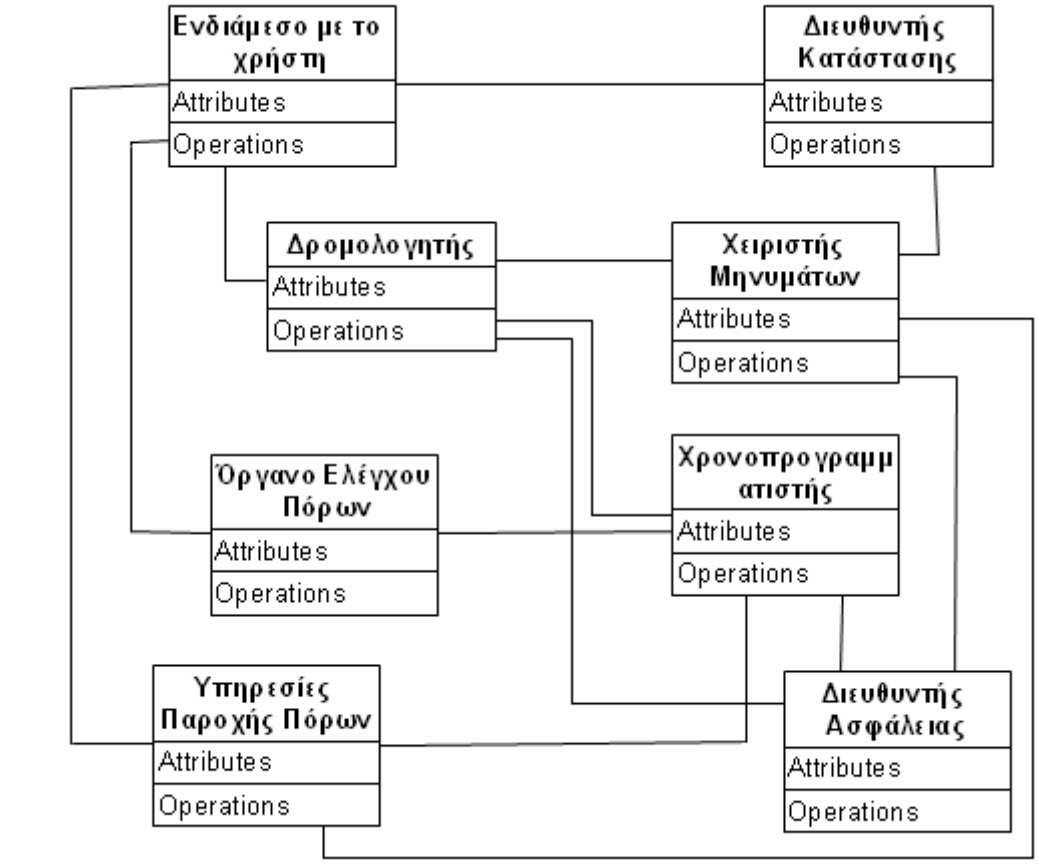
Πρόσφατα, οι εφαρμογές peer-to-peer κέρδισαν μεγάλη δημοσιότητα λόγω των μοιραζόμενων εφαρμογών όπως το Napster, Gnutella, Kazaa, eDonkey, και άλλα. Το peer-to-peer αναφέρεται σε μια αποκεντρωμένη αρχιτεκτονική στην οποία όλοι οι κόμβοι (δηλ. υπολογιστές) έχουν τις ίδιες ικανότητες και ευθύνες και όλη η επικοινωνία είναι συμμετρική. Τα peer-to-peer συστήματα έχουν το μοναδικό πλεονέκτημα να εκμεταλλεύονται αδρανείς πόρους όπως κύκλους υπολογισμού, το εύρος ζώνης, και την αποθήκευση των συμμετεχόντων υπολογιστών. Λειτουργούν σε μεγάλη κλίμακα, σε ιδιαίτερα αναξιόπιστο και επισφαλές περιβάλλον όπως το Διαδίκτυο [31].
Ένα peer to peer ή «P2P» δίκτυο υπολογιστών χρησιμοποιεί διαφορετική συνδεσιμότητα μεταξύ των συμμετεχόντων σε ένα δίκτυο και ένα αθροιστικό εύρους ζώνης (bandwidth) των συμμετεχόντων δικτύων, αντί των συμβατικών συγκεντρωμένων πόρων όπου ένας χαμηλός αριθμός των κεντρικών υπολογιστών (servers) παρέχουν την αξία πυρήνα (core value) σε μια υπηρεσία ή μια εφαρμογή. Τα P2P δίκτυα χρησιμοποιούνται χαρακτηριστικά για τη σύνδεση των κόμβων μέσω των ειδικών συνδέσεων. Τέτοια δίκτυα είναι χρήσιμα για πολλούς λόγους. Μοίρασμα αρχείων που περιέχουν ήχο, βίντεο, δεδομένα ή οτιδήποτε σε ψηφιακή μορφή και σε πραγματικού χρόνου δεδομένα, όπως η τηλεφωνία.
Ένα καθαρό P2P δίκτυο δεν έχει την έννοια των πελατών ή των κεντρικών υπολογιστών αλλά ίσων όμοιων κόμβων που λειτουργούν ταυτόχρονα, σαν «πελάτες» και «κεντρικοί υπολογιστές» στους άλλους κόμβους στο δίκτυο, όπως φαίνεται στο σχήμα 8.4.1. Αυτό το μοντέλο της ρύθμισης δικτύων διαφέρει από το μοντέλο κεντρικών υπολογιστών -πελατών όπου η επικοινωνία είναι συνήθως από και προς έναν κεντρικό υπολογιστή. Στο σχήμα 8.4.1 φαίνεται αυτή η διάταξη [88].
Αντίθετα από τα παραδοσιακά διανεμημένα συστήματα, τα επιμέρους συστατικά ενός συστήματος peer-to-peer μπορούν ανεξάρτητα να είναι ανοικτοί ή κλειστοί, να ενωθούν ή να αφήσουν το σύστημα. Μπορούν να είναι κατασκευασμένα από χαμηλού κόστους και χαμηλής-αξιοπιστίας τμήματα [35]. Μια πρόσφατη μελέτη ενός δημοφιλούς peer-to-peer συστήματος που διαμοιράζει αρχεία, διαπίστωσε ότι η πλειοψηφία των peers είχε ποσοστό διαθεσιμότητας επιπέδου εφαρμογής κάτω από 20 τοις εκατό [34].
Ένα P2P σύστημα χαρακτηρίζεται από τα ακόλουθα αντικείμενα [32]:
1. Αποτελείται από τους εννοιολογικά εξίσου σημαντικούς κόμβους.
2. Κάθε κόμβος μπορεί να παρέχει και να καταναλώσει τις υπηρεσίες.
3. Αυτές οι σχέσεις υπηρεσιών πρέπει να είναι δυναμικές.
4. Ένας κόμβος ή μια υπηρεσία που αποτυγχάνει δεν μπορεί να ρίξει ολόκληρο το σύστημα.



Σχήμα 8.4.1 Στο πρώτο σχέδιο απεικονίζεται το peer-to-peer σύστημα σε αντίθεση με το δεύτερο σχέδιο που υπάρχει κεντρικός υπολογιστής.

Μερικά δίκτυα και κανάλια όπως το Napster, το OpenNAP και τα κανάλια κεντρικών υπολογιστών IRC χρησιμοποιούν δομή πελάτη-εξυπηρετητή για μερικές στοιχειώδεις εργασίες (π.χ. αναζήτηση-searching) και μια P2P δομή για άλλες εργασίες. Τα δίκτυα όπως το Gnutella [33] και το Freenet χρησιμοποιούν μια P2P δομή για όλους τους σκοπούς και αναφέρονται μερικές φορές ως αληθινά P2P δίκτυα.

Οι συνδέσεις μεταξύ των συμμετεχόντων που ανήκουν στον ίδιο κόμβο διευκρινίζονται στο επόμενο διάγραμμα κλάσεων.



Σχήμα 8.4.2 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Peer to Peer.

Οι συμμετέχοντες που φαίνονται και στο σχήμα 8.4.2 είναι οι εξής [89]:

1). Ο Χειριστής Μηνυμάτων (Message Handler) επιτρέπει στο peer να δημιουργήσει μια εικονική επικάλυψη δικτύων πάνω από την υπάρχουσα φυσική υποδομή δικτύου.

Υποστηρίζει διαφορετικά πρωτόκολλα μεταφορών επιτρέποντας τις μεταδόσεις μηνυμάτων, firewalls ή NATs.
2). Μια ομάδα των Υπηρεσιών Παροχής των Πόρων (Resource Provision Services) εφαρμόζει τους μηχανισμούς διοίκησης των πόρων για τον τοπικό και τηλεχειρισμό των πόρων του peer. Ο ίδιος πόρος μπορεί να χρησιμοποιηθεί με διάφορους τρόπους, εξαρτώμενος από τους περιορισμούς διανομής. Αυτοί οι περιορισμοί μπορούν δυναμικά να αλλάξουν, βασισμένοι στη φύση της επιτρεπόμενης πρόσβασης, και στους συμμετέχοντες σε ποιους η πρόσβαση επιτρέπεται.
3). Οι πόροι του peer παρέχουν μηχανισμούς, που επιτρέπουν την ανάλυση της δομής, της κατάστασης, και των ικανοτήτων τους. Το Όργανο Ελέγχου των Πόρων (Resource Monitor) στηρίζεται σε αυτές τις εγκαταστάσεις για να συγκεντρώσει και να εκθέσει τις πληροφορίες για τους τοπικούς πόρους. Το Όργανο Ελέγχου των Πόρων είναι αρμόδιο για την παροχή ενός καταλόγου διαθέσιμων πόρων, διατηρώντας τις σχετικές πληροφορίες, αναγνωρίζοντας και αναφέροντας τις αποτυχίες.
4). Το Ενδιάμεσο με τον Χρήστη (User Interface) είναι το σημείο εισόδου για να αλληλεπιδράσουν οι χρήστες με το σύστημα των Peers. Είναι αρμόδιο για τη διαχείριση των εντολών χρηστών, οι οποίες είναι μεταφρασμένες σε τοπικές διαδικασίες ελέγχου ή τα σε μηνύματα ελέγχου που διαδίδονται στο δίκτυο. Το Ενδιάμεσο με τον Χρήστη επιτρέπει να ρωτήσει το Όργανο Ελέγχου των Πόρων και να διαμορφώσει τους τοπικούς πόρους, να εκφράσει σύνθετες ερωτήσεις για μακρινούς πόρους και να λειτουργήσει σε αυτούς.
5). Ο Δρομολογητής (Router) καθορίζει τους κανόνες για τις διευθύνσεις, το φιλτράρισμα, την αποστολή και παραλαβή των μηνυμάτων. Οι περισσότερες από τις Peer διαδικασίες, όπως η δημοσίευση, η αναζήτηση και η παράδοση των πόρων, στηρίζονται στο Δρομολογητή.
6). Ο Χρονοπρογραμματιστής (Scheduler) είναι αρμόδιος για τη διαχείριση των αιτημάτων εκτέλεσης στόχου, βασισμένος στις πληροφορίες που παρέχονται από το Όργανο Ελέγχου των Πόρων. Δεδομένου ότι πολλοί Peers μπορούν να παρέχουν τους ίδιους πόρους, οι ενότητες Χρονοπρογραμματιστών τους πρέπει να είναι σε θέση να συνεργαστούν για τη διανομή φόρτου εργασίας.
7). Ο Διευθυντής Ασφάλειας (Security Manager) είναι κυρίως αρμόδιος για την προστασία των κοινών πόρων. Στηρίζεται σε μηχανισμούς για την προστασία , την ακεραιότητα και την αυθεντικότητα των στοιχείων, για να εξασφαλίσει την ιδιωτικότητα-μυστικότητα και την εμπιστευτικότητα στις επικοινωνίες και για να παρέχει τα μέσα για την αυθεντικοποίηση και την έγκριση των χρηστών. Επιτρέπει στα Peers να καθιερώσουν την ταυτότητά τους μέσα σε μια ομάδα. Γενικότερα, ο Διευθυντής Ασφάλειας πρέπει να καθορίσει τις ικανότητες του Peer μέσα σε μια ομάδα peer.
8). Ο Διευθυντής Κατάστασης (State Manager) παρέχει τις εγκαταστάσεις για να ελέγξει και να αλλάξει η κατάσταση ενός Peer. Οι αλλαγές κατάστασης μπορεί να έχουν υπόψη τις ανάγκες του χρήστη, ή να είναι εξαρτώμενες από τη διαμόρφωση της ομάδας των Peer.

Το Ενδιάμεσο με τον Χρήστη (User Interface) γνωρίζει μια ή περισσότερες υπηρεσίες παροχής των πόρων, είτε τοπικές είτε μακρινές. Επιπλέον, πρέπει να είναι σε θέση να ρωτήσει το Όργανο Ελέγχου των Πόρων και τον Διευθυντή Κατάστασης, να συλλέξει τις πληροφορίες για το σύστημα και να τις παρέχει στο χρήστη. Τέλος, το Ενδιάμεσο με τον Χρήστη χρησιμοποιεί το Δρομολογητή όταν δημοσιεύει τις υπηρεσίες παροχής των πόρων του ή ψάχνει για μακρινές.

Ο Διευθυντής Κατάστασης μπορεί να τροποποιήσει την κατάσταση του Peer μέσω μιας εντολής χρηστών που παραλαμβάνεται από το Ενδιάμεσο με τον Χρήστη, ή μετά από την παράδοση ενός γεγονότος από το Χειριστή Μηνυμάτων. Για αυτόν τον λόγο ο Διευθυντής Κατάστασης αναφέρεται και από το Ενδιάμεσο με τον Χρήστη και από το Χειριστή Μηνυμάτων.

Ο Δρομολογητής υπολογίζει τον προορισμό των μηνυμάτων που κατασκευάζονται από το Ενδιάμεσο με τον Χρήστη. Τα μηνύματα μπορούν να διανεμηθούν χρησιμοποιώντας τα υποστηριζόμενα πρωτόκολλα μεταφορών που προσεγγίζονται μέσω ενός χειριστή μηνυμάτων. Όταν ένα μακρινό αίτημα για την παροχή υπηρεσιών παραλαμβάνεται, ο Χρονοπρογραμματιστής ρωτά το Όργανο Ελέγχου των Πόρων για τους διαθέσιμους πόρους. Στη θετική απάντηση, ο Χρονοπρογραμματιστής ενεργοποιεί την αρμόδια υπηρεσία παροχής των πόρων που εδρεύει στο χειριστή κλήσης.
Ο Διευθυντής Ασφάλειας αλληλεπιδρά με το Χρονοπρογραμματιστή, το Δρομολογητή, και το Χειριστή Μηνυμάτων, για να τροποποιήσει τη συμπεριφορά του Peer.
Τα οφέλη του προτύπου είναι [90]:
1). Ικανότητα επαναχρησιμοποίησης. Όλοι οι κόμβοι μοιράζονται την ίδια αρχιτεκτονική, και διακρίνονται μόνο από τις διαμορφώσεις χρόνου εκτέλεσής τους.
2). Αποδοτική διανομή φόρτου εργασίας. Η εύρεση και η παροχή των πόρων εκτελούνται από τις δυναμικές ομάδες συνεργαζόμενων Peer.
3). Έλλειψη ενιαίων σημείων της αποτυχίας. Οι πληροφορίες για τους κοινούς πόρους διανέμονται ομοιόμορφα.

Το πρότυπο έχει μερικά μειονεκτήματα:
1). Σχεδιασμός σύνθετης εφαρμογής. Η απουσία συγκεντρωμένων δεικτών των πόρων απαιτεί αποδοτικούς διανεμημένους αλγορίθμους αναζήτησης. Επιπλέον, οι Υπηρεσίες Παροχής των Πόρων πρέπει να είναι σε θέση να εξυπηρετήσουν ένα αίτημα από μόνες τους, ή με τη συνεργασία με άλλα Peers.
2). Δύσκολη εξέταση και διόρθωση. Ο έλεγχος ροής ταλαντεύεται μεταξύ των Peers, αποτρέποντας να γίνεται ο σωστός έλεγχος σε κάθε peer σε χρόνο εκτέλεσης.
Το Πρότυπο αλυσίδας ευθύνης (Chain of Responsibility Pattern) (βλέπε κεφάλαιο 5.10) αποσυνδέει τον πομπό ενός αιτήματος από το δέκτη του, δίνοντας σε περισσότερα από ένα αντικείμενα που διατίθενται σε μια αλυσίδα μιας πιθανότητας να χειριστούν το αίτημα. Τα αιτήματα μπορούν να εκδοθούν χωρίς να καθορίσουν το δέκτη (το σύνολο αντικειμένων που μπορεί να χειριστεί ένα αίτημα πρέπει να προσδιοριστεί δυναμικά). Το Πρότυπο αλυσίδας ευθύνης μπορεί να χρησιμοποιηθεί επίσης εάν τα αντικείμενα διανέμονται, ίσως, από κοινού με τα ακόλουθα πρότυπα [36].

8.5 Υπονοούμενη επίκληση (Implicit invocation)

Η υπονοούμενη επίκληση είναι ένας όρος που χρησιμοποιείται από ορισμένους συντάκτες ως στυλ της αρχιτεκτονικής λογισμικού, στο οποίο ένα σύστημα είναι δομημένο γύρω από το χειρισμό ενός γεγονότος, χρησιμοποιώντας μια μορφή της επανάκλησης. Συσχετίζεται πολύ με την Αντιστροφή του ελέγχου (Inversion of control), αυτό που είναι γνωστό ανεπίσημα ως Hollywood Principle [37].

Είναι μια από τις ευρύτερα αποδεκτές αρχιτεκτονικές μορφές στην τεχνολογία λογισμικού. Τα παραδείγματα των Implicit Invocation συστημάτων αφθονούν, συμπεριλαμβάνοντας όλα τα σύγχρονα λειτουργικά συστήματα, τα ενσωματωμένα περιβάλλοντα ανάπτυξης, και συστήματα διαχείρισης βάσεων δεδομένων [91].

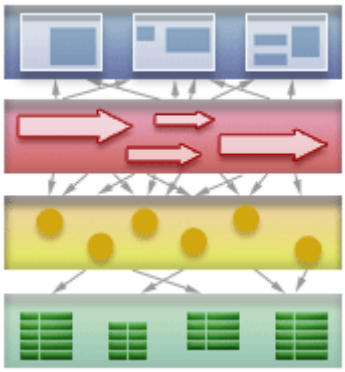
Οι Garland και Shaw περιέγραψαν τα συστήματα Υπονοούμενης επίκλησης (Implicit invocation) ως «Η ιδέα πίσω από την υπονοούμενη επίκληση είναι ότι αντί να κληθεί μια διαδικασία άμεσα, ένα συστατικό μπορεί να αναγγείλει ένα ή περισσότερα γεγονότα. Άλλα συστατικά στο σύστημα μπορούν να καταχωρήσουν ένα ενδιαφέρον για ένα γεγονός ενώνοντας μια διαδικασία με το γεγονός. Όταν το γεγονός αναγγέλλεται το ίδιο το σύστημα επικαλείται όλες τις διαδικασίες που έχουν καταχωρηθεί για το γεγονός. Κατά συνέπεια μια ανακοίνωση γεγονότος προκαλεί σιωπηρά την επίκληση των διαδικασιών σε άλλες ενότητες» [37] [91].
Τα υπονοούμενα συστήματα επίκλησης οδηγούνται από τα γεγονότα. Τα γεγονότα προκαλούνται, όποτε το σύστημα πρέπει να κάνει κάτι, όπως να ανταποκριθεί σε ένα εισερχόμενο αίτημα. Τα γεγονότα μπορούν να πάρουν πολλές μορφές στους διαφορετικούς τύπους εφαρμογών. Συχνά για τα συστήματα βασισμένα σε αντικείμενο, ένα γεγονός είναι ένα αντικείμενο, του οποίου οι ιδιότητες περιέχουν οποιεσδήποτε πληροφορίες που απαιτούνται για να επεξεργαστούν το γεγονός.

Όταν ένα γεγονός αναγγέλλεται, το σύστημα κοιτάζει επάνω στα τμήματα ακροατών (listeners) για εκείνο το γεγονός. Οι ακροατές ταιριάζουν τα ίδια κριτήρια για συστατικά που έχουν ήδη συζητηθεί - είναι λειτουργικές ενότητες του συστήματος. Τα συστατικά που επιθυμούν να ενεργήσουν ως ακροατές καταχωρούνται για να ακούσουν (listen) για ορισμένα γεγονότα στο χρόνο διαμόρφωσης. Όταν ένα γεγονός καλείται, όλοι οι εγγεγραμμένοι ακροατές εκείνου του γεγονότος περνούν το γεγονός με τη κλήση μιας δυναμικά καθορισμένης μεθόδου. Κατά αυτόν τον τρόπο, οι λειτουργίες επικαλούνται σιωπηρά. Αυτή η διαδικασία, όπου ανακοινώνεται στους ακροατές ένα γεγονός καλείται ανακοίνωση γεγονότος (Event announcement). Τα γεγονότα και οι ακροατές μπορούν οι ίδιοι να προκαλέσουν άλλα γεγονότα.
Οι υπονοούμενες αρχιτεκτονικές επίκλησης (Implicit invocation architectures) διαφέρουν από τα ρητά συστήματα επίκλησης (Explicit invocation systems) [46]. Τα συστατικά των συστημάτων υπονοούμενης επίκλησης χρησιμοποιούν γεγονότα για να επικοινωνήσουν το ένα με το άλλο. Οι σύνδεσμοι (Connectors) σε τέτοιες αρχιτεκτονικές είναι συσχετισμοί μεταξύ των γεγονότων και των μεθόδων συστατικών. Επειδή αυτοί οι συσχετισμοί καθορίζονται δυναμικά στο χρόνο εκτέλεσης, τα συστατικά συνδέονται αόριστα-χαλαρά. Ορισμένα συστατικά μπορεί να αντικατασταθούν από άλλα συστατικά [48]. Η χαλαρή σύζευξη προσφέρει στους αρχιτέκτονες λογισμικού το μεγάλο όφελος της αυξημένης ευελιξίας και της συντηρησιμότητας. Τα νέα συστατικά μπορούν να προστεθούν απλά αν καταχωρηθούν ως ακροατές γεγονότος. Επομένως η εξέλιξη των συστημάτων μπορεί να γίνει πιο εύκολα [47].
Τα αόριστα-χαλαρά συνδεμένα συστατικά λειτουργούν μαζί, αλλά δεν στηρίζονται το ένα στο άλλο για να κάνουν την εργασία τους. Η πολιτική αλληλεπίδρασης είναι χωριστή από τα αλληλεπιδρώντα συστατικά, παρέχοντας ευελιξία. Τα συστατικά μπορούν να εισαχθούν σε ένα σύστημα απλά με την καταχώρηση τους για τα γεγονότα του συστήματος, βοηθώντας πολύ στην ικανότητα επαναχρησιμοποίησης. Η εισαγωγή νέων συστατικών δεν απαιτεί την αλλαγή σε άλλες συστατικές διεπαφές, παρέχοντας εξελιξιμότητα, όπως πρόσθεση νέων χαρακτηριστικών γνωρισμάτων [45]. Γενικά, η υπονοούμενη επίκληση διευκολύνει την εξέλιξη των συστημάτων. Το

πρότυπο είναι ιδιαίτερα χρήσιμο για εφαρμογές που πρέπει να μπορούν να διαμορφωθούν όταν λειτουργούν [92].
Ως ένα πολύτιμο αρχιτεκτονικό πρότυπο, η σε γεγονός-βασισμένη υπονοούμενη επίκληση όχι μόνο υποστηρίζει τη χαλαρή σύζευξη μεταξύ των διαλογικών αντικειμένων, αλλά και συμπληρώνει τη βλάβη της υψηλής σύζευξης σε αντικειμενοστρεφή συστήματα [45].
Το μεγαλύτερο μειονέκτημα της υπονοούμενης επίκλησης είναι ότι τα συστατικά σταματούν τον έλεγχο του υπολογισμού που εκτελείται από το σύστημα. Όταν ένα συστατικό αναγγέλλει ένα γεγονός, δεν έχει καμία ιδέα ποια άλλα συστατικά θα ανταποκριθούν σε αυτό. Χειρότερα, ακόμα κι αν ξέρει ποια άλλα συστατικά ενδιαφέρονται για τα γεγονότα που αναγγέλλει, δεν μπορεί να στηριχθεί στη σειρά με την οποία καλούνται [48].
Ενώ η υπονοούμενη επίκληση έχει βρει την ευρεία χρήση σε άλλους τομείς της τεχνολογίας λογισμικού, δεν έχει χρησιμοποιηθεί σε μεγάλο βαθμό στις εφαρμογές Ιστού. Αλλά αυτό μπορεί να αλλάξει με τις έρευνες που βρίσκονται σε εξέλιξη. Παραδείγματα του προτύπου αυτού βρίσκονται στις εξής εργασίες [38] [39] [40] [41] [42] [43] [44].

8.6 Γυμνό Αντικείμενο (Naked Object)

Τα γυμνά αντικείμενα είναι ένα αρχιτεκτονικό πρότυπο με το οποίο τα αντικείμενα κύριας επιχείρησης, όπως ο πελάτης και το προϊόν, εκτίθενται άμεσα στο χρήστη αντί να καλύπτονται πίσω από τα συμβατικά κατασκευάσματα ενός ενδιάμεσου με τον χρήστη. Σε ένα επιχειρησιακό σύστημα που σχεδιάζεται χρησιμοποιώντας τα γυμνά αντικείμενα, όλες οι ενέργειες χρηστών περιλαμβάνουν την επίκληση των μεθόδων στα αντικείμενα επιχειρησιακών οντοτήτων [94].
Το πρότυπο γυμνών αντικειμένων περιγράφτηκε αρχικά στο PhD του Richard Pawson [49], το οποίο περιλαμβάνει μια λεπτομερή έρευνα για διάφορα προηγούμενα και εμπνεύσεις για το συγκεκριμένο πρότυπο, παραδείγματος χάριν, του ενδιάμεσου με τον χρήστη Morphic.
Τα περισσότερα νέα πρότυπα επιχειρησιακών συστημάτων εφαρμόζουν σιωπηρά το γενικό αρχιτεκτονικό πρότυπο τεσσάρων στρωμάτων που παρουσιάζεται στο σχήμα 8.6.1. Αυτό το πρότυπο τεσσάρων στρωμάτων καταγράφηκε αρχικά από τον Brown το 1995. Στο διάγραμμα τα τέσσερα στρώματα ονομάζονται παρουσίαση, ελεγκτής, αντικείμενο περιοχών, και διαχείριση δεδομένων. Σε μια δεδομένη εφαρμογή τα ονόματα των στρωμάτων μπορεί να διαφέρουν και τα τέσσερα κύρια στρώματα μπορούν να υποδιαιρεθούν σε επιπλέον στρώματα.
Χαρακτηριστική αρχιτεκτονική τεσσάρων-στρωμάτων [52] [95]:
Οι περισσότερες σύγχρονες εφαρμογές αποτελούνται από τέσσερα ή περισσότερα λογικά στρώματα. Ένα στρώμα παρουσίασης (presentation layer)(μπλε), ένα στρώμα ελεγκτή (controller layer)(κόκκινο), ένα πρότυπο στρώμα περιοχών (domain model layer) (κίτρινο) που αποτελούνται από τα αντικείμενα περιοχών ή/και τις υπηρεσίες επιχείρησης και ένα στρώμα διαχείρισης δεδομένων (data management)(πράσινο), όπως μια σχεσιακή βάση δεδομένων. Ακόμη και σε μια αποκαλούμενη αρχιτεκτονική «λεπτών πελατών», και τα τέσσερα στρώματα υπάρχουν, αλλά όλα στον κεντρικό υπολογιστή. Κάθε νέα διεργασία πρέπει να εφαρμοστεί σε τέσσερα επίπεδα.



Σχήμα 8.6.1 Απεικονίζονται τα τέσσερα στρώματα του αρχιτεκτονικού προτύπου, από τα οποία αποτελούνται οι περισσότερες εφαρμογές.

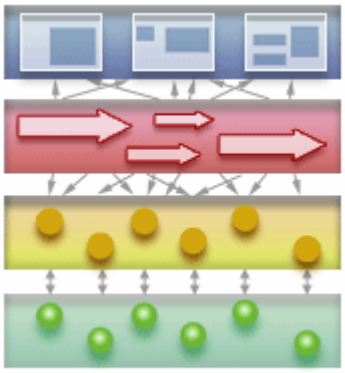
Σε αυτό το πρότυπο τεσσάρων στρωμάτων, μια ενιαία επιχειρησιακή έννοια (όπως ένας Πελάτης) θα αντιπροσωπευθεί συνήθως και στα τέσσερα στρώματα με διαφορετικές μορφές. Επιπλέον, όπως το σχήμα 8.6.1 δείχνει, οι σχέσεις μεταξύ των στοιχείων σε εκείνα τα τέσσερα στρώματα απαιτούν συχνά μια σύνθετη, πολλαπλή χαρτογράφηση. Αν και αυτό το γενικό αρχιτεκτονικό πρότυπο έχει εξελιχθεί κατά τη διάρκεια των ετών για να ικανοποιήσει ορισμένες ανάγκες, και αν και κάθε ένα από τα στρώματα μπορεί να είναι αντικειμενοστρεφές υπό κάποια έννοια, πολλές φορές χρειάζεται τροποποιήσεις.

Το στρώμα παρουσίασης, που επιτρέπει στο χρήστη να δει τα αντικείμενα και να επικαλεστεί τις συμπεριφορές σε αυτά, πρέπει να παρέχεται αυτόματα. Ο συντάκτης του προτύπου το έχει ονομάσει `γυμνό αντικείμενο', επειδή όσον αφορά στο χρήστη βλέπει και χειρίζεται τα γυμνά επιχειρησιακά αντικείμενα περιοχών.

Αυτά τα επιχειρησιακά αντικείμενα πραγματικά βρίσκονται σε ένα στρώμα αντικειμένου περιοχών της αρχιτεκτονικής, η οποία εφαρμόζεται συχνά σε μια κοινή πλατφόρμα κεντρικών υπολογιστών. Κατά συνέπεια ο χρήστης δεν βλέπει αυστηρά και δεν αλληλεπιδρά με τα επιχειρησιακά αντικείμενα, αλλά μάλλον με τις όψεις ή/και τους ελεγκτές που αντιστοιχούν σε εκείνα τα αντικείμενα και υπάρχουν σε ένα στρώμα παρουσίασης. Η έννοια των γυμνών αντικειμένων υπονοεί μια επικοινωνία μεταξύ των δύο στρωμάτων, έτσι η παραίσθηση του χειρισμού των επιχειρησιακών αντικειμένων είναι συνολική. Αυτό φαίνεται στο σχήμα 8.6.1.

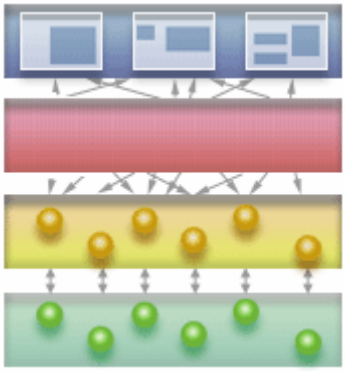
Χαρτογράφηση συγγενικού αντικειμένου:

Με τις συγγενικές τεχνολογίες χαρτογράφησης αντικειμένου (ORM-Object Relational Mapping, είναι τώρα δυνατό να παραχθεί αυτόματα το στρώμα διαχείρισης δεδομένων άμεσα από τον πρότυπο καθορισμό περιοχών, εξαλείφοντας κατά συνέπεια την ανάγκη να καθοριστεί και να διατηρηθεί ένα ολόκληρο στρώμα. Οι χρήσεις γυμνών αντικειμένων χρησιμοποιούν Hibernate για να εκτελέσουν αυτήν την λειτουργία. Το Hibernate είναι μια πολύ ισχυρή διαχείριση δεδομένων υψηλής απόδοσης [51]. Η πραγματική καινοτομία των γυμνών αντικειμένων, είναι ότι εφαρμόζουν την ίδια έννοια προς τα πάνω για να αποβάλουν τα δύο άλλα στρώματα.



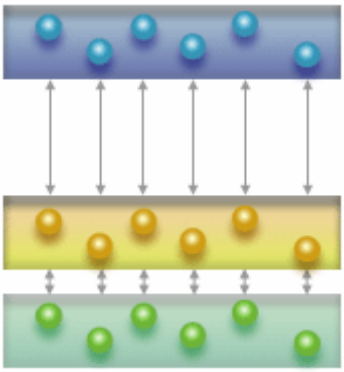
Σχήμα 8.6.2 Απεικονίζονται τα τέσσερα στρώματα του προτύπου, όπου το στρώμα διαχείρισης δεδομένων (πράσινο χρώμα) παρέχεται αυτόματα.

Πλήρη αντικείμενα περιοχών Συμπεριφοράς (Behaviourally):
Πρέπει να εξασφαλιστεί ότι όλη η επιχειρησιακή λογική είναι ενθυλακωμένη ως μέθοδοι στα αντικείμενα περιοχών. Κατά συνέπεια, ένα αντικείμενο Πελατών πρέπει όχι μόνο να είναι μια συλλογή των δεδομένων για τον πελάτη, πρέπει να ενθυλακώσει όλες τις συμπεριφορές, που πρέπει να εφαρμοστούν σε έναν πελάτη. Αυτό δεν είναι μια νέα ιδέα, είναι μόνο μια πολύ ισχυρή δέσμευση για την αρχή της ενθυλάκωσης αλλά οι περισσότεροι υπεύθυνοι για την ανάπτυξη εφαρμογής δεν είναι αποτελεσματικοί, με αποτέλεσμα το μεγαλύτερο μέρος να καταλήγει στους ελεγκτές. Τα (κίτρινα) αντικείμενα περιοχών αναπαρίστανται ως στερεές σφαίρες σε αυτό το διάγραμμα και παρακάτω.



Σχήμα 8.6.3 Στα τέσσερα στρώματα τα αντικείμενα περιοχών έχουν μετατραπεί σε στέρεες σφαίρες σε Πλήρη αντικείμενα περιοχών Συμπεριφοράς (κίτρινο χρώμα).

Γυμνά αντικείμενα:
Μόλις δημιουργηθούν τα πλήρη αντικείμενα περιοχών Συμπεριφοράς, είναι δυνατό αυτόματα να δημιουργηθεί ένα πλήρες ενδιάμεσο με τον χρήστη, απλά με την έκθεση των αντικειμένων περιοχών (με τις μεθόδους τους) άμεσα στο χρήστη, υπό μορφή αντικειμενοστρεφούς ενδιάμεσου με τον χρήστη. Στο χρόνο εκτέλεσης που το πλαίσιο γυμνών αντικειμένων επιθεωρεί την περιοχή αντικειμένων χρησιμοποιώντας αντανάκλαση (reflection) και καθιστά τα αντικείμενα και τις μεθόδους ορατά στο χρήστη. Γράφοντας μια εφαρμογή γυμνών αντικειμένων, αποτελείται απλώς από το γράψιμο του προτύπου αντικειμένου περιοχών. Αυτή η έννοια έχει γίνει γνωστή ως το αρχιτεκτονικό πρότυπο των γυμνών αντικειμένων. Το πλαίσιο γυμνών αντικειμένων είναι η κύρια εφαρμογή ανοικτού κώδικα αυτού του προτύπου.



Σχήμα 8.6.4 Με τα γυμνά αντικείμενα τα αντικείμενα περιοχών καθίστανται ορατά στο χρήστη μέσω ενός απολύτως γενικού στρώματος παρουσίασης. Όλη η απαραίτητη επιχειρησιακή λειτουργία πρέπει επομένως να ενθυλακωθεί στα αντικείμενα περιοχών.

Το πρότυπο γυμνού αντικειμένου καθορίζεται από τρεις αρχές [50] [93]:
1). Όλη η λογική πρέπει να ενσωματωθεί επάνω στα αντικείμενα περιοχών (domain objects). Αυτή η αρχή δεν είναι μοναδική στα γυμνά αντικείμενα, είναι μια ισχυρή δέσμευση για την ενθυλάκωση.
2). Το ενδιάμεσο με τον χρήστη πρέπει να είναι μια άμεση αναπαράσταση των αντικειμένων περιοχών, με όλες τις ενέργειες χρηστών που αποτελούνται, από τη δημιουργία ή την ανάκτηση των αντικειμένων περιοχών ή/και την επίκληση των μεθόδων σε εκείνα τα αντικείμενα. Αυτή η αρχή δεν είναι μοναδική στα γυμνά αντικείμενα. Είναι μια συγκεκριμένη ερμηνεία ενός αντικειμενοστρεφούς ενδιάμεσου με τον χρήστη.
Η αρχική ιδέα στο σχέδιο γυμνού αντικειμένου προκύπτει από το συνδυασμό αυτών των δύο, για να διαμορφώσει την τρίτη αρχή.
3). Το ενδιάμεσο με τον χρήστη πρέπει να δημιουργηθεί 100% αυτόματα από τον καθορισμό των αντικειμένων περιοχών. Αυτό μπορεί να γίνει χρησιμοποιώντας διάφορες διαφορετικές τεχνολογίες, συμπεριλαμβανομένης της παραγωγής πηγαίου κώδικα (source code generation).

Τα οφέλη των γυμνών αντικειμένων είναι [50] [93]:
1). Γίνεται γρηγορότερος κύκλος ανάπτυξης, επειδή υπάρχουν λιγότερα στρώματα που αναπτύσσονται. Σε ένα συμβατικότερο σχεδιασμό, ο υπεύθυνος για την ανάπτυξη πρέπει να καθορίσει και να εφαρμόσει τρία ή περισσότερα χωριστά στρώματα: το στρώμα αντικειμένου περιοχών, το στρώμα παρουσίασης, και τα χειρόγραφα στόχου ή διαδικασίας που συνδέουν τα δύο.
2). Μεγαλύτερη ευκινησία, που αναφέρεται στην ευκολία με την οποία μια εφαρμογή μπορεί να αλλαχτεί για να προσαρμόσει τις μελλοντικές αλλαγές στις επιχειρησιακές απαιτήσεις. Εν μέρει αυτό προκύπτει από τη μείωση του αριθμού αναπτυγμένων στρωμάτων που πρέπει να κρατηθούν συγχρονισμένα. Επίσης η ένα προς ένα επικοινωνία μεταξύ της παρουσίασης χρηστών και του προτύπου περιοχών, αναγκάζει τη μοντελοποίηση αντικειμένου υψηλότερης ποιότητας, η οποία βελτιώνει στη συνέχεια την ευκινησία.
3). Ένα πιο δυνατό στυλ ενδιάμεσου με τον χρήστη. Αυτό το όφελος αποδίδεται πραγματικά στο προκύπτον αντικειμενοστρεφές ενδιάμεσο με τον χρήστη, παρά στα γυμνά αντικείμενα, αν και προβάλλεται ότι τα γυμνά αντικείμενα το καθιστούν πολύ ευκολότερο να συλλάβουν και να εφαρμόσουν ένα αντικειμενοστρεφές ενδιάμεσο με τον χρήστη.

4). Ευκολότερη ανάλυση απαιτήσεων. Με το πρότυπο γυμνών αντικειμένων, τα αντικείμενα περιοχών διαμορφώνουν μια κοινή γλώσσα μεταξύ των χρηστών και των υπεύθυνων για την ανάπτυξη και ότι αυτή η κοινή γλώσσα διευκολύνει τη όλη διαδικασία. Συνδυασμένο με ένα γρήγορο κύκλο ανάπτυξης, είναι δυνατό να χειριστεί λειτουργικές εφαρμογές πρωτοτύπων σε πραγματικό χρόνο.

8.7 Πρότυπο (Μαυρο)πίνακα (Blackboard Pattern)

Το πρότυπο πίνακα είναι χρήσιμο για τα προβλήματα όπου καμία μη ντετερμινιστική στρατηγική λύσης δεν είναι γνωστή [64]. Στον πίνακα διάφορα εξειδικευμένα υποσυστήματα συγκεντρώνουν τη γνώση τους για να χτίσουν μια μερική ή κατά προσέγγιση λύση [53] [54] [59].

Η ιδέα πίσω από την αρχιτεκτονική πινάκων είναι μια συλλογή ανεξάρτητων προγραμμάτων που λειτουργούν συνεταιριστικά σε μια κοινή δομή δεδομένων. Κάθε πρόγραμμα είναι εξειδικευμένο για την επίλυση ενός ιδιαίτερου μέρους του γενικού στόχου, και όλα τα προγράμματα λειτουργούν μαζί για τη λύση. Αυτά τα εξειδικευμένα προγράμματα είναι ανεξάρτητα το ένα από το άλλο. Δεν καλεί το ένα το άλλο, ούτε υπάρχει μια προκαθορισμένη σειρά για την ενεργοποίησή τους. Αντ' αυτού, η κατεύθυνση που λαμβάνεται από το σύστημα καθορίζεται κυρίως από τη τρέχουσα κατάσταση της προόδου. Ένα κεντρικό τμήμα ελέγχου αξιολογεί την τρέχουσα κατάσταση της επεξεργασίας και συντονίζει τα εξειδικευμένα προγράμματα. Αυτό το στοιχείο-κατευθυνόμενο καθεστώς ελέγχου καθιστά τον πειραματισμό με τους διαφορετικούς αλγορίθμους πιθανό [54] [97].

Έστω ότι υπάρχει ήδη ένα σύνολο πρακτόρων που είναι ειδικευμένοι για να εκτελέσουν μια συγκεκριμένη δευτερεύουσα υποχρέωση ενός γενικού στόχου. Πρέπει να παραχθεί ένα μέσο που επιτρέπει σε αυτούς να ελέγξουν ο ένας τον άλλον και να έχουν μια αμοιβαία πρόοδο. Το πρόβλημα είναι πώς θα εξασφαλιστεί η συνοχή μιας ομάδας ειδικευμένων πρακτόρων.

Οι πράκτορες πρέπει να συνεργαστούν να εκτελέσουν τους σύνθετους στόχους που επεκτείνονται πέρα από τις μεμονωμένες ικανότητές τους. Ένας τρόπος να επιτραπεί η συνεργασία είναι να είναι καταγεγραμμένες οι σχέσεις μεταξύ των πρακτόρων, παραδείγματος χάριν, μέσω ενός καταλόγου σε κάθε πράκτορα. Εντούτοις, αυτό είναι δύσκολο να επιτευχθεί εάν οι θέσεις των συνεργαζόμενων πρακτόρων δεν καθορίζονται, εάν μερικοί πράκτορες δεν έχουν δημιουργηθεί όταν ένας πράκτορας θέλει να αλληλεπιδράσει με αυτούς, ή εάν οι πράκτορες είναι κινητοί.

Οι πράκτορες μπορεί να είχαν σχεδιαστεί ανεξάρτητα. Σε αυτήν την περίπτωση, θα μπορούσε να είναι δύσκολο να επιβληθεί ένας κοινός τρόπος να αναπαρασταθούν οι σχέσεις μεταξύ των πρακτόρων που κάθε πράκτορας πρέπει να εφαρμόσει. Θα ήταν καταλληλότερο εάν οι πράκτορες δεν έκαναν υποθέσεις για το ποιοι συγκεκριμένοι πράκτορες θα είναι διαθέσιμοι για να συνεργαστούν και χτίστηκαν με την έννοια ότι θα υπάρξουν μερικοί πράκτορες διαθέσιμοι, άγνωστοι στον πράκτορα στο χρόνο σχεδιασμού.

Το πρωτόκολλο συντονισμού που απαιτείται για τη συνεργασία πρακτόρων μπορεί να εκφραστεί χρησιμοποιώντας τους μηχανισμούς πρόσβασης δεδομένων του μέσου συντονισμού. Αυτό σημαίνει ότι η λογική συντονισμού ενσωματώνεται στους πράκτορες. Αν και ένας λογικός χωρισμός μεταξύ των αλγοριθμικών και ζητημάτων συντονισμού θα παρείχε περισσότερη ευελιξία, το κόστος ενός πιο σύνθετου μέσου συντονισμού είναι υψηλό. Διατηρώντας την διεπαφή στο μέσο συντονισμού μικρή κάνει τους πράκτορες εύκολα να χρησιμοποιήσουν ένα άλλο μέσο συντονισμού.

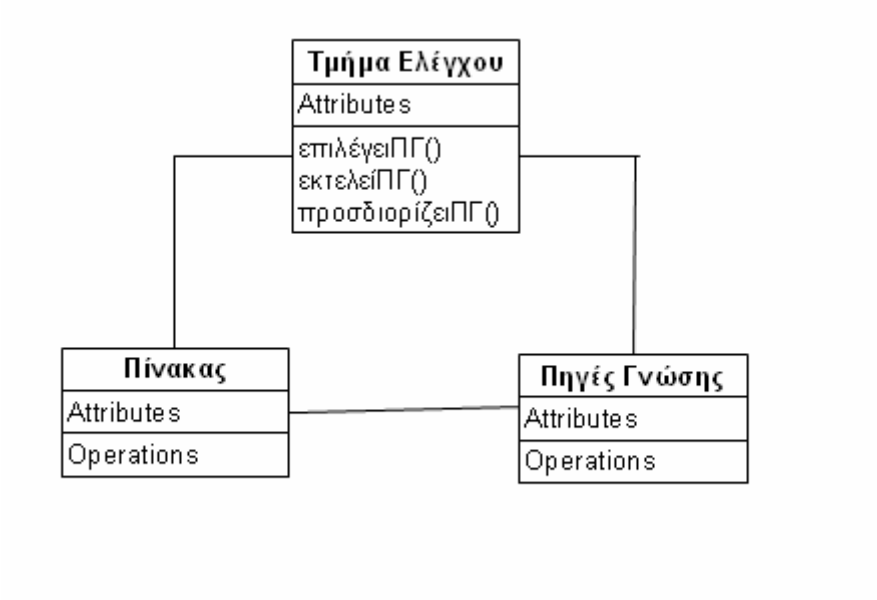
Η λύση στο πρόβλημα περιλαμβάνει έναν πίνακα στον οποίο οι πράκτορες μπορούν να προσθέσουν δεδομένα και που τους επιτρέπει να προσυπογράψουν για τις αλλαγές δεδομένων στον τομέα ενδιαφέροντος τους. Οι πράκτορες μπορούν επίσης να ενημερώσουν τα δεδομένα και να τα σβήσουν από τον πίνακα. Οι πράκτορες ελέγχουν συνεχώς τον πίνακα για αλλαγές και κάνουν σήμα όταν θέλουν να προσθέσουν, να σβήσουν ή να ενημερώσουν τα δεδομένα [61].
Όταν οι πολλοί πράκτορες θέλουν να ανταποκριθούν σε μια αλλαγή, ένας πράκτορας επόπτης (supervisor) αποφασίζει ποιος ειδικός πράκτορας μπορεί να κάνει μια τροποποίηση στον πίνακα. Ο πράκτορας επόπτης αποφασίζει επίσης πότε η κατάσταση του πίνακα έχει προχωρήσει αρκετά και μια λύση στο σύνθετο στόχο έχει βρεθεί. Ο επόπτης ενεργεί μόνο ως χρονοπρογραμματιστής για τους ειδικούς πράκτορες και αποφασίζουν εάν και πότε θα αφήσει έναν ειδικό πράκτορα να τροποποιήσει τον πίνακα. Δεν διευκολύνει την αλληλεπίδρασή τους του ενός με τον άλλον.

Η λογική του προτύπου είναι η εξής. Το πρότυπο πινάκων αποσυνδέει τους πράκτορες που αλληλεπιδρούν μεταξύ τους. Αντί να επικοινωνήσουν άμεσα, οι πράκτορες αλληλεπιδρούν μέσω της μεσολάβησης ενός ενδιάμεσου πράκτορα. Αυτός ο μεσάζων παρέχει και το χρόνο και τη διαφάνεια θέσης στους αλληλεπιδρώντας πράκτορες. Η διαφάνεια του χρόνου επιτυγχάνεται δεδομένου ότι οι πράκτορες που θέλουν να ανταλλάξουν τα δεδομένα δεν είναι απαραίτητο να λάβουν τα δεδομένα όταν στέλνονται αλλά μπορούν να τα επιλέξουν αργότερα. Οι θέσεις των λαμβανόντων πρακτόρων είναι διαφανείς δεδομένου ότι ο πράκτορας που στέλνει δεν πρέπει να απευθυνθεί σε οποιοδήποτε άλλο πράκτορα, συγκεκριμένα από το όνομά του. Ο ενδιάμεσος πράκτορας διαβιβάζει τα δεδομένα αναλόγως.

Η κινητικότητα (mobility) υποστηρίζεται επίσης καλά από αυτό το πρότυπο μόλις γίνει μια μικρή επέκταση στο πρωτόκολλο μεταξύ των ειδικών πρακτόρων και του πίνακα. Δεδομένου ότι η ανταλλαγή των δεδομένων είναι ασύγχρονη, οι πράκτορες μπορούν να αφήσουν δεδομένα άλλους πράκτορες σε έναν πίνακα. Οι πράκτορες που φθάνουν αφού έχει φύγει ο δημιουργός των δεδομένων μπορούν ακόμα να διαβάσουν τα δεδομένα από τον πίνακα. Μια μικρή αλλαγή απαιτείται στο πρωτόκολλο μεταξύ των ειδικών πρακτόρων και του πίνακα. Όταν φτάσει σε μια θέση που υπάρχει ένας πίνακας, ένας πράκτορας προσυπογράφει σε όλους τους τομείς ενδιαφέροντος. Επιπλέον, ο πίνακας πρέπει να ειδοποιήσει τον πράκτορα για οποιοδήποτε δεδομένο που στάλθηκε στον πίνακα πριν προσυπογράφηκε ο πράκτορας.

Ο πίνακας είναι ένα παράδειγμα ενός παθητικού μέσου συντονισμού. Ενώ επιτρέπει στους πράκτορες για να μοιραστούν τα δεδομένα, δεν διευκρινίζει πώς οι πράκτορες αναμένονται να αντιδράσουν στα δεδομένα που λαμβάνουν. Με άλλα λόγια, όλη η πραγματική γνώση συντονισμού παραμένει κρυμμένη στους πράκτορες. Ο λόγος για αυτό είναι ότι τα πρωτόκολλα συντονισμού πρέπει να εκφραστούν χρησιμοποιώντας τη διεπαφή πρόσβασης δεδομένων σε έναν πίνακα. Εάν δεν είναι δυνατό να εκφραστεί ένα πρωτόκολλο συντονισμού κατά αυτόν τον τρόπο, οι πράκτορες αναγκάζονται να εφαρμόσουν το πρωτόκολλο συντονισμού στον κώδικά τους.

Η δομή του προτύπου παρουσιάζεται στο σχήμα 8.7.1 [4] [54] [58] [60] [62] [97].



Σχήμα 8.7.1 Απεικονίζεται το πρότυπο πίνακα.

Το τμήμα πινάκων παρέχει ένα κεντρικό κατάστημα στοιχείων για όλες τις υποθέσεις που δημιουργούνται κατά τη διάρκεια της ανάλυσης. Οι κύριες λειτουργίες είναι:

1). Αποθήκευση τρέχουσας κατάστασης καταστημάτων κατά τη διάρκεια της λύσης.

2). Αποθηκεύει όλα τα δεδομένα ελέγχου.

3). Παραδίδει τα ζητούμενα δεδομένα.

4). Δεδομένου ότι ο πίνακας είναι ουσιαστικά μια κοινή αποθήκη μνήμης, οι τεχνικές συγχρονισμού μπορούν να απαιτηθούν για τους πολλούς συγγραφείς και τους αναγνώστες.

Τα τρία συστατικά του προτύπου είναι :

1). Ο πίνακας είναι μια δομημένη σφαιρική μνήμη που περιέχει τα αντικείμενα από το διάστημα λύσης. Αυτά τα αντικείμενα λέγονται επίσης και κόμβοι πινάκων. Οργανώνονται ιεραρχικά σε επίπεδα ανάλυσης και μπορούν να συνδεθούν το ένα με το άλλο.

2). Οι πηγές γνώσης (είναι τα δεδομένα) μπορούν να εννοηθούν ως ειδικευμένες ενότητες με την αναπαράστασή τους. Χαρακτηρίζονται από ένα σύνολο όρων και έναν εκτελέσιμο κώδικα που ανακτά τα δεδομένα από τον πίνακα και προσθέτει τη συμβολή του στο πρόβλημα.

3). Το τμήμα ελέγχου (επόπτης) επιλέγει, προσδιορίζει και εκτελεί τις πηγές γνώσης. Ο καθορισμός των εκτελέσιμων πηγών γνώσης είναι βασισμένος στην κατάσταση της διαδικασίας επίλυσης προβλήματος όπως εκφράζεται στον πίνακα.

Το πρότυπο πινάκων παρέχει αποτελεσματική λύση στο σχεδιασμό και εφαρμογή πολυσύνθετων συστημάτων όπου οι ετερογενείς ενότητες πρέπει να συνδυαστούν δυναμικά για να λύσουν ένα πρόβλημα. Εξασφαλίζει επίσης πολύ σημαντικές μη λειτουργικές ιδιότητες όπως [60]:

Επαναχρησιμοποίηση. Οι πηγές γνώσης είναι ανεξάρτητοι ειδικοί που μπορούν να επαναχρησιμοποιηθούν σε διαφορετικές εργασίες. Η επαναχρησιμοποίηση προγραμμάτων γίνεται ευκολότερη από το γεγονός ότι δεν υπάρχει καμία άμεση επικοινωνία μεταξύ των πηγών γνώσης.

Υψηλό επίπεδο διαφοροποίησης και συντηρησιμότητας. Υψηλό επίπεδο διαχωρισμού του ελέγχου και των πηγών γνώσης διευκολύνει πολύ την συντηρησιμότητα [64].

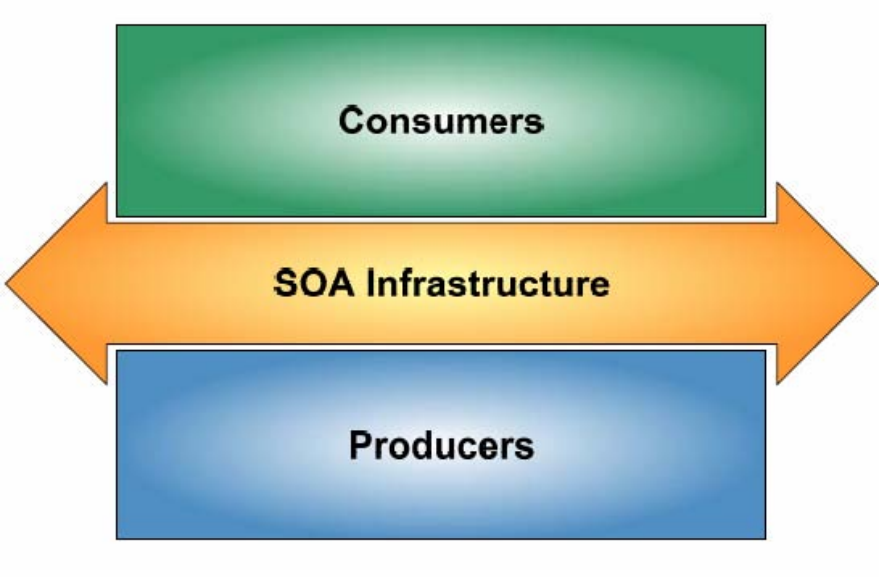
Το μεγαλύτερο μειονέκτημα είναι η δυσκολία καθορισμού μιας καλής στρατηγικής ελέγχου. Ο έλεγχος βασίζεται σε heuristics, που μπορούν να ληφθούν μέσα από πειράματα [63].

Αυτό το πρότυπο έχει χρησιμοποιηθεί σε πολλά συστήματα. Ένα από αυτά είναι το σύστημα Hearsay-II. Αυτός ο στόχος συστημάτων λεκτικής αναγνώρισης ήταν να απαντήσει ερωτήσεις για τα έγγραφα και μετά να ανακτούν τα έγγραφα από μια βάση δεδομένων λογοτεχνίας [55]. Επίσης έχει χρησιμοποιηθεί στο σύστημα HASP/SIAP. Αυτό το σύστημα ανίχνευσης εχθρικών υποβρύχιων χρησιμοποιούσε σειρές υδροτηλεφώνου (hydrophone) για να ελέγξει μια περιοχή στη θάλασσα συλλέγοντας sonar σήματα. Η ερμηνεία των σημάτων έγινε από το σύστημα πίνακα [56]. Το σύστημα CRYSALIS από τα δεδομένα διάθλασης της ακτίνας X, αυτό το σύστημα συμπέρανε την τρισδιάστατη δομή των πρωτεϊνικών μορίων [57].

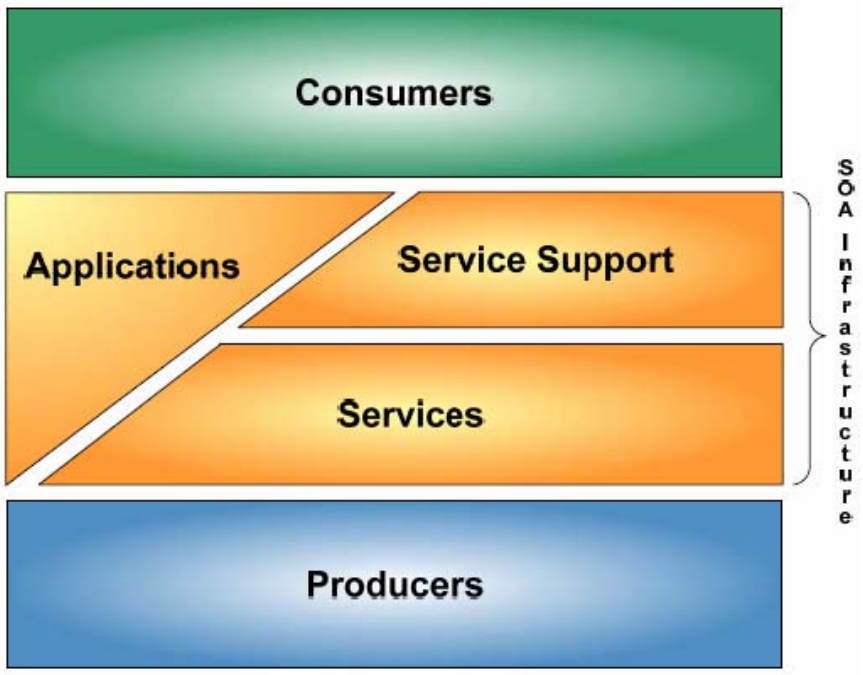
8.8 Προσανατολισμένη στις υπηρεσίες Αρχιτεκτονική (Service Oriented Architecture-SOA)

Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) είναι ένα παράδειγμα για την οργάνωση και τη χρησιμοποίηση των διανεμημένων ικανοτήτων που μπορούν να είναι υπό τον έλεγχο των διαφορετικών περιοχών ιδιοκτησίας και εφαρμόζονται χρησιμοποιώντας διάφορους σωρούς τεχνολογίας. Γενικά, οι οντότητες (άνθρωποι και οργανώσεις) δημιουργούν τις ικανότητες για να λύσουν ή να υποστηρίξει μια λύση για τα προβλήματα που αντιμετωπίζουν κατά τη διάρκεια της εργασίας τους. Είναι φυσικό να σκεφτεί τις ανάγκες ενός ατόμου που ικανοποιούνται από τις ικανότητες που προσφέρονται από κάποιο άλλο ή στον κόσμο του διανεμημένου υπολογισμού, οι απαιτήσεις ενός πράκτορα υπολογιστών καλύπτονται από έναν πράκτορα υπολογιστών που ανήκει σε έναν διαφορετικό ιδιοκτήτη. Ο όρος ιδιοκτήτης μπορεί εδώ να χρησιμοποιηθεί για να δείξει τα διαφορετικά τμήματα μιας επιχείρησης ή ανεξάρτητες οντότητες στις διαφορετικές χώρες. Δεν υπάρχει απαραίτητα συσχετισμός ένα προς ένα μεταξύ των αναγκών και των ικανοτήτων. Οι ανάγκες και οι ικανότητες ποικίλλουν από θεμελιώδεις σε σύνθετες και οποιαδήποτε ανάγκη μπορεί να απαιτήσει έναν συνδυασμό πολυάριθμων ικανοτήτων ενώ οποιαδήποτε ενιαία ικανότητα μπορεί να καλύψει περισσότερες από μια ανάγκες [68].
Η προσανατολισμένη προς τις υπηρεσίες αρχιτεκτονική (SOA) περιγράφτηκε αρχικά από τον Gartner το 1996 [65] [66], αλλά το ενδιαφέρον για την αρχιτεκτονική έχει επικεντρωθεί πρόσφατα στην εμφάνιση των Υπηρεσιών Ιστού (Web Services) [67]. Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) θεωρείται αυτήν την περίοδο ως επόμενο βήμα για τις αρχιτεκτονικές λογισμικού, που ικανοποιούν τη συγκέντρωση των συστημάτων λογισμικού σε πιο σύνθετες λύσεις με λιγότερες δαπάνες [69].
Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική είναι ένα αρχιτεκτονικό παράδειγμα που μπορεί να χρησιμοποιηθεί για να χτιστούν υποδομές επιτρέποντας σε εκείνους με τις ανάγκες (καταναλωτές) και εκείνους με τις ικανότητες (προμηθευτές) να αλληλεπιδράσουν μέσω των υπηρεσιών στις περιοχές της τεχνολογίας και της ιδιοκτησίας. Οι υπηρεσίες ενεργούν ως βοηθοί του πυρήνα των ηλεκτρονικών δεδομένων και απαιτούν πρόσθετους μηχανισμούς προκειμένου να λειτουργήσουν. Στο έγγραφο με τίτλο ‘‘Service Oriented architecture’’ [70] χαρακτηριστικά αναφέρεται: ‘’ Η αρχιτεκτονική παρέχει ένα νέο τρόπο όχι μόνο για να μετατραπεί η επιχείρησή, αλλά για να συνδεθεί με τους προμηθευτές, τους συνεργάτες και τους πελάτες’’. Διάφορες νέες τάσεις στη βιομηχανία υπολογιστών στηρίζονται επάνω σε αυτήν την αρχιτεκτονική SOA. Αυτές περιλαμβάνουν την αυτοματοποίηση της επιχειρησιακής διαχείρισης διαδικασιών (BPM), των σύνθετων εφαρμογών (εφαρμογές που αθροίζουν πολλές υπηρεσίες να λειτουργήσουν), και του πλήθους

νέων αρχιτεκτονικών και σχεδιαστικών προτύπων που αναφέρονται ως Ιστός 2.0 (Web 2.0) [68].
Η SOA είναι μια αρχιτεκτονική έννοια, και ως εκ τούτου για να πραγματοποιήσει μια λύση SOA, πρέπει να χαρτογραφήσει την αρχιτεκτονική σε ένα λογικό κατασκεύασμα που ακολουθείται από μια εφαρμογή, χρησιμοποιώντας ένα συγκεκριμένο σύνολο τεχνολογιών/προϊόντων/πλατφορμών. Όπως με οποιαδήποτε λύση, μια SOA χαρακτηρίζεται από ένα σύνολο και υποχρεωτικών και προαιρετικών συστατικών. Μια λύση SOA αποτελείται από τα ακόλουθα τρία κύρια λογικά συστατικά που παρουσιάζονται στο σχήμα 8.8.1 [77].



Σχήμα 8.8.1 Απεικονίζονται τα βασικά λογικά συστατικά της SOA.
Το στρώμα υποδομής SOA μπορεί να διαιρεθεί σε τρία μικρότερα συστατικά, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 8.8.2 Απεικονίζονται τα τρία υποσυστατικά του στρώματος υποδομής της SOA.

Οπότε τα πέντε κύρια συστατικά του SOA είναι [71] [77]:

1). Καταναλωτές. Μια οντότητα που χρησιμοποιεί την υπηρεσία που προσφέρεται από τον παραγωγό.
2). Εφαρμογές. Παρέχουν μια γραφική διεπαφή και μια ποικιλία βαθμών της στενά συνδεμένης επιχειρησιακής λογικής για τους καταναλωτές για να εκτελεστούν οι στόχοι.
3). Υπηρεσίες. Μια οντότητα που εκτελεί έναν συγκεκριμένο στόχο όταν επικαλείται.
4). Υποστήριξη υπηρεσίας. Μια οντότητα που παρέχει λειτουργίες υποστήριξης υποβάθρου για τη SOA.
5). Παραγωγοί. Μια οντότητα που προσφέρει μια συγκεκριμένη υπηρεσία ή μια λειτουργία.
Κάθε ένα από αυτά τα συστατικά περιέχει πρόσθετα μικρά υποσυστατικά.
Η προσανατολισμένη στις υπηρεσίες αρχιτεκτονική (SOA) θεωρείται ως το επόμενο βήμα για τις αρχιτεκτονικές λογισμικού, που ικανοποιούν τα συστήματα λογισμικού με πιο σύνθετες λύσεις με τις δαπάνες να κυμαίνονται σε λογικά επίπεδα.
Η SOA παρέχει ένα απλό εξελικτικό παράδειγμα για την οργάνωση των μεγάλων δικτύων συστημάτων που απαιτούν διαλειτουργικότητα. Η SOA είναι εξελικτική επειδή κάνει τις πιο λίγες πιθανές υποθέσεις για το δίκτυο και ελαχιστοποιεί επίσης οποιεσδήποτε υποθέσεις εμπιστοσύνης που συχνά γίνονται στα συστήματα μικρότερης κλίμακας.

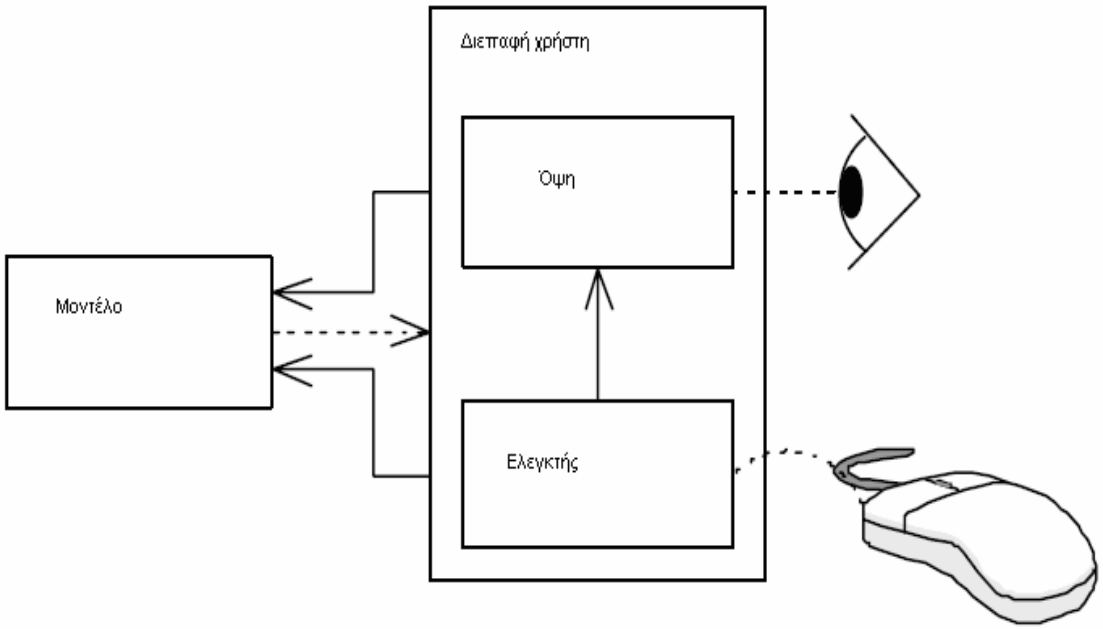
8.9 Μοντέλο-Όψη-Ελεγκτής (Model View Controller-MVC)

Το MVC περιγράφηκε αρχικά το 1979 από τον Trygve Reenskaug στο Xerox PARC [73]. Η αρχική εφαρμογή περιγράφεται σε βάθος στο έγγραφο Applications in Smalltalk-80: How to use Model-View-Controller [74]. Σήμερα το πρότυπο υποστηρίζεται από τη βιβλιοθήκη Swing της γλώσσας προγραμματισμού Java [83]. Έχουν υπάρξει διάφορα παράγωγα του MVC μια από τα πιο γνωστά, λόγω της χρήσης του από τη Microsoft είναι το πρότυπο Μοντέλο- Όψη-Παρουσιαστής (Model View Presenter) που εμφανίστηκε στις αρχές της δεκαετίας του 1990 και είχε ως σκοπό να είναι μια εξέλιξη του MVC. Εντούτοις το πρότυπο μοντέλο-όψη-ελεγκτής παραμένει ακόμα ευρέως χρησιμοποιημένο. Το Νοέμβριο 2002 η W3C ψήφισε να κάνει MVC δομές μέρος της αρχιτεκτονικής XForms για όλες τις μελλοντικές εφαρμογές Ιστού [75]. Αυτές οι προδιαγραφές θα ενσωματωθούν τώρα άμεσα στις προδιαγραφές XHTML 2.0. Υπάρχουν τώρα πάνω από 20 προμηθευτές που υποστηρίζουν τα πλαίσια XForms με MVC που ενσωματώνονται στην εφαρμογή.
Το MVC αναπτύχθηκε προκειμένου να υποστηρίξει συγκεκριμένους στόχους. Ένας από αυτούς ήταν η απαίτηση για υποστήριξη προσαρμογών σε μια διεπαφή έτσι ώστε αφενός η διεπαφή να είναι κατάλληλη για διαφορετικούς χρήστες και αφετέρου η ίδια πληροφορία να μπορεί να παρουσιάζεται κατάλληλα σε διαφορετικές μονάδες παρουσίασης. Επίσης οι προσαρμογές αυτές θα πρέπει να μπορούν να εφαρμόζονται εύκολα και γρήγορα, ακόμη και κατά τη διάρκεια της εκτέλεσης μιας εφαρμογής. Οι τρόποι παρουσίασης δεν πρέπει να επηρεάζουν την λειτουργικότητα της εφαρμογής. Ένας ακόμη στόχος ήταν η υποστήριξη του πλήρους διαχωρισμού μεταξύ επεξεργασίας, εισόδου, εξόδου δεδομένων σε μια διεπαφή.
Το MVC είναι ένα αρχιτεκτονικό πρότυπο που χρησιμοποιείται στην τεχνολογία λογισμικού. Η επιτυχημένη χρήση του προτύπου απομονώνει την επιχειρησιακή λογική από τις εκτιμήσεις ενδιάμεσων με τον χρήστη, με αποτέλεσμα μια εφαρμογή όπου είναι ευκολότερο να τροποποιηθεί είτε η οπτική εμφάνιση της εφαρμογής είτε οι κανόνες της χωρίς να επηρεάζει άλλες. Το μοντέλο αντιπροσωπεύει τις πληροφορίες

(τα δεδομένα) της εφαρμογής, η όψη αντιστοιχεί στα στοιχεία του ενδιάμεσου με τον χρήστη όπως το κείμενο και ο ελεγκτής διαχειρίζεται την επικοινωνία των δεδομένων και των επιχειρησιακών κανόνων που χρησιμοποιούνται για να χειριστούν τα δεδομένα από και προς το μοντέλο [72]. Είναι ένα ευρέως υιοθετημένο πρότυπο, σε πολλές γλώσσες και πλαίσια εφαρμογής, ο σκοπός των οποίων είναι να επιτευχθεί ένας καθαρός διαχωρισμός μεταξύ τριών συστατικών των περισσότερων εφαρμογών Ιστού.

Το πρότυπο MVC λειτουργεί ως εξής. Ο χρήστης αλληλεπιδρά με το ενδιάμεσο του χρήστη με κάποιον τρόπο, για παράδειγμα πατάει ένα κουμπί του ποντικιού. Μετά ο ελεγκτής χειρίζεται το γεγονός εισόδου από το ενδιάμεσο με το χρήστη, συχνά μέσω ενός χειριστή ή μιας επανάκλησης. Ο ελεγκτής ειδοποιεί το μοντέλο για την ενέργεια του χρήστη, έχοντας ως αποτέλεσμα μια αλλαγή στην κατάσταση του μοντέλου. Μια όψη χρησιμοποιεί το μοντέλο για να παράγει έμμεσα ένα κατάλληλο ενδιάμεσο με το χρήστη. Η όψη παίρνει τα δεδομένα της από το μοντέλο. Το μοντέλο και ο ελεγκτής δεν έχουν άμεση γνώση της όψης. Το ενδιάμεσο με το χρήστη περιμένει για επόμενες αλληλεπιδράσεις, κάτι το οποίο ξαναρχίζει τον κύκλο από την αρχή.

Το πρότυπο MVC διαιρεί ένα σύστημα σε τρία ξεχωριστά μέρη όπως φαίνεται στο παρακάτω σχήμα [78] [79] [80] [82].



Σχήμα 8.9.1 Απεικονίζονται τα τρία μέρη του προτύπου MVC.

Τα τρία μέρη του προτύπου MVC αναλύονται παρακάτω [76] [78] [81][84]:

1). Μοντέλο. Αυτό το τμήμα αφορά τη λειτουργικότητα και τα δεδομένα της εφαρμογής. Ενσωματώνει και διαχειρίζεται τα δεδομένα μιας εφαρμογής υποστηρίζοντας μεθόδους ενημέρωσης δεδομένων, τα οποία μπορεί να καλέσει ο ελεγκτής και μεθόδους πρόσβασης σε καταστάσεις τις οποίες μπορούν να απαιτήσουν όψης και ελεγκτή. Δηλαδή το μοντέλο διατηρεί έναν κατάλογο εξαρτημένων όψεων και ελεγκτών, που ενημερώνονται σχετικά με ενδεχόμενες αλλαγές στα δεδομένα.

2). Όψη. Αυτό παρέχει μια οπτική ή μη οπτική αντιπροσώπευση του προτύπου. Είναι ένας μηχανισμός αντιστοίχησης δεδομένων ενός μοντέλου σε προβολές που γίνονται αντιληπτές μέσω μιας συσκευής εξόδου. Υπάρχουν συχνά περισσότεροι από ένας ελεγκτές όψης. Χαρακτηριστικά, ο χρήστης πρέπει να είναι σε θέση να δει, ή να έχει μια όψη, για το τι κάνει το πρόγραμμα . Η όψη είναι παθητικός παρατηρητής και δεν

πρέπει να έχει επιπτώσεις στο πρότυπο. Το πρότυπο πρέπει να είναι ανεξάρτητο από την όψη. Όταν ένα συγκεκριμένο γνώρισμα αλλάξει κατάσταση, η αλλαγή γνωστοποιείται σε όλες τις ενδιαφερόμενες όψεις, οι οποίες ενημερώνουν την παρουσίασή τους.
3). Ελεγκτής. Ο ελεγκτής εφαρμόζει τη λογική ελέγχου που χαρτογραφεί τις ενέργειες σε αλλαγές στο πρότυπο. Ο ελεγκτής αποφασίζει τι είναι να κάνει το πρότυπο. Ο ελεγκτής και το πρότυπο μπορούν σχεδόν πάντα να χωριστούν. Ο ελεγκτής εξαρτάται από το μοντέλο. Το μοντέλο δεν πρέπει να εξαρτάται από τον ελεγκτή. Το τμήμα ελεγκτή καθορίζει και διαχειρίζεται τον τρόπο που η όψη αντιδρά στην εισαγωγή του χρήστη. Όταν ένα δεδομένο γεγονός συμβαίνει σε μια όψη, ο ελεγκτής είναι αυτός που πιάνει το γεγονός και αποφασίζει τι θα συμβεί μετά. Μερικές φορές ο ελεγκτής και η όψη συνδυάζονται, ειδικά στα μικρά προγράμματα.
Το πρότυπο MVC έχει αρκετά πλεονεκτήματα [83] [85]. Το πρότυπο MVC επιτρέπει σε κάθε μέρος να αλλαχτεί ανεξάρτητα από τα άλλα. Επίσης υποστηρίζει πολλές όψεις για το ίδιο μοντέλο και συγχρονισμό όψεων. Αυτό είναι σημαντικό, δεδομένου του πλήθους των συσκευών που να είναι διαθέσιμες για την πρόσβαση σε περιεχόμενο που παρέχεται μέσω του Παγκόσμιου Ιστού. Επίσης πολλοί μηχανισμοί ελέγχου μπορούν να εφαρμοστούν και η παρουσίαση και ο έλεγχος του προτύπου μπορούν να αλλάξουν χωρίς να χρειάζεται να ξαναγραφτεί το ίδιο το πρότυπο. Άρα υπάρχει και εύκολη συντήρηση.
Το βασικότερο μειονέκτημα είναι η πολυπλοκότητα του, η οποία απαιτεί πλήθος ενημερώσεων και μηνυμάτων ακόμη και για απλά αντικείμενα.

Αναφορές:

[1]: Avgeriou, Paris; Uwe Zdun (2005). *Architectural patterns revisited:a pattern language*. 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, July.
[2]: Buschmann F., Meunier R., Rohnert H. & Sommerlad P. & Stal M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons.
[3]: Ralph E. Johnson, (1998), *Architectural Patterns Pipes and filters, Observer network, Client-server, Layers, Active Object-Model*, Object-oriented Programming and Design - Copyright 1998.
[4]: D. D. Corkill, "*Collaborating Software: Blackboard and Multi-Agent Systems & the Future.,"* Proceeding of International Lisp Conference, New York, October, 2003.
[5]: Saadi, *Developing Mobile GUIs*, Mobile Computing Principles, Chapter 6 Cambridge University Press, 2005 page 316-335.
[6]: Support by Dr. Charun Sanrach *Advanced Topics in Human Computer Interaction and Multimedia Interfaces, 8 Implementation* , KMiTNB 274430.
[7]: David Aspinall, (2007), Slides: *Interface Implementation* HCI Lecture 11, Informatics University of Edinburgh 26th October 2007.
[8]: Joëlle Coutaz. *PAC, on object oriented model for dialog design.*,In *Interact'87*, 1987.
[9]: Slides: *Lecture 9 Object- based implementation Decisions* February 23, 2004, Implementation of Telematics Systems.
[10]: Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M., *Pattern-Oriented Software Architecture: A System of Patterns, Vol. 1*. Wiley, 1996.
[11]: Soren Lauesen and Morten Borup Harning.(2001), *Virtual windows: Linking user tasks, datamodels, and interface design.*IEEE Software, pp. 67–75, July/August 2001.
[12]:http://en.wikipedia.org/wiki/Pipeline_(software)
[13]: Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill, (2004), *Patterns for Parallel Programming*, published Sept 15, 2004 by Addison- Wesley Professional
[14]:
http://www.cise.ufl.edu/research/ParallelPatterns/PatternLanguage/AlgorithmStructure/Pipeline.htm
[15]: Bach, M.J.(1986), *The Design of the UNIX Operating System*, Prentice Hall, 1986.

[16]: Hartmann, J., C. Reichetzeder, and M. Varian, *CMS Pipelines*, http://www.akhwien.ac.at/pipeline.html
[17]: Sethna, J., *LASSPTools: Graphical and Numerical Extensions to Unix*, http://www.lassp.cornell.edu/LASSPTools/LASSPTools.html
[18]: Gregor Hohpe, Bobby Woolf. (2004), *Enterprise Integration Patterns – Messaging Systems (chapter 3)*, Addison Wesley Signature Series.
[19]: Master's Thesis Avinash Kewalramani, (2005), Slides: *BioFilter An architecture for parallel deployment and dynamic chaining of standalone BioInformatics tools*. IPDPS 2005.
[20]: Joel Jeffery, *A Real Framework for a Service Oriented World Methods – Solution Architecture*, Capgemini Microsoft Architect Insight Conference, Drive the Debate.
[21]: André Langhorst, Martin Steinle, Hasso-Plattner, *Pipes and Filters Architectural Pattern* Institute for Software Systems Engineering.
[22]: Roger Pressman, (2005), *Software Engineering: A Practitioner's Approach*, 6th edition, McGraw Hill, 2005.
[23]: Dr. James A. Bednar and Dr. David Robertson. (2007), *Architectural Patterns* SAPM Spring 2007: Architecture 1.
[24]: Mary Shaw, *Some Patterns for Software Architectures*, Pattern Languages of Program Design, Vol 2, pp. 255-269, Addison-Wesley, 1996.
[25]: Norman Delisle and David Garlan. *Applying Formal Specification to Industrial Problems: A Specification of an Oscilloscope*. IEEE Software, September 1990.
[26]: M. J. Bach.1986, *The Design of the UNIX Operating System*. Software Series, Prentice-Hall 1986, sec 5.12, pp. 111-119.
[27]: M. R. Barbacci, C. B. Weinstock, and J. M. Wing.(1988), *Programming at the Processor-Memory-Switch Level*. Proc 10th Int'l Conf. on Software Engineering, April 1988.
[28]: G. Kahn. (1974), *The semantics of a simple language for parallel programming.*Information Processing, 1974.North Holland Publishing Company.
[29]: V. Seshadri et al.(1988), *Semantic Analysis in a Concurrent Compiler*. Proceedings of ACM SIGPLAN '88 Conference on Programming Language Design and Implementation.
[30]: Rahul Premraj. (2007), Slides: *Software Architecture Software Engineering* WS 07/08.
[31]: Dominik Grolimund ,Supervisor: Prof. Peter Muller. *Semester Project Design Patterns in Peer-to-Peer Systems*, Department of Computer Science, ETH Zurich
[32]: Ekaterina Chtcherbina, and Markus Völter, (2002), *P2P Patterns Results*, EuroPLoP 2002 Focus Group, Version 0.4, December 14, 2002.
[33]: Gnutella homepage, http://www.gnutella.com
[34]: S. Saroiu, P. K. Gummadi, and S. D. Gribble, (2002), A measurement study of peer-topeer file sharing systems. In MMCN, 2002.
[35]: Ranjita Bhagwan, David Moore, Stefan Savage and Geoffrey M. Voelker, *Replication Strategies for Highly Available Peer-to-Peer Storage*, In Proceedings of the International Workshop on Future Directions in Distributed Computing, Bertinoro, Italy, June 2002.
[36]: E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
[37]: http://en.wikipedia.org/wiki/Implicit_invocation
[38]: Robert M. Balzer, (1986), *Living with the Next Generation Operating System*. Proc 4th World Computer Conf., September 1986.
[39]: David Garlan, Gail E. Kaiser, and David Notkin. (1992), *Using Tool Abstraction to Compose Systems*. IEEE Computer, vol 25, June 1992.
[40]: C. Gerety. HP Softbench. (1989), *A New Generation of Software Development Tools*. TR SESD-89-25, Hewlett-Packard Software Engineering System Division, Ft Collins CO, November 1989.
[41]: Nico Habermann and David Notkin. Gandalf, (1986), *Software Development Environments.* IEEE Tr on Software Engineering, vol SE-12, December 1986.
[42]: Carl Hewitt. Planner, (1969). *A Language for Proving Theorems in Robots*. Proc First Int'l Joint Conf. in Artificial Intelligence, 1969.
[43]: G. Krasner and S. Pope. (1988), *A Cookbook for Using the Model-View-Controller User Interface Paradigm* in Smalltalk-80. Journal of Object Oriented Programming, vol 1, August/September 1988.
[44]: S. P. Reiss.(1990), *Connecting Tools Using Message Passing in the Field Program Development Environment*. IEEE Software, July 1990.
[45]: LIANG Xianzhong, WANG Zhenyu , Wuhan. (2002), *Event-based Implicit Invocation Decentralized in Ada** , ACM SIGAda Ada Letters, Volume XXII , Issue 1 (March 2002) REVIEWS 2002 pp.11 – 16.

[46]: ROLANDO BLANCO, PAULO ALENCAR David R. Cheriton. (2007), *Categorization of Implicit Invocation Systems*, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1, Technical Report CS-2007-31.
[47]: K. Sullivan and D. Notkin, (1990). *Reconciling environment integration and component independence*, In Proceedings of ACM SIGSOFT90: Fourth Symposium on Software Development Environments, pp. 22-23, December 1990.
[48]: V.Ambriola and G.Tortora, (1993), *An Introduction to Software Architecture*," *Advances in Software Engineering and Knowledge Engineering, Volume I,* , World Scientific Publishing Company, New Jersey, 1993.
[49]: Richard Pawson, (2004), *Naked objects*, A thesis submitted to the University of Dublin, Trinity College for the degree of Doctor of Philosophy, Department of Computer Science, Trinity College, Dublin, June 2004.
[50]: http://wapedia.mobi/en/Naked_objects
[51]: http://www.hibernate.org/
[52]: http://www.nakedobjects.net/tutorial/index.shtml
[53]: Sergio Caltagirone, Matthew Keys, Bryan Schlief, Mary Jane Willshire, (2002) *An Architecture for a Massively Multiplayer Online Role Playing Game Engine*, Journal of Computing Sciences in Colleges, Volume 18 , Issue 2 (December 2002)
pp.105 – 116.
[54]: http://www.vico.org/pages/PatronsDisseny/Pattern%20Blackboard/
[55]: Engelmore, R. and T. Morgan (Eds), (1988), *Blackboard Systems*, Addison-Wesley, 1988
[56]: Nii, H.P.(1986), *Blackboard Systems, Part I and II*, The AI Magazine, vol. 7, nos 2 (pp. 38-53) and 3(pp. 82-106), 1986.
[57]: Terry, A. (1985), *Using Explicit Strategic Knowledge to Control Expert Systems*, originally published in 1985, reproduced in Blackboard Systems, pp. 159-188.
[58]: http://chat.carleton.ca/~narthorn/project/patterns/BlackboardPattern-display.html
[59]: Andy Bulka, (2004), *Blackboard Architectural Pattern,*, Melbourne Patterns Group – 2004.
[60]: Philippe Lalanda, Thomson CSF, (1998), *Two complementary patterns to build multi-expert systems*, Corporate Research Laboratory Domaine de Corbeville, OOSA '98.
[61]: http://chat.carleton.ca/~narthorn/project/patterns/BlackboardPattern-display.html
[62]: F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, vol. 1. West Sussex, England: Wiley, 1996.
[63]: James A. Bednar and Dr. David Robertson, (2006), *Architectural Patterns* SAPM Spring 2006: Architecture.
[64]: Jing Dong and Shanguo Chen, Jun Jang Jeng, *Event-Based Blackboard Architecture for Multi-Agent Systems* , Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference Volume 2, Issue , 4-6 April 2005 pp.379 - 384 Vol. 2.
[65]: Gartner, SSA Research Note SPA-401-068, 12 April 1996, *Service Oriented' Architectures*, Part 1
[66]: Gartner, SSA Research Note SPA-401-068, 12 April 1996, *Service Oriented' Architectures*, Part 2
[67]: Yefim V. Natis, (2003), *Service-Oriented Architecture Scenario*, Gartner Research 16 April 2003
[68]: Duane Nickull, Laurel Reitman, James Ward, Jack Wilber and Mellon, (2007), *Service Oriented Architecture (SOA) and Specialized Messaging Patterns*, Technical White Paper.
[69]: Carlos Pedrinaci, Matthew Moran, and Barry Norton , *Towards a Semantic Event-Based Service-Oriented Architecture*. 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006) at The 5th International Semantic Web (ISWC 2006).
[70]: Jay DiMare,(2005), *Service-oriented architecture A practical guide to measuring return on that investment*, IBM Global Business Services, Application Innovation Services.
[71]: Mark Brooks, *Service Oriented Architecture & Grid Computing*, The MITRE Corporation.
[72]: http://en.wikipedia.org/wiki/Model-view-controller
[73]: Trygve M. H. Reenskaug/MVC—XEROX PARC 1978-79.
[74]: Steve Burbeck,(1987), *How to use Model–View–Controller (MVC)*, Applications Programming in Smalltalk-80(TM).
[75]: Dubinko, Micah (2004-11-12). "XForms 1.0 Basic Profile". http://www.w3.org/TR/2002/CR-xforms-20021112/. Retrieved on 2008-10-16.
[76]: Glenn E. Krasner and Stephen T. Pope, *A Description of the Model-View-Controller User Interface Paradigm* ,Smalltalk-80 System ParcPlace Systems, Inc.
[77]: *Geospatial Service-Oriented Architecture (SOA)*, An ESRI White Paper, June 2007.
[78]: Lawrence A. Rowe, (2001), *Model-View-Controller*, Computer Science Division , Smalltalk PARC.

[79]: James Landay, (2002), Slides: *Model View Controller*, CS 160, Spring 2002, April 29, 2002 User Interface Design, Prototyping, & Evaluation.
[80]: Robert F. Zant, (2006), Slides: *Model-View-Controller Architecture in a Systems Analysis and Design*, Illinois State University
[81]: http://cristobal.baray.com/indiana/projects/mvc.html
[82]: Anshuman Tiwari. Kaushal Mittal,(2005), *Study of J2EE Patterns* , J2EE PATTERNS.
[83]: Δημοσθένης Ακουμιανάκης,(2006), *ΔΙΕΠΑΦΗ ΧΡΗΣΤΗ ΥΠΟΛΟΓΙΣΤΗ μια σύγχρονη προσέγγιση*, ΔΙΕΠΑΦΗ ΑΝΘΡΩΠΟΥ ΥΠΟΛΟΓΙΣΤΗ ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ 2006.
[84]: Martin Hunter, (2006). *Tidying the House: The MVPC Software Design Pattern,* Extending the MVP pattern to abstract the objectives of a presentation from the detail of implementation using modern presentation frameworks, July 2006.
[85] : Chris North, (2006), *User Interface Programming in C#: Model-View-Controller*
[86]: Helmut Windl, (2002), *Siemens AG Presentation Abstraction Control* ,First International Conference on Usage-Centered Design Portsmouth, New Hampshire, 25-28 August 2002.
[87]: Beverly A. Sanders,Berna Massingill, Timothy G Mattson , (2005), The Algorithm Structure Design Space in Parallel Programming, Mar 11, 2005,Sample Chapter is provided courtesy of Addison-Wesley Professional.
[88] : http://en.wikipedia.org/wiki/Peer-to-peer#Advantages_of_P2P_networks
[89] : M. Amoretti, M. Reggiani, F. Zanichelli, G. Conte ,*Peer: an Architectural Pattern* ,Distributed Systems Group Plop2005.
[90] : Groove Networks, http://www.oreillynet.com/pub/d/312
[91] : Benjamin Edwards, *An Introduction to Implicit Invocation Architectures* ,
http://greatbiztoolsllc-trac.cvsdude.com/mach-ii/wiki/IntroductionToImplicitInvocationArchitectures
[92]: Mary Shaw, (1994), *Pattern Languages of Program Design, Vol 2 Some Patterns for SoftwareArchitectures*, First Annual Conference on the Pattern Languages of Programming,Plop94.
[93] : http://en.wikipedia.org/wiki/Naked_objects
[94] : Richard Pawson and Vincent Wade, (2003), *Agile Development Using Naked Objects* Extreme Programming and Agile Processes in Software Engineering, Springer Berlin / Heidelberg, Volume 2675/2003.
[95] : http://www.nakedobjects.org/tutorial/index.shtml
[96] : *Design Patterns for Events and Wrappers Lecture 15*, 2002,
http://www.castle-cadenza.demon.co.uk/wrapper.htm
[97] : Adrian Giurca (2003). *Towards a Java Framework for Knowledge Representation and Inference*, In Proc. of 3th International Conference on Artificial Intelligence and Digital Communication, AIDC'2003, June 2003, pp.35-48, ISBN: 973-8419-71-9.
[98] : Marc Brooks, *Service Oriented Architecture & Grid Computing*, The MITRE Corporation.

9. Εξειδικευμένα πρότυπα για περιοχές ανάπτυξης λογισμικού

9.1 Εισαγωγή στα Εξειδικευμένα πρότυπα

Σε ορισμένες περιπτώσεις τα πρότυπα λογισμικού δεν καλύπτουν τις ανάγκες των σχεδιαστών. Οι ανάγκες των σχεδιαστών μπορεί να είναι αυξημένες επειδή οι περιοχές ανάπτυξης λογισμικού που θέλουν να δημιουργήσουν είναι εξειδικευμένες και αναφέρονται σε συγκεκριμένους τομείς. Σε αυτές τις περιπτώσεις ή πρέπει τα υπάρχοντα πρότυπα να συνδυαστούν μεταξύ τους ή να δημιουργηθούν νέα πρότυπα, τα οποία θα καλύπτουν τις ανάγκες αυτές. Ανάλογα με τις απαιτήσεις που έχει κάθε

σχεδιαστής για το λογισμικό, συνδυάζει και τα ανάλογα πρότυπα. Σε κάθε περίπτωση τα νέα πρότυπα που δημιουργούνται είναι εξειδικευμένα.
Συνεχώς δημιουργούνται καινούρια Εξειδικευμένα Πρότυπα, που καλύπτουν ολοένα και περισσότερες και διαφορετικές ανάγκες. Θα αναφερθούν ορισμένα από τα Εξειδικευμένα Πρότυπα που υπάρχουν. Αυτά είναι το Κρυπτογραφικό Μεταπρότυπο (Cyptographic Metapattern), τα Πρότυπα Χρόνου (Time Patterns), τα Πρότυπα για Ασφαλιστικά Συστήματα (Patterns for Insurance Systems) και τα Πρότυπα Αντικειμένων Μεταφορών (Transport objects patterns TOPs).

9.2 Κρυπτογραφικό Μεταπρότυπο (Cyptographic Metapattern)

Στις μέρες μας οι κρυπτογραφικές τεχνικές χρησιμοποιούνται ευρέως σε πολλές εφαρμογές. Αυτή η μεγάλη διάδοση των τεχνικών, το ενδιαφέρον και η έρευνα σε αρχιτεκτονικές λογισμικού και πρότυπα, οδήγησε σε αρχιτεκτονικές κρυπτογραφικού λογισμικού και κρυπτογραφικά πρότυπα.

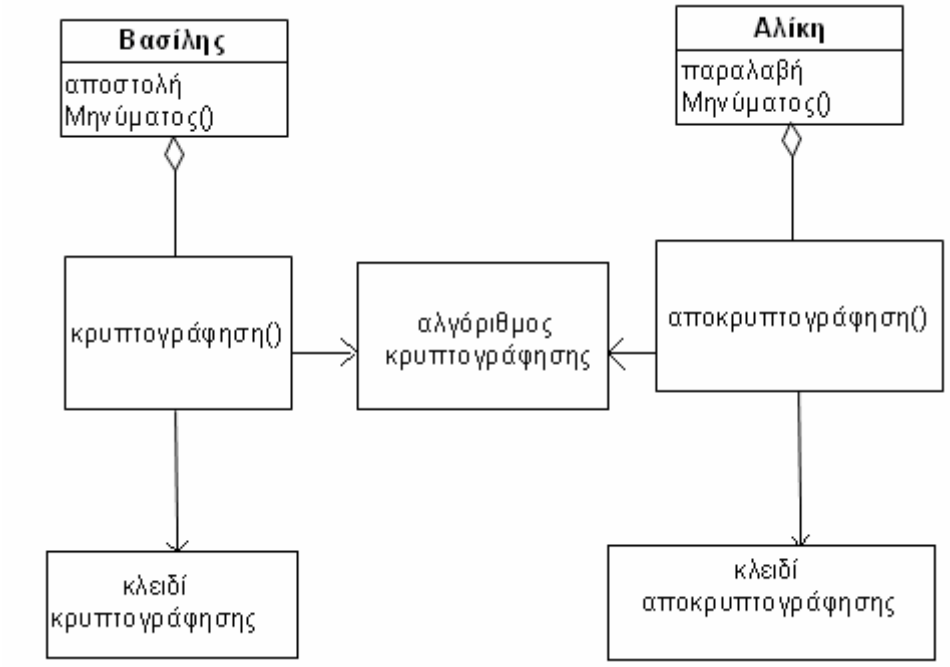
Τα κρυπτογραφικά συστήματα χρησιμοποιούνται για να παρέχουν μυστικότητα (secrecy), ακεραιότητα δεδομένων (data integrity), πιστοποίηση χρηστών (user authentication), αδυναμία απάρνησης (non-repudiation).
Κρυπτογράφηση είναι η διεργασία μετασχηματισμού ενός μηνύματος μεταξύ ενός αποστολέα και ενός παραλήπτη σε μια ακατανόητη μορφή, ώστε αυτό να μην είναι αναγνώσιμο από τρίτους [4] [9] [11].
Αποκρυπτογράφηση είναι η διεργασία ανάκτησης ενός αρχικού μηνύματος από την κρυπτογραφημένη μορφή του [2].
Σύστημα κρυπτογράφησης είναι ο αλγόριθμος μετασχηματισμού δεδομένων από μια αρχική μορφή (plain text) σε μία νέα (cipher text), από την οποία δεν προκύπτει νόημα.
Κλειδί (key) είναι ένας αριθμός που χρησιμοποιείται μαζί με τον αλγόριθμο της κρυπτογράφησης [8]. Το μέγεθος των κλειδιών μετριέται σε bits. Κλειδιά μεγάλου μεγέθους παρέχουν ανθεκτικότερη κρυπτογράφηση. Η ανθεκτικότητα μιας κρυπτογράφησης εξαρτάται περισσότερο από το μέγεθος του κλειδιού, παρά από τον αλγόριθμο κρυπτογράφησης. Διαφορετικοί αλγόριθμοι απαιτούν διαφορετικά μήκη κλειδιών για να πετύχουν το ίδιο επίπεδο ανθεκτικότητας κρυπτογράφησης [1] [4].
Αυθεντικοποίηση είναι η εγκυρότητα ενός χρήστη, ενός ηλεκτρονικού υπολογιστή, ή ενός ψηφιακού αντικειμένου. Δηλαδή η διαβεβαίωση ότι είναι αυτοί που ισχυρίζονται ότι είναι [10]. Η ακεραιότητα των δεδομένων εξασφαλίζει ότι τα δεδομένα δεν θα τροποποιηθούν [13] [30].

Θα περιγραφούν εννιά κρυπτογραφικά πρότυπα. Αυτά είναι Μυστικότητα Πληροφοριών (Information Secrecy) , Αυθεντικότητα Μηνυμάτων (Message Authentication), Ακεραιότητα Μηνυμάτων (Message Integrity), Αυθεντικότητα Αποστολέα (Sender Authentication), Μυστικότητα με Αυθεντικότητα (Secrecy with Authentication), Μυστικότητα με Υπογραφή (Secrecy with Signature), Μυστικότητα με Ακεραιότητα (Secrecy with Integrity), Υπογραφή με Παράρτημα (Signature with Appentix) και Μυστικότητα με Υπογραφή με Παράρτημα (Secrecy with Signature with Appentix).

Το Κρυπτογραφικό Μεταπρότυπο παράγει την μικροαρχιτεκτονική για όλα τα τέσσερα βασικά πρότυπα. Αυτά είναι Αυθεντικότητα Μηνυμάτων (Message Authentication), Μυστικότητα Πληροφοριών (Information Secrecy), Αυθεντικότητα Αποστολέα (Sender Authentication), Ακεραιότητα Μηνυμάτων (Message Integrity). Όλα τα άλλα πρότυπα δημιουργούνται από συνδυασμό αυτών.

9.2.1 Μυστικότητα Πληροφοριών (Information Secrecy)

Έστω ότι η Αλίκη θέλει να στείλει μηνύματα στον Βασίλη. Θέλει να κρατήσει τα μηνύματα αυτά κρυφά από την Μαρία, την οποία υποψιάζεται η Αλίκη ότι προσπαθεί να διαβάσει τα μηνύματά της. Πώς μπορεί να στείλει η Αλίκη ένα μήνυμα στον Βασίλη με τέτοιο τρόπο, ώστε η Μαρία να μην μπορεί να τα διαβάσει; [12]
Η κρυπτογράφηση είναι αργή. Αν είναι σημαντική η απόδοση, η χρήση της πρέπει να αξιολογηθεί προσεκτικά. Η ασφάλεια μιας κρυπτογραφημένης πληροφορίας είναι το κλειδί της κρυπτογράφησης. Επομένως, το κόστος για το σπάσιμο- ανακάλυψη του κλειδιού πρέπει να είναι μικρότερο από ότι η αξία του μηνύματος. Η δύναμη του συστήματος κρυπτογράφησης βασίζεται στη μυστικότητα ενός μεγάλου κλειδιού. Όσο μεγαλύτερο είναι το κλειδί , τόσο πιο αργός είναι και ο αλγόριθμος.
Το πρότυπο αυτό υποστηρίζει κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων. Η Αλίκη και ο Βασίλης συμφώνησαν από πριν σε μια κρυπτογραφική μηχανή και σε ένα κοινό κλειδί. Ο Βασίλης κρυπτογραφεί το μήνυμα και το στέλνει στην Αλίκη. Η Αλίκη αποκρυπτογραφεί το κρυπτογραφημένο μήνυμα και αποκαλύπτει το αρχικό γνήσιο μήνυμα. Η δομή του προτύπου φαίνονται στο σχήμα 9.2.1.1 [29].

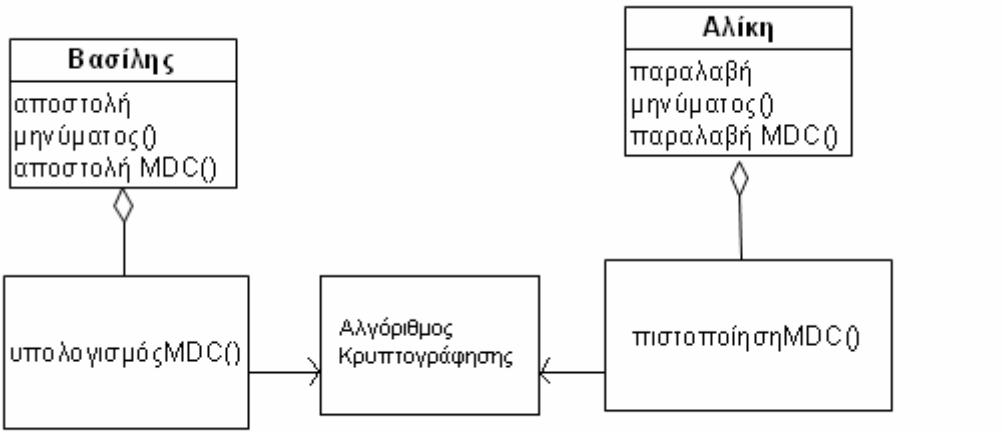


Σχήμα 9.2.1.1 Απεικονίζεται η δομή της Μυστικότητας Πληροφοριών.

9.2.2 Ακεραιότητα Μηνυμάτων (Message Integrity).

Η Αλίκη στέλνει μεγάλα μηνύματα στον Βασίλη. Αυτός θέλει να πιστοποιήσει την ακεραιότητα των ληφθέντων μηνυμάτων, τα οποία η Μαρία μπορεί να έχει αλλάξει.

Η Αλίκη και ο Βασίλης δεν μοιράζονται κλειδιά κρυπτογράφησης, έτσι δεν μπορούν να υπογράψουν και να πιστοποιήσουν μηνύματα με τους κώδικες αυθεντικοποίησης μηνυμάτων (Message Authentication Codes). Πώς μπορεί ο Βασίλης να αποφασίσει, αν ένα μήνυμα τροποποιήθηκε, αφού στάλθηκε;
Η ακεραιότητα των μηνυμάτων μπορεί να υπάρξει, στέλνοντας κάθε μήνυμα δυο φορές και συγκρίνοντας τα αντίγραφα. Στέλνοντας μικρά μηνύματα δυο φορές μπορεί να είναι πιο ακριβό, από ότι υπολογίζοντας και στέλνοντας ένα σχετικά μεγάλο κώδικα ανίχνευσης τροποποίησης (Modification Detection Code - MDC). Είναι απαραίτητο να πιστοποιηθεί ένας μικρός κώδικας ανίχνευσης τροποποίησης για να καθοριστεί πότε ένα μεγάλο ποσό δεδομένων έχει τροποποιηθεί ή όχι. Οι κωδικοί ανίχνευσης τροποποίησης μόνοι τους δεν εγγυώνται ότι ένα μήνυμα το έχει γράψει ο αναμενόμενος συντάκτης. Οι κώδικες ανίχνευσης τροποποίησης (Modification Detection Code - MDC θα μπορούσαν να χρησιμοποιηθούν ως μοναδικά προσδιοριστικά των ηλεκτρονικών νομισμάτων στις εφαρμογές ηλεκτρονικού εμπορίου [3].
Η Αλίκη και ο Βασίλης συμφωνούν να χρησιμοποιήσουν ένα κωδικό ανίχνευσης τροποποίησης. Ο Βασίλης υπολογίζει τον κωδικό ανίχνευσης τροποποίησης του μηνύματος και τον συγκρίνει με αυτό που έλαβε από την Αλίκη. Αν είναι ίδιοι, τότε το μήνυμα δεν αλλάχτηκε [29].

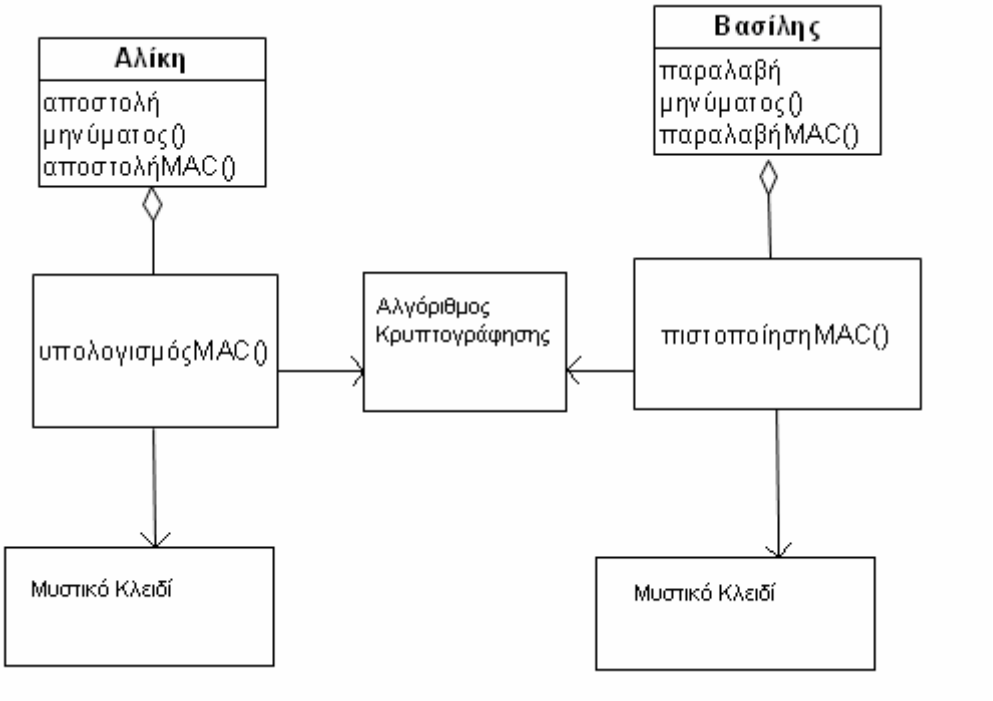


Σχήμα 9.2.2.1 Απεικονίζεται το πρότυπο ακεραιότητας μηνυμάτων.

9.2.3 Αυθεντικότητα Μηνυμάτων (Message Authentication)

Η Αλίκη και ο Βασίλης θέλουν να ανταλλάξουν μηνύματα, αλλά δεν μπορούν να διακρίνουν τα δικά τους μηνύματα από της Μαρίας που μπορεί να έχουν μπει στο κανάλι επικοινωνίας. Έχουν την ικανότητα να μοιράζονται μυστικά με ένα ασφαλή τρόπο. Πώς μπορούν τα γνήσια μηνύματα να διακριθούν από τα πλαστά;
Ένα μήνυμα μπορεί να αυθεντικοποιηθεί, εάν η πληροφορία αναγνωρίζεται μόνο από τα μέρη που επικοινωνούν με αυτό. Αν η πληροφορία αυθεντικότητας είναι μια συνάρτηση δεδομένων και ενός μυστικού, τότε η αλλαγή των μηνυμάτων ή τμημάτων από αυτά, μπορεί να εντοπιστεί. Δεν μπορεί να είναι εγγυημένη η αυθεντικότητα, έως ότου και οι δυο πλευρές να μπορούν να υπολογίσουν σωστά κώδικες ανίχνευσης τροποποίησης (MAC). Η αξία των δεδομένων πρέπει να είναι μεγαλύτερη από το κόστος εγγύησης της αυθεντικότητας [29].

Η Αλίκη και ο Βασίλης συμφώνησαν από πριν να μοιραστούν ένα μυστικό κλειδί και ένα αλγόριθμο κρυπτογράφησης για να παράγουν MACs (Message Authentication Codes). Η Αλίκη υπολογίζει το MAC ενός μηνύματος και στέλνει και το μήνυμα και το MAC στον Βασίλη. Ο Βασίλης υπολογίζει το MAC ξανά και το συγκρίνει με αυτό που έλαβε από τη Αλίκη. Αν ταιριάζουν, το μήνυμα είναι γνήσιο και το έχει στείλει η Αλίκη, επειδή εκτός του Βασίλη, μόνο η Αλίκη γνωρίζει το μυστικό κλειδί και υπολογίζει το σωστό MAC. Ένα όφελος χρησιμοποιώντας αυτό το πρότυπο είναι η υπονοούμενη ακεραιότητα των μηνυμάτων. Οι MACs μπορούν να χρησιμοποιηθούν στην αυθεντικότητα των IP των πακέτων στο Διαδίκτυο [5] [29].

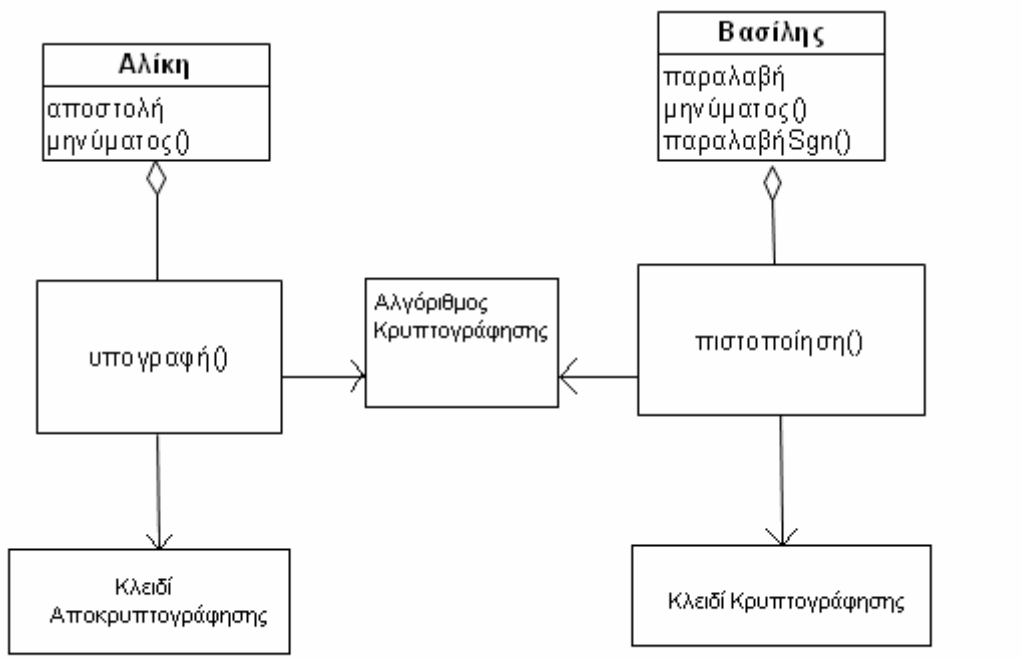


Σχήμα 9.2.3.1 Απεικονίζεται το πρότυπο Αυθεντικότητας Μηνυμάτων.

9.2.4 Αυθεντικότητα Αποστολέα (Sender Authentication)

Η Αλίκη στέλνει μηνύματα στον Βασίλη, αλλά δεν μπορούν να ξεχωρίσουν τα μηνύματα από αυτά που η Μαρία μπορεί να εισάγει στο κανάλι επικοινωνίας. Η Αλίκη μπορεί αργότερα να αμφισβητήσει ότι τα μηνύματα έχουν σταλεί από αυτήν την ίδια. Σε μια τέτοια κατάσταση ο Βασίλης δεν μπορεί να αποδείξει σε ένα τρίτο σύνολο (third party) ότι μόνο η Αλίκη θα μπορούσε να έχει στείλει τα μηνύματα. Η Αλίκη έχει ένα ζευγάρι δημόσιο-ιδιωτικό κλειδί ή την ικανότητα να δημιουργήσει ζευγάρια κλειδιών. Το δημόσιο κλειδί της είναι γνωστό σε όλους [29].
Πώς γίνεται να εγγυηθεί ότι τα μηνύματα έχουν ένα γνήσιο και αυθεντικό αποστολέα, με τέτοιο τρόπο που η Αλίκη δεν μπορεί να αρνηθεί ότι έχει στείλει ένα μήνυμα στον Βασίλη;
Ψεύτικα ψηφιακά μηνύματα μπορούν να δημιουργηθούν ή να αντιγραφούν εύκολα. Πρέπει να παρθούν μέτρα για να συσχετιστεί ένα μήνυμα με τον νόμιμο αποστολέα του. Ψηφιακές υπογραφές προσφέρουν τον τρόπο ώστε ο νόμιμος αποστολέας ενός υπογεγραμμένου μηνύματος δεν μπορεί να αρνηθεί ότι το έγραψε αυτός. Οι υπογραφές είναι συνήθως τόσο μεγάλες όσο και τα δεδομένα που υπογράφονται. Το

κόστος παροχής της αυθεντικότητας ενός μηνύματος δεν μπορεί να είναι μεγαλύτερο από την πραγματική αξία του.
Η Αλίκη και ο Βασίλης συμφωνούν στη χρήση ενός δημόσιου κλειδιού πρωτοκόλλου ψηφιακής υπογραφής και ο Βασίλης έχει το δημόσιο κλειδί της Αλίκης. Η Αλίκη κρυπτογραφεί ένα μήνυμα με το ιδιωτικό της κλειδί για να το υπογράψει και το στέλνει στον Βασίλη. Αυτός αποκρυπτογραφεί το μήνυμα με το δημόσιο κλειδί της Αλίκης ώστε να το πιστοποιήσει. Αν το νέο μήνυμα έχει νόημα τότε ο Bob ξέρει ότι η Αλίκη είναι ο αποστολέας του μηνύματος [29].



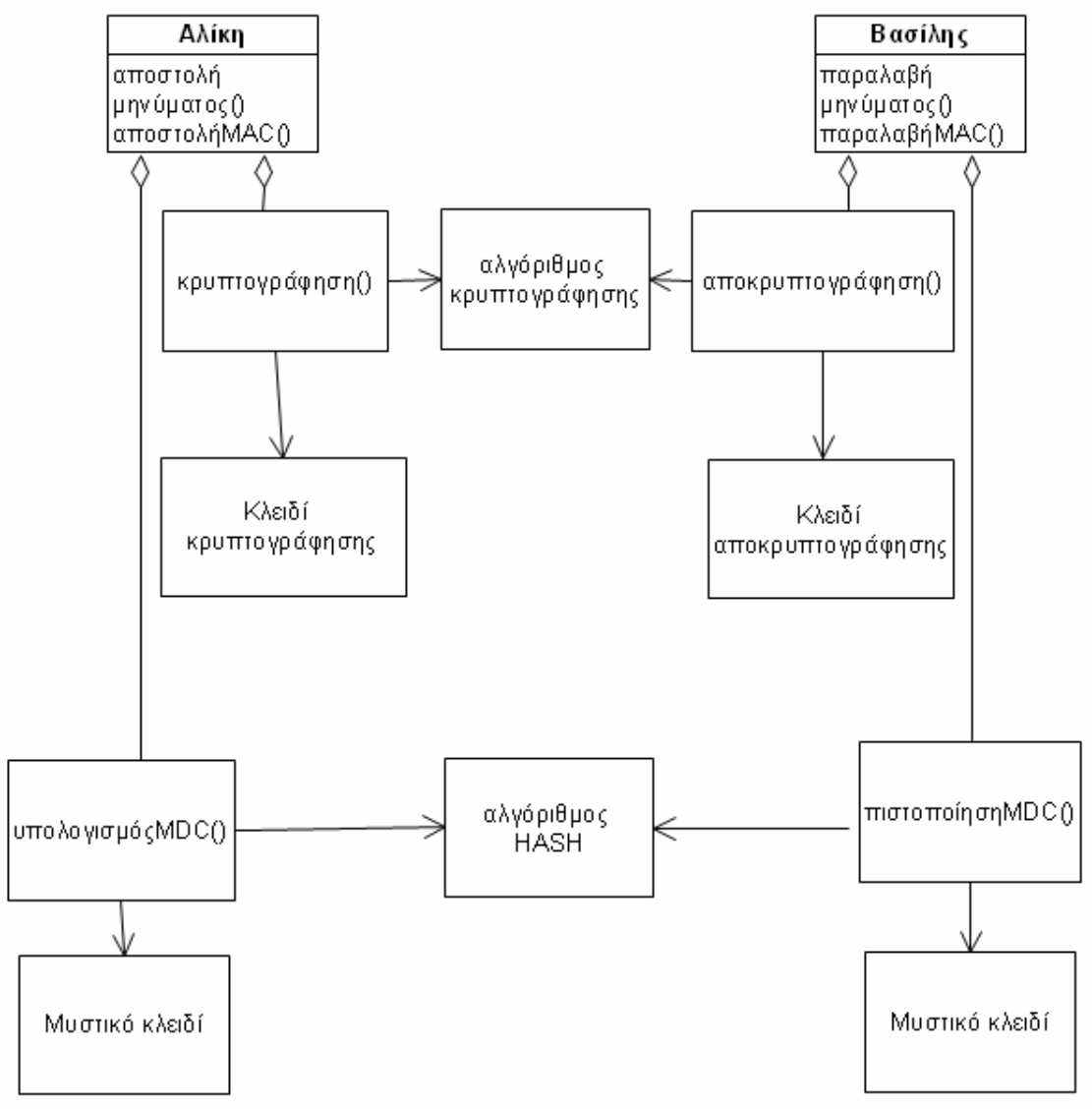
Σχήμα 9.2.4.1 Απεικονίζεται το πρότυπο Αυθεντικότητας Αποστολέα.

Οι ψηφιακές υπογραφές χρησιμοποιούνται στις εφαρμογές ηλεκτρονικού εμπορίου στην επικύρωση των πελατών [3] [6]. Θα μπορούσαν, επίσης να χρησιμοποιηθούν για να εγγυηθούν την αυθεντικότητα των πληροφοριών που λήφθηκε μέσω του Διαδικτύου [7].

9.2.5 Μυστικότητα με Αυθεντικότητα (Secrecy with Authentication)

Η Αλίκη και ο Βασίλης χρησιμοποιούν κρυπτογράφηση δημόσιου κλειδιού για να ανταλλάξουν κρυπτογραφημένα μηνύματα. Η Μαρία μπορεί να σταματήσει, αλλά δεν μπορεί να διαβάσει τα μηνύματα. Μπορεί όμως να αλλάξει τα μηνύματα με τέτοιο τρόπο ώστε ο Βασίλης και η Αλίκη να μην εντοπίσουν αλλαγές [29].
Πώς μπορεί η Αλίκη να αυθεντικοποιήσει ένα κρυπτογραφημένο μήνυμα , χωρίς να χάσει μυστικότητα;
Στην κρυπτογράφηση δημοσίου κλειδιού, σωστά κρυπτογραφημένα μηνύματα μπορούν να παραχθούν από οποιονδήποτε που έχει πρόσβαση στο δημόσιο κλειδί. Αν η Αλίκη και ο Βασίλης χρησιμοποιούν κρυπτογράφηση μυστικού κλειδιού και είναι οι μόνοι που μοιράζονται το μυστικό, η αυθεντικότητα μηνύματος είναι άχρηστη. Η αυθεντικότητα και κρυπτογράφηση είναι ανεξάρτητες υπηρεσίες . Η αυθεντικότητα μηνυμάτων περιορίζει τον αριθμό των οντοτήτων που μπορούν να παράγουν κρυπτογραφημένα μηνύματα. Επίσης προσθέτει ένα νέο βήμα και στην

κρυπτογράφηση και στην αποκρυπτογράφηση, ώστε να υπολογίσει και να πιστοποιήσει ένα MAC. Μπορεί εύκολα να επιτευχθεί και μυστικότητα και αυθεντικότητα μηνυμάτων.
Δυο προηγούμενα πρότυπα κρυπτογράφησης συνδυάζονται για να λύσουν το πρόβλημα. Αυθεντικότητα Μηνυμάτων (Message Authentication) και Μυστικότητα Πληροφοριών (Information Secrecy). Το MAC πρέπει να υπολογιστεί στο αρχικό μήνυμα. Και το κρυπτογραφημένο μήνυμα και το MAC στέλνονται στον Βασίλη. Το μυστικό κλειδί που χρησιμοποιείται για να υπολογίσει το MAC, πρέπει να είναι διαφορετικό από αυτό που χρησιμοποιήθηκε για κρυπτογράφηση. Η μυστικότητα και η αυθεντικότητα μπορούν να συνδυαστούν για να σιγουρευτούν τα πακέτα IP στο Διαδίκτυο [5] [29].

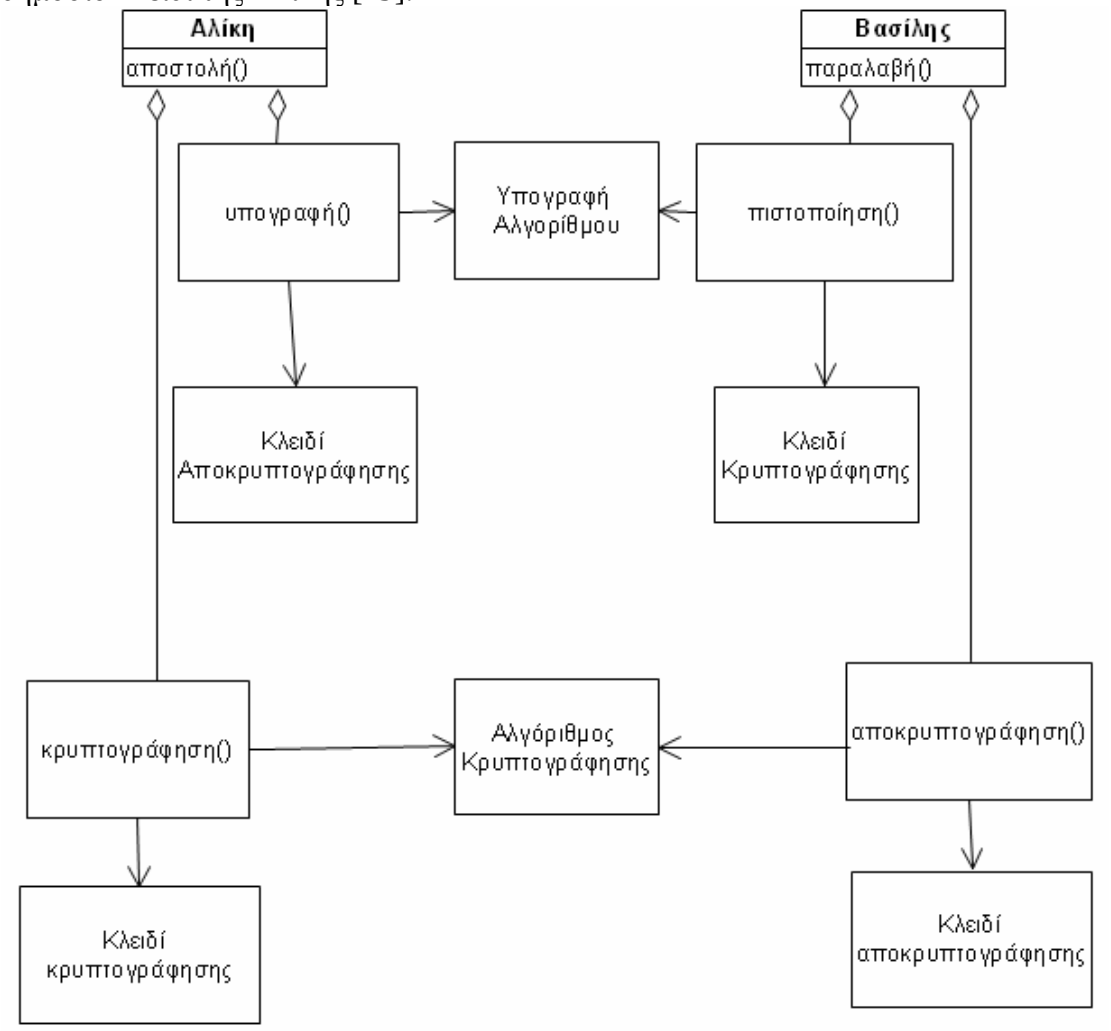


Σχήμα 9.2.5.1 Απεικονίζεται το πρότυπο Μυστικότητα με Αυθεντικότητα..

9.2.6 Μυστικότητα με Υπογραφή (Secrecy with Signature).

Η Αλίκη και ο Βασίλης ανταλλάσουν κρυπτογραφημένα μηνύματα, αλλά δεν μπορούν να εγγυηθούν την αυθεντικότητα σε ένα κρυπτογραφημένο μήνυμα. Επιπλέον η Μαρία μπορεί να τροποποιήσει, να αντικαταστήσει και να εισάγει

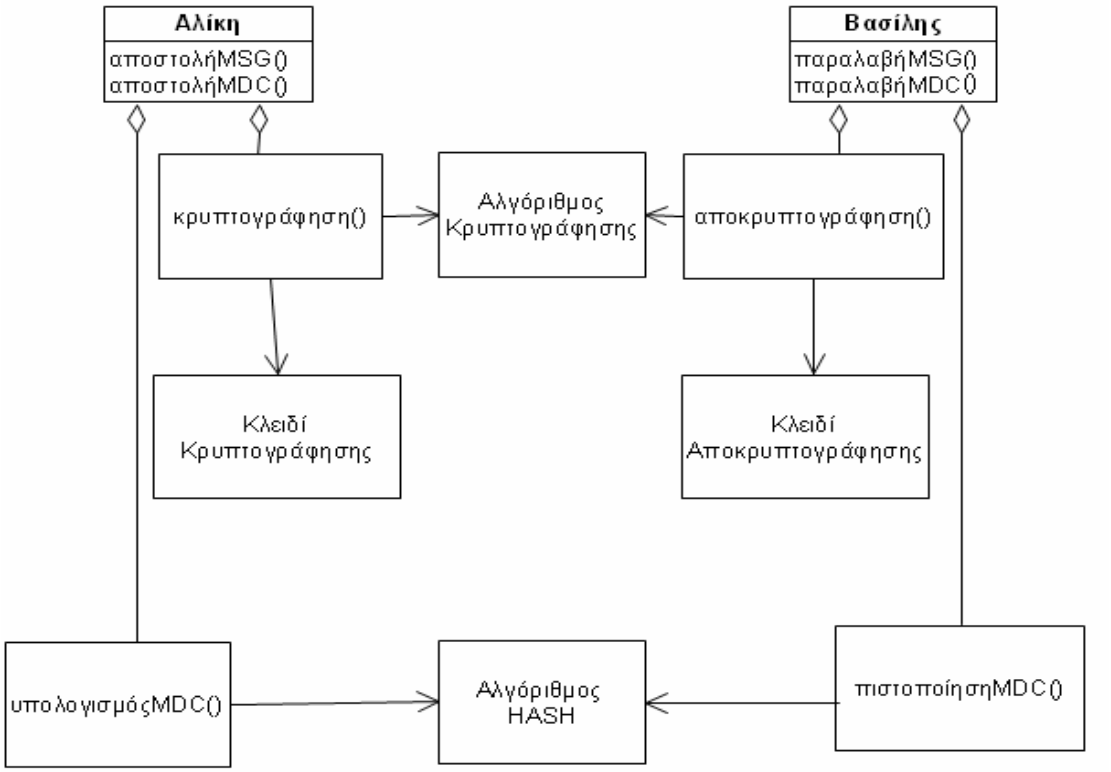
μηνύματα στο κανάλι επικοινωνίας με τέτοιο τρόπο ώστε η Αλίκη και ο Βασίλης να μη μπορούν να εντοπίσουν ψευδή μηνύματα. Η Αλίκη και ο Βασίλης ήδη μοιράζονται κλειδιά για μυστικούς σκοπούς [29].
Πώς μπορεί ο Βασίλης να αποδείξει την αυθεντικότητα ενός κρυπτογραφημένου μηνύματος χωρίς απώλεια της μυστικότητας με τέτοιο τρόπο, ώστε να εγγυηθεί και την ακεραιότητα και την αυθεντικότητα;
Οποιοσδήποτε που μπορεί να παράγει κρυπτογραφημένα μηνύματα μπορεί να αρνηθεί την αυθεντικότητά τους στο μέλλον. Απόδειξη αυθεντικότητας και μυστικότητας είναι ανεξάρτητες υπηρεσίες, η παρουσία της μιας δεν υπονοεί και την παρουσία της άλλης. Η αυθεντικότητα αποστολέα μπορεί να εγγυηθεί αυθεντικότητα κρυπτογραφημένων μηνυμάτων, εισάγοντας ένα βήμα στη διαδικασία κρυπτογράφησης και αποκρυπτογράφησης, προκαλώντας μείωση της απόδοσης. Ο συνδυασμός μυστικότητας και ψηφιακών υπογραφών μπορεί να γίνει εύκολα, χρησιμοποιώντας ένα σχεδιασμό, στον οποίο η διαμόρφωση των συστατικών διευκολύνει την επαναχρησιμοποίηση.
Δυο προηγούμενα πρότυπα κρυπτογράφησης συνδυάζονται για να λύσουν το πρόβλημα. Το Μυστικότητα Πληροφοριών (Information Secrecy) και Αυθεντικότητα Αποστολέα (Sender Authentication). Η Αλίκη υπογράφει ένα μήνυμα με το ιδιωτικό της κλειδί, κρυπτογραφεί το υπογεγραμμένο μήνυμα με το δημόσιο κλειδί του Βασίλη και το στέλνει στον Βασίλη. Ο Βασίλης αποκρυπτογραφεί το κρυπτογραφημένο μήνυμα με το ιδιωτικό του κλειδί και πιστοποιεί το υπογεγραμμένο μήνυμα με το δημόσιο κλειδί της Αλίκης [29].



Σχήμα 9.2.6.1 Απεικονίζεται το πρότυπο Μυστικότητα με Υπογραφή.

9.2.7 Μυστικότητα με Ακεραιότητα (Secrecy with Integrity).

Η Αλίκη και ο Βασίλης ανταλλάσουν κρυπτογραφημένα μηνύματα, αλλά δεν μπορούν να εντοπίσουν τροποποιήσεις ή αλλαγές που πιθανώς να έκανε η Μαρία. Δεν θέλουν να μοιραστούν ένα μυστικό κλειδί μόνο για σκοπό αυθεντικότητας. Πώς διασφαλίζεται η ακεραιότητα ενός κρυπτογραφημένου μηνύματος χωρίς απώλεια της μυστικότητας;
Τα λάθη κατά την μετάδοση και διάφορες αλλαγές μπορούν να προκαλέσουν αλλαγή σε σωστά δεδομένα και αλλαγή στο νόημά τους μετά από την αποκρυπτογράφηση. Διατήρηση ακεραιότητας δεδομένων και μυστικότητα είναι ανεξάρτητες υπηρεσίες και η παρουσία της μιας δεν συνεπάγεται την παρουσία της άλλης. Ο υπολογισμός και η πιστοποίηση ενός MDC (Modification Detection Code) μπορεί να εγγυηθεί την ακεραιότητα των κρυπτογραφημένων μηνυμάτων, αλλά μπορεί να προκαλέσει μείωση της απόδοσης. Ο συνδυασμός μυστικότητας και αυθεντικότητας μπορεί να γίνει εύκολα, χρησιμοποιώντας ένα σχεδιασμό, στον οποίο η διαμόρφωση των συστατικών διευκολύνει την επαναχρησιμοποίηση.
Δυο προηγούμενα πρότυπα συνδυάστηκαν για να λυθεί το πρόβλημα αυτό. Μυστικότητα Πληροφοριών (Information Secrecy) και Ακεραιότητα Μηνυμάτων (Message Integrity). Το MDC πρέπει να υπολογίζεται στο αρχικό μη κρυπτογραφημένο μήνυμα. Και το κρυπτογραφημένο μήνυμα και το MDC στέλνονται στον Βασίλη. Αυτό το πρότυπο χρειάζεται μόνο ένα ζεύγος δημόσιου / ιδιωτικού κλειδιού για κρυπτογράφηση [29].



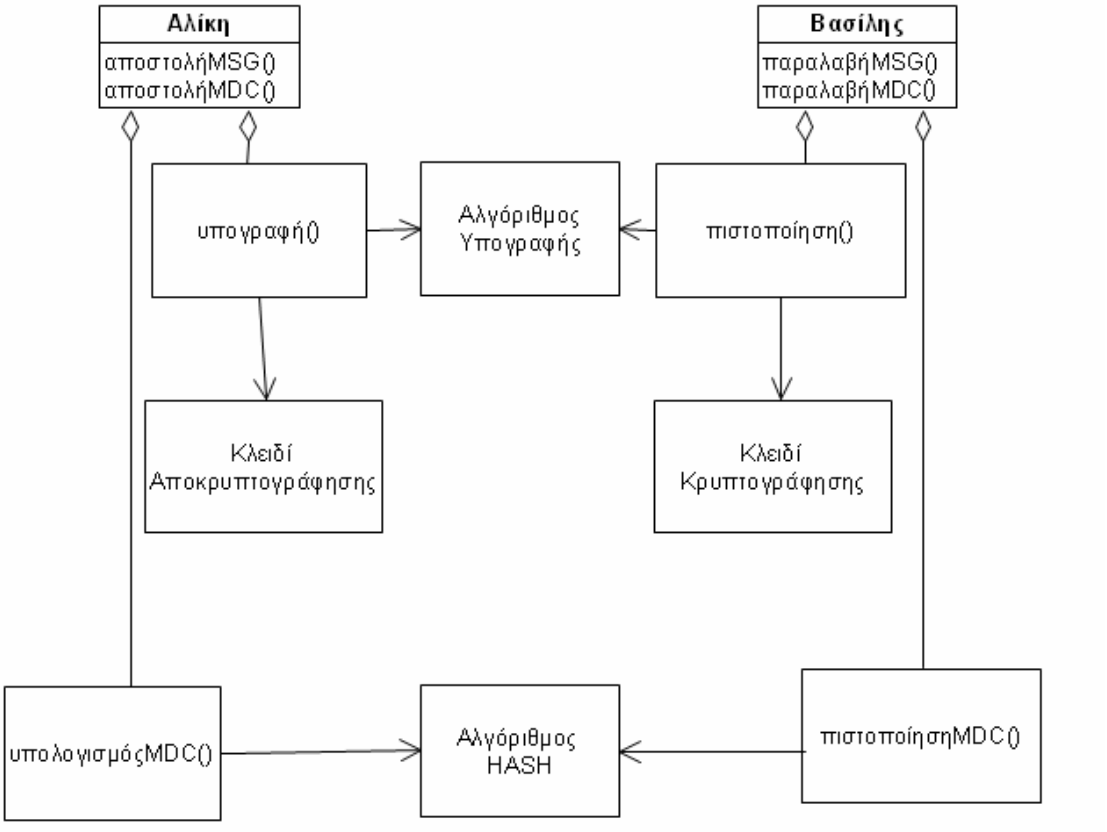
Σχήμα 9.2.7.1 Απεικονίζεται το πρότυπο Μυστικότητα με Ακεραιότητα.

9.2.8 Υπογραφή με Παράρτημα (Signature with Appentix).

Η Αλίκη και ο Βασίλης υπογράφουν τα μηνύματα που ανταλλάσουν με σκοπό να αποτρέψουν μετατροπές και αλλαγές και για να παρέχουν αυθεντικότητα αποστολέα. Χρειάζονται να χειρίζονται περιορισμένη μνήμη και υπολογιστικούς πόρους. Όμως τα μηνύματα που ανταλλάσουν είναι πολύ μεγάλα, άρα παράγουν και μεγάλες υπογραφές. Πώς θα μειωθεί ο χώρος αποθήκευσης που απαιτείται για ένα μήνυμα και της υπογραφής του, ενώ αυξάνεται η απόδοση του πρωτοκόλλου ψηφιακής υπογραφής; [29]

Όταν δεν χρησιμοποιείται καμία τεχνική μείωσης της υπογραφής, οι ψηφιακές υπογραφές είναι συνήθως όσο μεγάλες όσο και τα δεδομένα που υπογράφονται. Όμως, αν τα μηνύματα είναι μικρά, δεν χρειάζεται υπολογισμός για μείωση και της υπογραφής. Σημαντική είναι η σχέση μεταξύ επίδρασης πρόσθετου υπολογισμού και απόδοσης και η αναλογία μεταξύ του μεγέθους των μηνυμάτων και των υπογραφών τους.

Τα πρότυπα που συνδυάζονται για να λυθεί το πρόβλημα είναι το Αυθεντικότητα Αποστολέα (Sender Authentication) και Ακεραιότητα Μηνυμάτων (Message Integrity). Αυτό το πρότυπο εφαρμόζει πρωτόκολλο ψηφιακής υπογραφής σε μήνυμα τιμής hash, το οποίο είναι MDC. Η Αλίκη υπολογίζει μια τιμή hash του μηνύματος και την υπογράφει. Και το μήνυμα και η υπογεγραμμένη τιμή hash στέλνονται στον Βασίλη . Ο Βασίλης αποκρυπτογραφεί την υπογραφή και αποκαλύπτει την τιμή hash. Μετά υπολογίζει μια νέα τιμή hash και την συγκρίνει με αυτήν που βρήκε από την υπογραφή. Αν ταιριάζουν η υπογραφή είναι αληθινή [29].



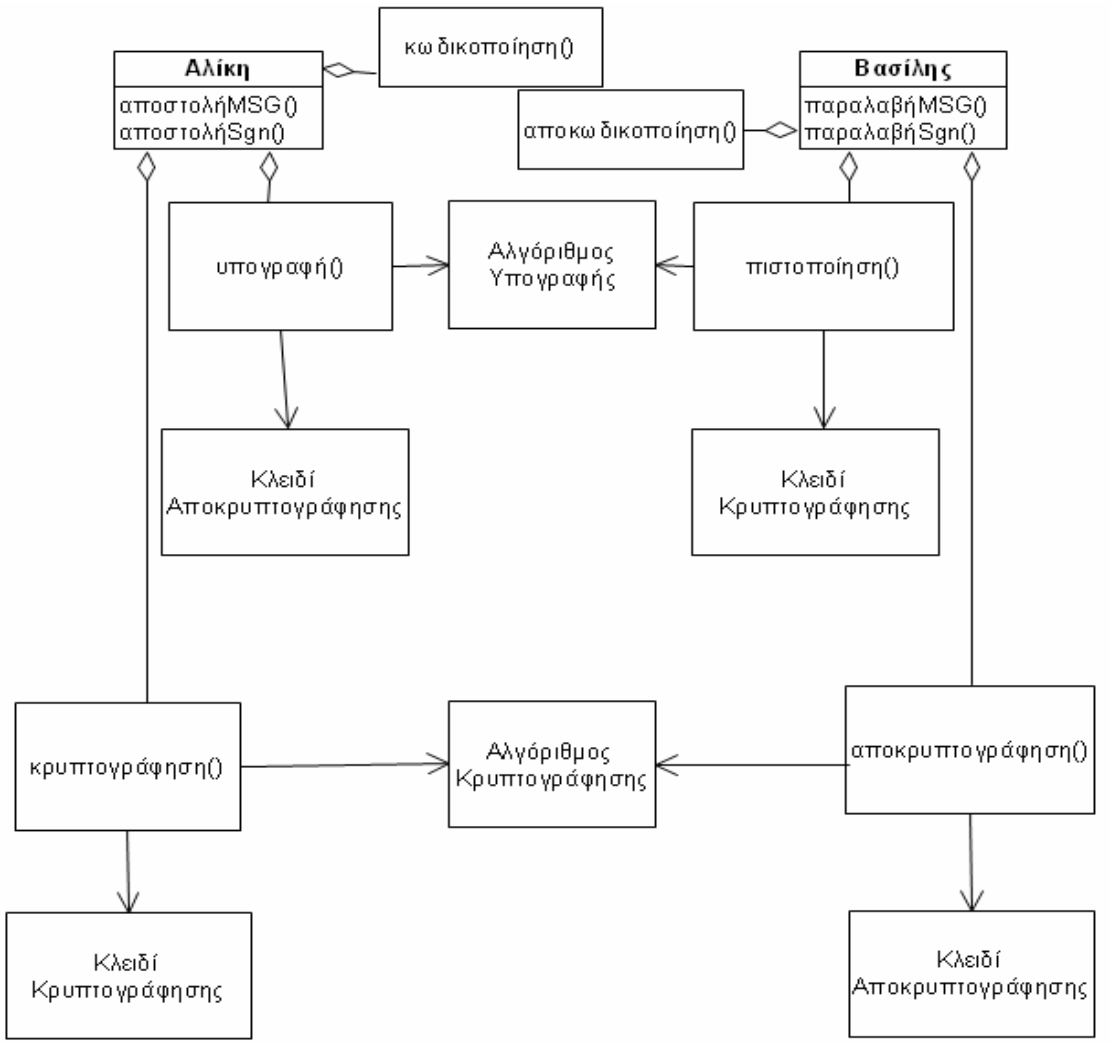
Σχήμα 9.2.8.1 Απεικονίζεται το πρότυπο Υπογραφή με Παράρτημα.

9.2.9 Μυστικότητα με Υπογραφή με Παράρτημα (Secrecy with Signature with Appentix).

Η Αλίκη και ο Βασίλης ανταλλάσουν κρυπτογραφημένα υπογεγραμμένα μηνύματα για να αποφύγουν αλλαγές και να επιτύχουν μυστικότητα και αυθεντικότητα του αποστολέα. Χρειάζεται να χειριστούν λίγους πόρους αποθήκευσης και διαχείρισης. Όμως τα μηνύματα που ανταλλάσουν είναι πολύ μεγάλα και παράγουν μεγάλες υπογραφές. Ήδη υπάρχει χρονική καθυστέρηση από την διαδικασία κρυπτογράφησης. Πώς θα μειωθεί η μνήμη που είναι απαραίτητη να αποθηκεύσει ένα μήνυμα και την υπογραφή του, καθώς αυξάνει η απόδοση του συστήματος, χωρίς να χαθεί η μυστικότητα;

Είναι δύσκολη απόφαση να συμπεριληφθεί ένα νέο βήμα επεξεργασίας, για να μειωθεί το μέγεθος της υπογραφής με ένα υπολογισμό που έχει δυο φάσεις επεξεργασίας, μια για να κρυπτογραφεί /αποκρυπτογραφεί δεδομένα και μια άλλη για να υπολογίζει και να πιστοποιεί μια υπογραφή. Η απόδοση θα μειωθεί λίγο, ώστε να αποθηκευτούν μεγάλα ποσά χώρου και να μειωθούν τα ποσά δεδομένων που θα μεταδοθούν. Ο συνδυασμός της Μυστικότητας Πληροφορίας (Information Secrecy) και Υπογραφής με Παράρτημα (Signature with Appentix) μπορεί εύκολα να γίνει, όταν χρησιμοποιείται σχεδιασμός, όπου η διαμόρφωση των συστατικών διευκολύνει την επαναχρησιμοποίηση.

Δυο πρότυπα συνδυάζονται για να λύσουν το πρόβλημα. Μυστικότητα Πληροφοριών (Information Secrecy) και Υπογραφή με Παράρτημα (Signature with Appentix). Η Αλίκη υπολογίζει μια τιμή hash του μηνύματος και το υπογράφει με το ιδιωτικό της κλειδί. Κρυπτογραφεί το αρχικό μήνυμα με το δημόσιο κλειδί του Βασίλη. Και το κρυπτογραφημένο μήνυμα και η υπογεγραμμένη τιμή hash στέλνονται στον Βασίλη. Εκείνος αποκρυπτογραφεί το κρυπτογραφημένο μήνυμα με το ιδιωτικό του κλειδί και αποκρυπτογραφεί την υπογεγραμμένη τιμή hash με το δημόσιο της Αλίκης. Αν ταιριάζουν, η υπογραφή είναι αληθινή. Η τεχνική ισχύει ακόμα και αν χρησιμοποιηθεί κρυπτογράφηση κλειδιού [29].

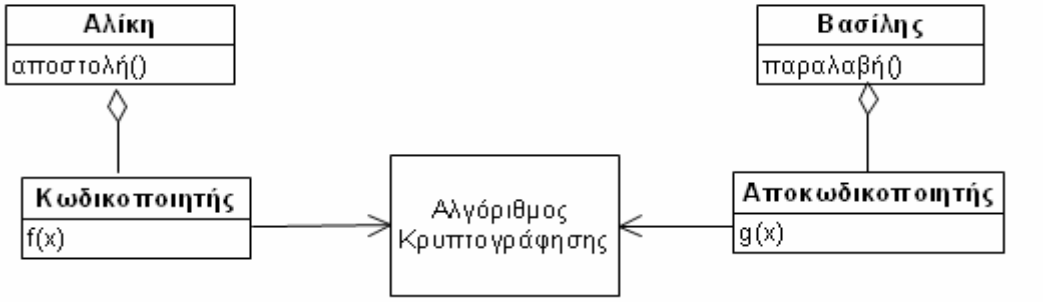


Σχήμα 9.2.9.1 Απεικονίζεται το πρότυπο Μυστικότητα με Υπογραφή με Παράρτημα.

9.2.10 Κρυπτογραφικό Μεταπρότυπο (Cryptographic Metapattern)

Δυο αντικείμενα, ο Βασίλης και η Αλίκη ανταλλάσουν δεδομένα μέσω μηνυμάτων. Χρειάζεται να γίνουν κρυπτογραφικοί μετασχηματισμοί στα δεδομένα. Χρειάζονται ένα συστατικό που θα είναι ευέλικτο και θα μπορεί να χρησιμοποιηθεί εύκολα μαζί με άλλους κρυπτογραφικούς μετασχηματισμούς. Πώς θα σχεδιαστεί μια ευέλικτη αντικειμενοστρεφής αρχιτεκτονική για ένα κρυπτογραφικό πρότυπο, ώστε να αυξηθεί η επαναχρησιμοποίηση των συστατικών; [29]
Όλοι οι μετασχηματισμοί κρυπτογράφησης έχουν μια γενική συμπεριφορά που μπορεί να γενικευθεί σε ένα ευέλικτο πρότυπο. Καλά δομημένες αρχιτεκτονικές λογισμικού διευκολύνουν την επαναχρησιμοποίηση και είναι ευέλικτες σε μελλοντικές εφαρμογές. Ευέλικτα και εύκολα προσαρμόσιμα συστήματα με κρυπτογράφηση βασισμένη σε απαιτήσεις ασφάλειας μπορούν να αποκτηθούν εύκολα, όταν οι αλγόριθμοι κρυπτογράφησης αποσυνδέονται από τις εφαρμογές τους και αυτά τα δυο αποσυνδέονται από τις υπηρεσίες κρυπτογράφησης, που χρησιμοποιούν τα συστήματα.
Η Αλίκη εκτελεί ένα μετασχηματισμό κρυπτογράφησης στα δεδομένα πριν τα στείλει στον Βασίλη. Ο Βασίλης λαμβάνει το μήνυμα και εκτελεί την αντίστροφη διαδικασία για να αποκαλύψει τα δεδομένα. Η Αλίκη και ο Βασίλης πρέπει να συμφωνούν για το

ποιο μετασχηματισμό θα χρησιμοποιήσουν και θα μοιραστούν ή θα διαμοιράσουν τα κλειδιά. Το διάγραμμα κλάσεων, που φαίνεται πιο κάτω, γενικεύει τον μετασχηματισμό κρυπτογράφησης με μια διεπαφή μετασχηματισμού και διαχωρίζει τους ρόλους Αποστολέα και Αποδέκτη από τους ρόλους Κωδικοποιητή και Αποκωδικοποιητή.
Το Κρυπτογραφικό Μεταπρότυπο (Cryptographic Metapattern) είναι υψηλότερου επιπέδου αφαίρεση για όλα τα πρότυπα κρυπτογράφησης. Το Μεταπρότυπο έχει τις κλάσεις Αλίκη, Βασίλης, Κωδικοποιητής, Αποκωδικοποιητής. Η κλάση Κωδικοποιητής έχει μια συνάρτηση f(x) που εκτελεί μετασχηματισμό κρυπτογράφησης στο x. Η κλάση Αποκωδικοποιητής έχει μια συνάρτηση g(x) που εκτελεί αντίστροφο μετασχηματισμό κρυπτογράφησης x=g(f(x)). Ο μετασχηματισμός και ο αντίστροφός του, βασίζονται στον ίδιο αλγόριθμο κρυπτογράφησης [29].



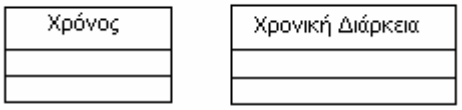
Σχήμα 9.2.10.1 Απεικονίζεται το διάγραμμα κλάσεων του Κρυπτογραφικού Μεταπροτύπου

9.3 Πρότυπα Χρόνου (Time Patterns)

Πολλά συστήματα πρέπει να έχουν σχέση με το χρόνο. Ο χρόνος μπορεί να είναι ακριβώς ένας πρόσθετος τύπος, όπως για την ημερομηνία γέννησης, ή μπορεί να είναι μια πρόσθετη διάσταση, όπως όταν απαιτείται μια ιστορία των δεδομένων. Θα παρουσιαστούν μερικά από τα πρότυπα που βοηθούν να σχεδιαστούν αυτά τα συστήματα.
Ο Christian August Crusius (1715-1775), Γερμανός φιλόσοφος έγραψε ότι κάθε υπάρχον πράγμα έχει την δικιά του θέση στο χώρο και το χρόνο [14]. Η βιομηχανία λογισμικού χρησιμοποιεί αντικείμενα για να αναπαραστήσει τα υπάρχοντα πράγματα. Επιπλέον, τα αντικείμενα μπορούν επίσης να αναπαραστήσουν τις έννοιες και τις ιδέες. Υπάρχουν μόνο λίγα έγγραφα σχετικά με το χρόνο, αλλά πολύ περισσότερα που εξετάζουν το χώρο, υποθέτοντας ότι ένα ιδιαίτερο αντικείμενο μπορεί να υπάρξει μόνο μια φορά.
Η προσθήκη του χρόνου σε ένα σύστημα είναι όπως η προσθήκη μιας άλλης διάστασης. Πολλά συστήματα έχουν την ανάγκη να εξεταστεί ο χρόνος. Ένα σύστημα για το διαχείριση των πόρων χρειάζεται τις περιόδους για την εξασφάλιση ότι ο ίδιος πόρος δεν χρησιμοποιείται δύο φορές οποιαδήποτε στιγμή. Ένα σύστημα ελέγχου αναθεώρησης πρέπει να ακολουθήσει τις αλλαγές των αρχείων με την πάροδο του χρόνου.
Τα πρότυπα που θα εξεταστούν είναι: Χρόνος ως Τύπος (Time as a Type), Timeserver, Ζευγάρια Χρόνου-Αξίας (Time-Value Pairs), Versioned Object, Χρονομετρημένη Αναφορά (Timed Reference), Προμηθευτής Ιστορίας (History Provider).

9.3.1 Χρόνος ως Τύπος (Time as a Type)

Ο χρόνος και οι χρονικές περίοδοι χρησιμοποιούνται συχνά. Μερικά περιβάλλοντα ανάπτυξης παρέχουν πολλούς τύπους για το χρόνο και την περίοδο, οι οποίοι μπορεί να είναι ή να μην είναι συμβατοί ο ένας με τον άλλο.
Έστω ότι η γλώσσα προγραμματισμού που χρησιμοποιείται δεν έχει κανέναν ενσωματωμένο τύπο για το χρόνο και την περίοδο. Το περιβάλλον προγραμματισμού έχει πολλούς τύπους για την αναπαράσταση του χρόνου ή/και της περιόδου και πρέπει να μειωθεί ο αριθμός των μετατροπών.
Οι τύποι για το χρόνο και την περίοδο που υπάρχουν στο περιβάλλον ανάπτυξης υποστηρίζουν ένα διαφορετικό σύνολο διαδικασιών και λειτουργιών κάθε ένας, αλλά κανένας από αυτούς δεν παρέχει τη λειτουργία που πρέπει να υπάρχει. Επίσης πρέπει να ελαχιστοποιηθεί ο αριθμός τύπων χρόνου και περιόδου που πρέπει να γίνει γνωστός.
Σε μερικές γλώσσες προγραμματισμού υπάρχει μια ολόκληρη δέσμη διαφορετικών βιβλιοθηκών με διαφορετικές εφαρμογές των κλάσεων για το χρόνο και τις χρονικές περιόδους. Αυτοί οι τύποι δεν είναι συχνά συμβατοί. Η ακρίβεια των διαθέσιμων τύπων μερικές φορές δεν είναι ικανοποιητική, για παράδειγμα να μην υποστηριχτούν περίοδοι ενός έτους ή περισσότερο.
Πρέπει να καθοριστεί και εφαρμοστεί μια κλάση και για το χρόνο και για την χρονική περίοδο. Παρέχει ένα πλούσιο σύνολο διαδικασιών μετατροπής μεταξύ των πρόσφατα εφαρμοσμένων κλάσεων και ήδη των υπαρχουσών [16].



Σχήμα 9.3.1.1 Απεικονίζονται οι κλάσεις Χρόνος και Χρονική Διάρκεια.
Τα πραγματικά δεδομένα διατηρούνται στα μέλη δεδομένων. Ο αριθμός και οι τύποι των μελών δεδομένων εξαρτώνται από τις απαιτήσεις. Μια καλή ανταλλαγή και για τις δύο κλάσεις είναι ότι και οι δύο έχουν τα μέλη δεδομένων για το έτος, το μήνα, την ημέρα, την ώρα, το λεπτό, το δευτερόλεπτο και τα χιλιοστά του δευτερολέπτου.
Εάν ο τύπος πρέπει να βελτιστοποιηθεί για το διάστημα, κατόπιν όσο το δυνατόν περισσότερα μέλη δεδομένων πρέπει να παραλειφθούν, όπως σε μερικές περιπτώσεις τα χιλιοστά του δευτερολέπτου δεν απαιτούνται. Εάν ο τύπος πρέπει να βελτιστοποιηθεί για ακρίβεια, τότε τα πρόσθετα μέλη μπορούν να προστεθούν στις κλάσεις. Και για τις δύο κλάσεις διάφορες μέθοδοι πρέπει να εφαρμοστούν, όπως μετατροπή από/σε άλλους τύπους των περιβαλλόντων ανάπτυξης.
Οι θετικές συνέπειες του προτύπου είναι οι ακόλουθες: Ο νέος τύπος δεδομένων μπορεί να χρησιμοποιηθεί κατά τον ίδιο τρόπο με τους ενσωματωμένους τύπους δεδομένων. Μόνο ένας τύπος για το χρόνο και την περίοδο χρησιμοποιείται σε όλο το πηγαίο κώδικα των εφαρμογών. Αυτό μειώνει τις δαπάνες συντήρησης και εκμάθησης. Επιπρόσθετα, ο νέος τύπος δεδομένων μπορεί να χρησιμοποιηθεί οπουδήποτε ένας διαφορετικός τύπος παραμέτρου χρόνου/περιόδου απαιτείται, δεδομένου ότι έχουν εφαρμοστεί οι ρουτίνες μετατροπής. Η ακρίβεια του τύπου μπορεί να προσαρμοστεί στις απαιτήσεις εφαρμογών [16].

Ένα μειονέκτημα είναι ότι η λύση λειτουργεί μόνο για τις γλώσσες που επιτρέπουν τον καθορισμό των νέων τύπων, όπως η C++ και η Java.

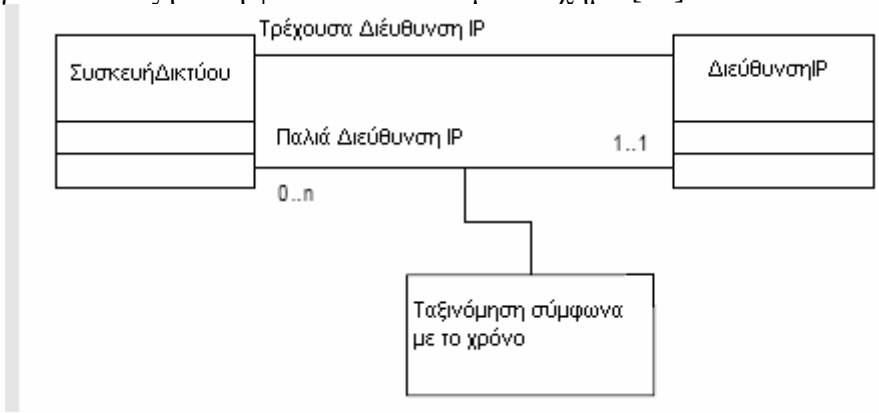
9.3.2 Πρότυπο Timeserver

Διάφοροι λόγοι υπάρχουν γιατί ο χρόνος συστήματος μιας μηχανής μπορεί να αλλάξει. Μερικές εφαρμογές πρέπει να έχουν μια αξιόπιστη χρονική σειρά, το οποίο σημαίνει ότι πρέπει να εγγυηθεί ότι για κάθε κλήση της χρονικής λειτουργίας η προκύπτουσα τιμή είναι νεότερη από όλες τις τιμές για τις προηγούμενες κλήσεις [16].
Μια πιθανή λύση στο πρόβλημα είναι πρόσθετο υλικό (hardware), που λαμβάνει τα σήματα από ένα ατομικό ρολόι. Ο χρόνος συστήματος συγχρονίζεται με το διαβιβασθέντα χρόνο. Ακόμα και όταν ένα σήμα ατομικών ρολογιών είναι διαθέσιμο, αυτό δεν λύνει όλα τα προβλήματα.
Πρέπει να ταξινομηθούν τα στοιχεία σύμφωνα με το χρόνο παρά τη δυνατότητα ότι ο χρήστης μπορεί να χειριστεί το χρόνο συστημάτων. Επίσης, η αίτηση χρειάζεται μια σειρά χρονικών γραμματοσήμων που είναι εγγυημένη για να ταξινομηθεί σε σειρά ανόδου.
Η λύση που δίνεται είναι η ακόλουθη. Τρέχει μια υπηρεσία συνεχώς ενώ και το σύστημα τρέχει. Πρέπει να είναι σίγουρο ότι η υπηρεσία τρέχει ενώ μια διαδικασία τρέχει, που αλλάζει το ρολόι συστήματος προγραμματιστικά ή ένας χρήστης συνδέεται που μπορεί να αλλάξει το ρολόι. Ο timeserver καθώς τρέχει διατηρεί έναν κατάλογο χρόνο-χρόνο ζευγαριών. Κάθε ζευγάρι αποτελείται από timestamp που αντιπροσωπεύει την πραγματική ώρα συστημάτων και το άλλο timestamp αντιπροσωπεύει το χρόνο που παρεκτείνεται των προηγούμενων χρόνων.
Οι συνέπειες του προτύπου αυτού είναι οι εξής [16]:
1). Χρησιμοποιώντας το TimeServer είναι δυνατόν να ταξινομηθούν τα στοιχεία με το χρόνο, ανεξάρτητα από τις αλλαγές στο ρολόι συστήματος. Ο TimeServer παρέχει έναν κατάλογο τιμών μεταξύ του ρολογιού συστήματος που συνδέονται με το στοιχείο και του πραγματικού χρόνου.
2). Είναι δυνατό να ακολουθηθούν οι αλλαγές στο ρολόι συστήματος. Το ιστορικό περιέχει τις πληροφορίες, πότε άλλαξαν το ρολόι, από ποια τιμή το άλλαξαν και τι ώρα το ρολόι συστήματος θα ήταν, εάν η αλλαγή δεν θα είχε γίνει.
3). Εάν η εφαρμογή παίρνει πάντα timestamps από τον timeserver, είναι εγγυημένο ότι παράγονται στη σειρά ανόδου.
4). Πρέπει να εξασφαλιστεί ότι η υπηρεσία τρέχει όλη την ώρα. Αυτό μπορεί να είναι ένα πρόβλημα, εάν ο φόρτος εργασίας ενός συστήματος είναι ήδη υψηλός.

9.3.3 Ζευγάρια Χρόνου-Αξίας (Time-Value Pairs)

Μερικά συστήματα πρέπει να ακολουθήσουν την ιστορία μιας ενιαίας ιδιότητας σε ένα αντικείμενο. Πρέπει όμως να καταναλωθεί όσο το δυνατόν λιγότερο μνήμη για την ιστορία. Επίσης μπορεί να χρειάζεται αναίρεση των αλλαγών σε μια τιμή μιας ιδιότητας αντικειμένου.
Η λύση είναι η εξής: Εφαρμόζονται οι ιδιότητες ως διατεταγμένη συλλογή με τα ζευγάρια χρόνου-τιμής. Τα ζευγάρια διατάσσονται με το χρόνο. Όταν μια αλλαγή στην τιμή εμφανίζεται, ένα νέο ζευγάρι δημιουργείται και προστίθεται στην ουρά της συλλογής, η οποία διατάσσεται με βάση το χρόνο.
Για παράδειγμα κάθε συσκευή δικτύων μπορεί να έχει διαφορετικές διευθύνσεις IP με την πάροδο του χρόνου, π.χ. όταν κινείται από ένα υποδίκτυο σε ένα διαφορετικό

υποδίκτυο. Για να παρακολουθήσει τη διεύθυνση IP, ένας κατάλογος διευθύνσεων IP προστίθεται στην κλάση ΣυσκευήΔικτύου. Ο κατάλογος διατάσσεται με βάση το χρόνο. Το τελευταίο στοιχείο στον κατάλογο είναι το πιο τρέχον. Η μεταβλητή Τρέχουσα Διεύθυνση IP πάντα περιέχει την τιμή του τελευταίου στοιχείου στον κατάλογο. Η διάταξη αυτή φαίνεται στο επόμενο σχήμα [16].



Σχήμα 9.3.3.1 Απεικονίζεται η δομή του προτύπου για το προηγούμενο παράδειγμα.

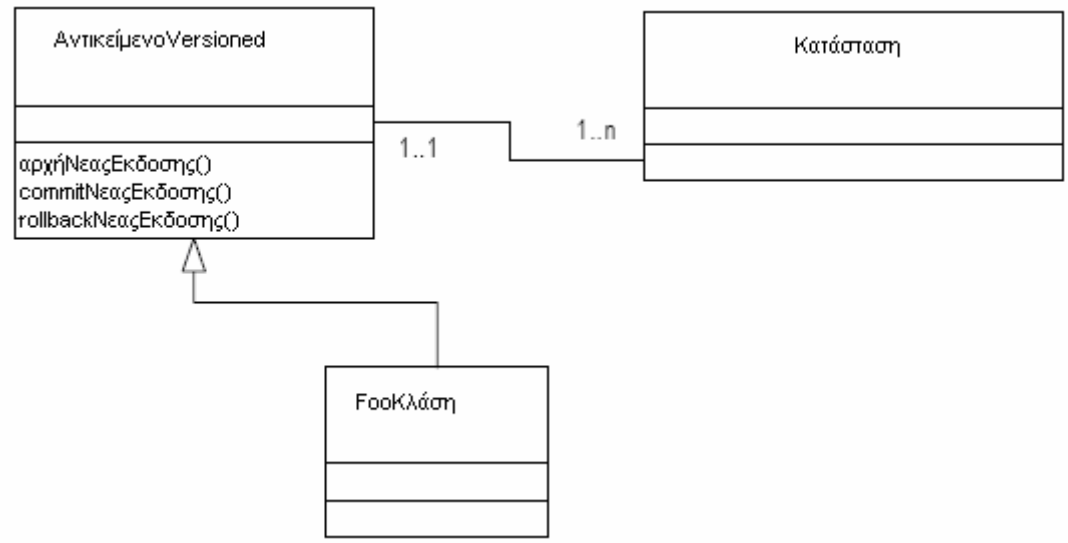
Ο πελάτης έχει τώρα τη δυνατότητα να ανακτήσει την τιμή, η οποία συνδέεται με έναν συγκεκριμένο χρόνο. Μερικά προβλήματα είναι [16]:
1. Τι συμβαίνει, εάν ένας συνεργάτης ενός αντικειμένου συσκευών δικτύων προσπαθεί να πάρει μια τιμή για ένα χρόνο που είναι παλαιότερος από την παλαιότερη είσοδο στη συλλογή ΠαλίαΔιεύθυνση IP;
Αυτό μπορεί να λυθεί με διάφορους τρόπους, όπως με την επιστροφή ενός μηδενικού αντικειμένου [15] ή με τη εισάγοντας μια εξαίρεση (εάν η γλώσσα προγραμματισμού την υποστηρίζει).
2. Τι συμβαίνει, εάν ένας συνεργάτης ενός αντικειμένου συσκευών δικτύων προσπαθεί να πάρει ένα ζευγάρι χρόνου-τιμής που δεν υπάρχει;
Δύο σημαντικές λύσεις είναι δυνατές εδώ. Όποια επιλεγεί εξαρτάται από τις απαιτήσεις της εφαρμογής:
Κατ' αρχάς, ένα μηδενικό αντικείμενο μπορεί να επιστραφεί εδώ ή μια εξαίρεση μπορεί να εφαρμοστεί. Αυτή η λύση να είναι κατάλληλη για παράδειγμα για μια καιρική εφαρμογή που έχει τη θερμοκρασία μιας θέσης για δύο μη-διαδοχικές ημέρες. Η εφαρμογή δεν μπορεί να επιστρέψει απλά την προηγούμενη τιμή γιατί οι ημερομηνίες που είναι μεταξύ των δύο ημερών, όπως εκείνη η τιμή δεν είναι κανονικά σωστή.
Η δεύτερη δυνατότητα είναι ότι το αντικείμενο επιστρέφει το πιο στενό ζευγάρι χρόνου-τιμής ή το ζευγάρι χρόνου-τιμής με προηγούμενο timestamp.
Οι συνέπειες που έχει το πρότυπο είναι οι εξής [16]:
1). Μπορεί να υπάρχει πρόσβαση στην τρέχουσα τιμή με τον ίδιο τρόπο, όπως πριν, έτσι καμία αλλαγή δεν είναι απαραίτητη στον κώδικα των αντικειμένων συνεργασίας.
2). Μπορεί να αναιρεθούν όλες τις αλλαγές στην τιμή.
3). Γίνεται οικονομία στους πόρους, αφού μόνο αλλαγές αποθηκεύονται.
4). Μπορεί να ανακτηθεί οποιαδήποτε τιμή του παρελθόντος, ακόμη και για έναν χρόνο, όπου κανένα ζευγάρι χρόνου-τιμής δεν είναι διαθέσιμο. Στην τελευταία περίπτωση, θα επιστρεφόταν ακριβώς η προηγούμενη τιμή. Για παράδειγμα η διεύθυνση στις 08:00 ήταν η 1.1.1.1 και η διεύθυνση στις 09:00 ήταν η 2.2.2.2. Όταν αναζητείται η τιμή στις 08:30, η τιμή θα ήταν ακόμα η 1.1.1.1, δεδομένου ότι αυτή η τιμή ισχύει από τις 08:00 μέχρι στις 09:00.

5). Πρέπει να παρέχεται σε μια πρόσθετη διεπαφή για τη διευκρίνιση του χρόνου. Η διεπαφή της κλάσης γίνεται λίγο δυσκολότερο να χρησιμοποιηθεί.
6). Πρέπει να εφαρμοστεί ο κατάλογος και τα σωστά ζευγάρια προσθήκης και ανάκτησης χρόνου-τιμής.

9.3.4 Versioned Αντικείμενο

Τα ζευγάρια χρόνος-τιμή (Time -Value) μπορεί να μην είναι ικανοποιητικά για μερικά συστήματα ή μπορεί να μην είναι εφικτό να ακολουθηθούν διάφορες ιδιότητες χρησιμοποιώντας την προσέγγιση χρόνου-τιμής. Μεταβάλλοντας διάφορες ιδιότητες, τη μια μετά την άλλη, μπορεί να οδηγήσει σε ενδιάμεσες καταστάσεις αντικειμένων, όπου μόνο μερικές από τις ιδιότητες αλλάζουν. Αυτές οι ενδιάμεσες καταστάσεις δεν παρουσιάζουν κανένα ενδιαφέρον στην εφαρμογή.
Έστω ότι μια κλάση παρέχει δύο συναρτήσεις την f1 και την f2. Η συνάρτηση f1 αλλάζει μόνο την ιδιότητα a1 και η f2 αλλάζει την ιδιότητα a2. Έστω ότι στην αρχή, η ιδιότητα a1 έχει την τιμή v11 και η a2 έχει την τιμή v21. Αφότου έχει καλέσει ο πελάτης και τις δύο λειτουργίες η a1 έχει την τιμή v12 και η a2 έχει v22. Η f1 καλείται πρώτη. Η f2 καλείται μετά. Η σειρά των καταστάσεων ενός ιδιαίτερου αντικειμένου μπορεί να αναπαρασταθεί από την ακόλουθη σειρά (που διαβάζεται το s (α, β) ως «η κατάσταση αποτελείται από τις τιμές a και b»):
s1 (v11, v21) • s2 (v12, v21) • s3 (v12, v22)
Εάν το αντικείμενο θα ήταν επίμονο (persistent), και οι τρεις καταστάσεις μπορούν ακόμα να ισχύουν. Εντούτοις, από την άποψη εφαρμογών, μόνο οι καταστάσεις s1 και s3 είναι ενδιαφέρουσες. Ο πελάτης δεν θέλει να κρατήσει την ενδιάμεση κατάσταση s2. Γενικά, κατά την κλήση μιας μεθόδου σε ένα αντικείμενο, το αντικείμενο κάνει μια μετάβαση από μια κατάσταση σε μια άλλη [16].
S1 • S2 • S3 • S4 • S5 • S6 • … • Sn
Είναι επιθυμητό να κρατηθούν μόνο επιλεγμένες καταστάσεις ενός αντικειμένου στην ιστορία και πρέπει να διατηρηθεί η συνέπεια του αντικειμένου ακόμα και όταν αλλάζουν πολλές ιδιότητες. Επίσης είναι επιθυμητό να ακολουθηθούν οι επίλεκτες καταστάσεις των αντικειμένων των διαφορετικών κλάσεων. Σε μερικές περιπτώσεις είναι χρήσιμο να διατηρηθεί η ενθυλάκωση των ακολουθημένων αντικειμένων. Δεν γίνεται να χρησιμοποιηθεί ο μηχανισμός συναλλαγής του ελλοχεύοντος συστήματος βάσεων δεδομένων, δεδομένου ότι δεν ξέρει για τη σημασιολογία της εφαρμογής.
Η λύση που παρουσιάζεται ταιριάζει σε μηχανισμούς συναλλαγής των συστημάτων βάσεων δεδομένων. Ένας versioning μηχανισμός προστίθεται στο αντικείμενο. Η διαδικασία δημιουργίας νέων εκδόσεων υπακούει την ACID αρχή των συναλλαγών. Το ACID είναι η σύντμηση για την ατομικότητα, τη συνέπεια, την απομόνωση, και τη διάρκεια.
Ατομικότητα σημαίνει, ότι για κάθε συναλλαγή, είτε όλες, είτε καμία από τις αλλαγές στη συναλλαγή παραμένουν, όταν τελειώνει η συναλλαγή. Η συναλλαγή μπορεί να τελειώσει είτε επιτυχής, είτε μπορεί να ακυρωθεί από το χρήστη είτε το σύστημα, π.χ. λόγω ενός αδιεξόδου.
Εάν μια συναλλαγή αλλάζει τη βάση δεδομένων, τότε μια νέα κατάσταση πρέπει να είναι η σωστή κατάσταση της βάσης δεδομένων. Αυτό καλείται συνέπεια.
Εάν καμία από τις συναλλαγές δεν επηρεάζεται από οποιεσδήποτε άλλες συναλλαγές, τότε η συναλλαγή είναι απομονωμένη.
Όταν μια συναλλαγή ολοκληρώσει όλες τις αλλαγές, πρέπει να εμείνει ακόμη και στην περίπτωση διακοπής του συστήματος. Αυτό καλείται διάρκεια.

Κάθε συνεργάτης που έχει πρόσβαση στο versioned αντικείμενο αρχίζει μια νέα έκδοση, με την κλήση της αρχήΝεαςΕκδοσης(), καλεί τις μεθόδους σε εκείνο το αντικείμενο και δεσμεύει την έκδοση. Οποιαδήποτε στιγμή, μόνο ένας συνεργάτης μπορεί να δημιουργήσει μια νέα έκδοση ενός αντικειμένου [16].



Σχήμα 9.3.4.1 Απεικονίζεται το διάγραμμα κλάσεων του Versioned Αντικειμένου.
Ξεκινώντας μια νέα έκδοση, σημαίνει κλείδωμα του αντικειμένου και για τους δύο, την πρόσβαση ανάγνωσης και για την πρόσβαση γραψίματος. Η εφαρμογή της commitΝέαςΕκδοσης() ή rollbackΝεαςΕκδοσης() αφαιρεί την κλειδαριά.
Εάν ένας δεύτερος συνεργάτης προσπαθεί να έχει πρόσβαση σε ένα αντικείμενο για το οποίο μια νέα έκδοση είναι υπό κατασκευή, τότε η δεύτερη κλήση της αρχήΝεαςΕκδοσης() εμποδίζεται, έως ότου δεσμεύσει ο πρώτος συνεργάτης τη νέα έκδοσή του [16].
Το Versioned αντικείμενο μπορεί να έχει αναφορές σε άλλα Versioned αντικείμενα. Όταν ένας πελάτης αρχίζει μια νέα έκδοση το Versioned αντικείμενο είναι κλειδωμένο. Ο πελάτης δεν κλειδώνει αυτόματα τα παραπεμφθέντα αντικείμενα. Εξαρτάται από τις εφαρμογές των μεθόδων αντικειμένων εάν και πότε θα κλειδώσουν Versioned αντικείμενα. Εάν μια μέθοδος θέλει να έχει πρόσβαση στο Versioned αντικείμενο, η εφαρμογή μεθόδου πρέπει να αρχίσει μια νέα έκδοση, να χρησιμοποιήσει το παραπεμφθέν αντικείμενο και δεσμεύει τελικά τη νέα έκδοση.
Η λύση μπορεί να ενισχυθεί με την προσθήκη μιας παραμέτρου στην αρχήΝεαςΕκδοσης() - κλήση που δείχνει, εάν το αντικείμενο θα προσεγγιστεί για πρόσβαση ανάγνωσης ή πρόσβαση γραψίματος. Επιτρέποντας άλλα πρότυπα συναγωνισμού μπορεί να ενισχυθεί ο παραλληλισμός. Τα αδιέξοδα μπορούν να εμφανιστούν, εάν πολλά νήματα προσπαθούν να δημιουργήσουν τις νέες εκδόσεις του ίδιου αντικειμένου συγχρόνως.
Οι συνέπειες του προτύπου είναι [16]:
1). Τα αντικείμενα έχουν πάντα έχουν συνεπείς καταστάσεις, ακόμα και όταν χρησιμοποιούνται από επισκέπτες που εκτελούν από διαφορετικά νήματα.
2). Μπορεί να χρησιμοποιηθούν οι έξυπνοι δείκτες (pointers), για να διευκολυνθεί η χρήση του αντικείμενου. Ο έξυπνος δείκτης κρύβει το κώδικα (σχετικό με versioning) στην εφαρμογή του.
3). Οι πόροι συστημάτων περιορίζουν μόνο τον αριθμό αλλαγών κατάστασης.

4). Ενώ ένας πελάτης δημιουργεί μια νέα έκδοση του Versioned αντικειμένου, ο πελάτης μπορεί να είναι βέβαιος ότι κανένας άλλος πελάτης δεν αλλάζει την κατάσταση εκείνου του αντικειμένου.
5). Εάν ένα αντικείμενο Versioned αναφέρεται σε άλλα αντικείμενα Versioned, κατόπιν το αντικείμενο Versioned κλειδώνει τα παραπεμφθέντα αντικείμενα επίσης, εάν το αντικείμενο Versioned θέλει να το κάνει αυτό.
6). Οι κλάσεις αντικειμένων Versioned είναι πιο δύσκολες να εφαρμοστούν από τις κανονικές κλάσεις.
7). Τα αντικείμενα Versioned εκτελούν πιο αργά, λόγω των πρόσθετων γενικών εξόδων.
8). Ένας πελάτης μπορεί να έχει πρόσβαση στο Versioned αντικείμενο μόνο στα πλαίσια μιας νέας έκδοσης. Αυτό ισχύει επίσης για τη πρόσβαση ανάγνωσης.
9). Η προτεινόμενη λύση δεν λειτουργεί, εάν δημιουργώντας μια νέα έκδοση παίρνει πολύ χρόνο.

9.3.5 Χρονομετρημένη Αναφορά (Timed Reference)

Γενικά, ένα αντικείμενο είναι πρόσβαση μέσω αναφοράς. Ένας πελάτης μπορεί να καλέσει τα δημόσια μέλη του αντικειμένου μέσω αυτής της αναφοράς. Με άλλα λόγια: Συνδεμένο με την αναφορά έχουμε ένα αντικείμενο με μια συγκεκριμένη κατάσταση.
Εάν πολλές καταστάσεις του ίδιου αντικειμένου κρατούνται σε μια ιστορία, τότε μια παραδοσιακή αναφορά δεν είναι ικανοποιητική από εννοιολογική άποψη, που δείχνει πάντα την τρέχουσα έκδοση του αντικειμένου (οι προηγούμενες καταστάσεις απορρίπτονται). Πώς πρέπει μια αναφορά να επεκταθεί, να υποστηρίζει επίσης την αναφορά των προηγούμενων καταστάσεων του αντικειμένου που αναφέρεται;
Πρέπει να χρησιμοποιηθεί μια χρονομετρημένη αναφορά (Timed Reference) παρόμοια με μια κανονική αναφορά. Επίσης η αναφορά είναι χρήσιμη για να υποστηριχτούν ιστορικές καταστάσεις ενός ιδιαίτερου αντικειμένου.
Η λύση είναι η εξής [16]:
Εφαρμόζεται μια κλάση που αναπαριστά μια αναφορά. Επιτρέπει σε μέσα να βάλουν παραμέτρους σε κάθε στιγμιότυπο εκείνης της κλάσης με timestamp κατά τη διάρκεια του χρόνου εκτέλεσης:
Χρησιμοποιούνται τα στιγμιότυπα της κλάσης ΧρονομετρημένηΑναφορά, για να υπάρχει αναφορά στα αντικείμενα που έχουν τις πολλές καταστάσεις σε έναν κατάλογο ιστορίας. Εξετάζοντας το αντικείμενο ΧρονομετρημένηΑναφορά, το παραπεμφθέν αντικείμενο μπορεί να αποφασίσει ποια κατάσταση πρέπει να φορτώσει κατά την πρόσβαση.

Σχήμα 9.3.5.1 Απεικονίζεται το διάγραμμα κλάσεων του προτύπου Χρονομετρημένη Αναφορά.

Οι συνέπειες του έχει το πρότυπο είναι [16]:

1). Μπορεί να χρησιμοποιηθεί η Χρονομετρημένη Αναφορά με τον ίδιο τρόπο, όπως οι κανονικές αναφορές.

2). Μπορεί να υπάρχει πρόσβαση σε παλαιότερες καταστάσεις ενός αντικειμένου.

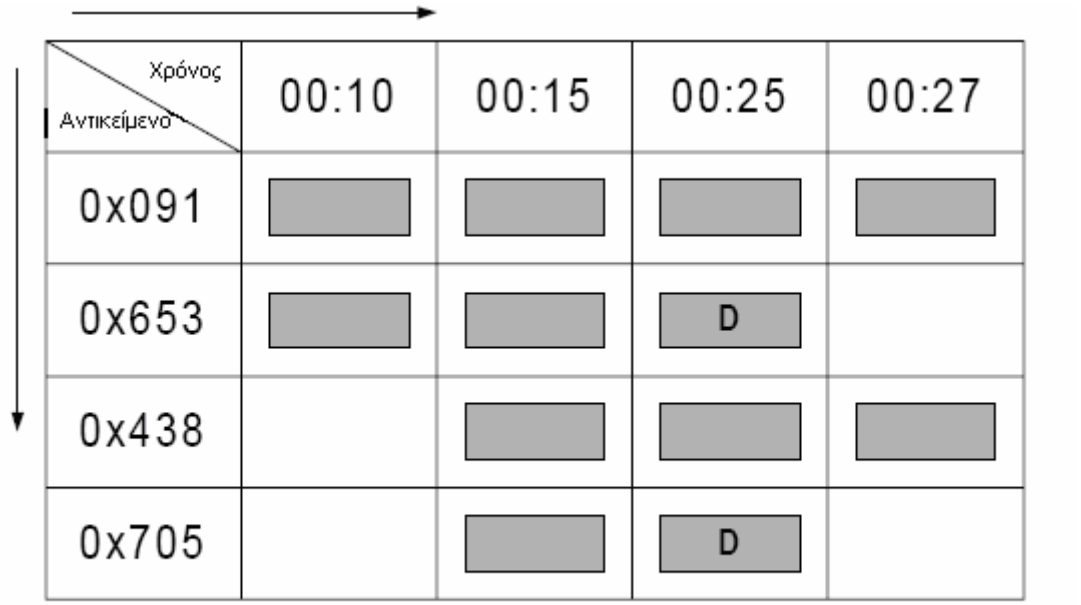
3). Το πρότυπο Χρονομετρημένη Αναφορά προσθέτει ένα επίπεδο εμμεσότητας (indirection), το οποίο προσθέτει επιβάρυνση. Με την προσθήκη ενός περαιτέρω επιπέδου indirection, το πρότυπο Χρονομετρημένη Αναφορά προσθέτει επιβάρυνση στην εφαρμογή σας.

9.3.6 Προμηθευτής Ιστορίας (History Provider)

Δυσεπίλυτα προβλήματα προκύπτουν όταν αναφέρονται τα αντικείμενα σε άλλα αντικείμενα, όπως με ενώσεις ή συναθροίσεις. Έχοντας την κατοχή μιας ιστορίας ανά αντικείμενο δεν είναι ικανοποιητικό. Ανεξάρτητα αντικείμενα δεν μπορούν να λύσουν το πρόβλημα της συνέπειας της ιστορίας διαφορετικών αντικειμένων που σχετίζονται το ένα με το άλλο [16].

Πρέπει να ακολουθηθούν οι αλλαγές όχι μόνο των αντικειμένων αλλά και των ενώσεων μεταξύ τους. Επίσης πρέπει να υπάρχει η δυνατότητα πλοήγησης πίσω για ένα αντικείμενο και να γίνεται εναλλαγή μεταξύ της κανονικής πλοήγησης και της πλοήγησης σε χρόνο στο παρελθόν.

Η λύση αποτελείται από διάφορα συστατικά. Η βασική ιδέα είναι να καταγραφούν όλες οι αλλαγές κατάστασης κάθε αντικειμένου. Ο προμηθευτής ιστορίας το κάνει αυτό. Παρακολουθεί το δρόμο που ακολούθησαν και την κατάσταση των αντικειμένων. Ένας πίνακας αναπαριστά τα αντικείμενα και τις καταστάσεις τους [16]:



Σχήμα 9.3.6.1 Απεικονίζεται ο πίνακας που αναπαριστά τα αντικείμενα και τις καταστάσεις τους.

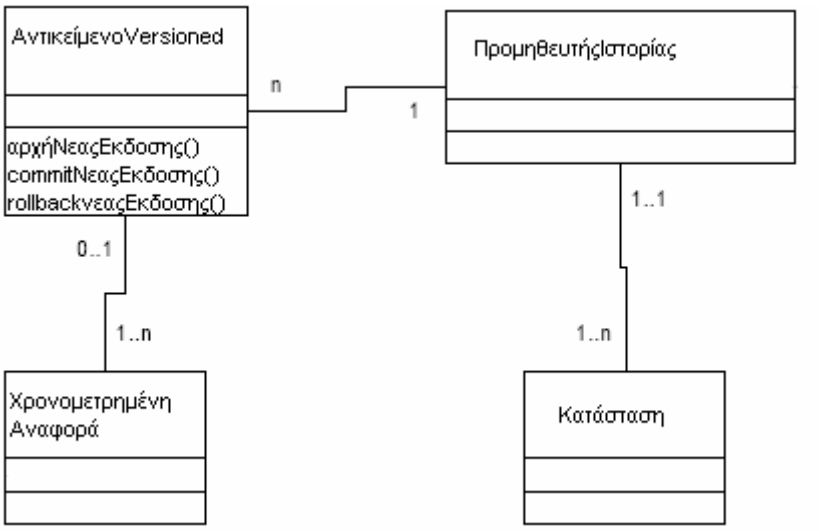
Ο πίνακας έχει δύο διαστάσεις: μια διάσταση είναι το προσδιοριστικό αντικειμένου, και η άλλη διάσταση είναι ο χρόνος. Με άλλα λόγια κάθε σειρά στον πίνακα αντιπροσωπεύει την ιστορία της κατάστασης του αντικειμένου, και κάθε στήλη αντιπροσωπεύει το σύνολο των καταστάσεων που είναι το αποτέλεσμα μιας συναλλαγής.

Το γκρι τετράγωνο δείχνει ότι μια κατάσταση για το αντικείμενο με εκείνη την ταυτότητα ID υπάρχει εκείνη τη στιγμή. Το γράμμα «D» σε ένα γκρίζο τετράγωνο δείχνει ότι το αντικείμενο έχει διαγραφεί εκείνη τη στιγμή. Δεν είναι απαραίτητο, οι χρονικές εκτάσεις μεταξύ των στηλών να είναι ίδιες [16].

Ο πίνακας παρουσιάζει τέσσερα αντικείμενα με διαφορετικό πρόγραμμα ζωής. Το αντικείμενο 0x091 υπάρχει όλο κατά τη διάρκεια της πλήρης περιόδου. Το αντικείμενο 0x653 έχει διαγραφεί στις 00:25. Το αντικείμενο 0x0438 δημιουργήθηκε στις 00:15. Τέλος, το αντικείμενο 0x705 δημιουργήθηκε στις 00:15 και διαγράφηκε στις 00:25.

Η λύση χειρίζεται τα χρονικά χάσματα ως εξής: Εάν ο πελάτης ζητήσει την έκδοση ενός αντικειμένου για έναν χρόνο και δεν είναι στον πίνακα, τότε ο προμηθευτής ιστορίας θα επιστρέψει την έκδοση που ήταν τελευταίο πριν από εκείνο τον ζητούμενο χρόνο. Εδώ, η υπόθεση είναι ότι όλα τα αντικείμενα αλλάζουν την κατάστασή τους συνεχώς. Δεν υπάρχει καμία συγκεκριμένη στιγμή, όπου δεν έχουν μια κατάσταση και κάθε κατάσταση παραμένει η ίδια, μέχρι να μεταβεί το αντικείμενο σε μια νέα κατάσταση.

Η δομή του προτύπου απεικονίζεται στο επόμενο σχήμα [16].



Σχήμα 9.2.6.2 Απεικονίζεται η δομή του προτύπου Προμηθευτής Ιστορίας
Οι συνέπειες του προτύπου είναι [16]:
1). Υπάρχει και παραδοσιακή, αλλά και έγκαιρη πλοήγηση για οποιαδήποτε από τα αντικείμενα Versioned.
2). Μπορεί εύκολα να αλλάξει μεταξύ των δύο μεθόδων πλοήγησης (παραδοσιακής εναντίον της έγκαιρης).
3). Μπορεί να χρησιμοποιηθεί ο Προμηθευτής ιστορίας, όταν χρησιμοποιείται επίσης τις χρονομετρημένες αναφορές (Timed Refereces).
4). Μπορεί να υπάρχει πρόσβαση στα αντικείμενά με έναν κανονικό τρόπο, εάν δεν υπάρχει ενδιαφέρον για το χαρακτηριστικό γνώρισμα ιστορίας.
5). Τα γενικά έξοδα εφαρμογής είναι δευτερεύουσας επιρροής, εάν η εφαρμογή χρησιμοποιεί επίσης συναλλαγές μιας βάσης δεδομένων.
6). Ο Προμηθευτής ιστορίας χειρίζεται επίσης τα χρονικά χάσματα, που σημαίνουν ότι για τους χρόνους μεταξύ δύο εκδόσεων, ο προμηθευτής ιστορίας επιστρέφει μια έγκυρη κατάσταση.

9.4 Πρότυπα για Ασφαλιστικά Συστήματα (Patterns for Insurance Systems)

Με την ύπαρξη περισσότερου ανταγωνισμού στην ασφαλιστική αγορά οι φορείς παροχής υπηρεσιών προσπαθούν να επεκταθούν στους τομείς νέων προϊόντων. Προσπαθούν να προσφέρουν τα μεμονωμένα προϊόντα προκειμένου να εξυπηρετηθούν οι αγορές θέσεων (niche markets) και προκειμένου να εξυπηρετηθούν οι υπάρχοντες πελάτες τους καλύτερα. Συγχρόνως η ανάγκη να μειωθούν οι δαπάνες απαιτεί εύκαμπτα συστήματα που επιτρέπουν τους ανωτέρω στόχους με αποδεκτό κόστος συντήρησης.
Τα συστήματα που χρησιμοποιούνται στην οικονομική βιομηχανία για να επιτύχουν αυτό το σύνολο στόχων παρουσιάζουν μεγάλες ομοιότητες. Η αποκαλούμενη προσανατολισμένη προς το προϊόν προσέγγιση μπορεί να βρεθεί στις τραπεζικές εργασίες καθώς επίσης και στη ασφαλιστική βιομηχανία. Αυτή η προσέγγιση δανείζεται από τις παραδοσιακές βιομηχανίες κατασκευής [17].
Θα αναλυθούν διάφορα πρότυπα για εύκαμπτα, προσανατολισμένα προς το προϊόν ασφαλιστικά συστήματα που μπορούν να βρεθούν σε πολλά συστήματα στην

ασφάλεια και την οικονομική βιομηχανία. Είναι πολύ πιθανό ότι πολλά από τα πρότυπα ισχύουν όμοια σε άλλες βιομηχανίες, αλλά όπως τα πρότυπα εδώ εφαρμόστηκαν στα ασφαλιστικά προγράμματα δεν θα γίνει και το ανάλογο για τις άλλες περιοχές.

Το σύνολο των απαιτήσεων που οδηγεί τα πρότυπα που παρουσιάζονται είναι συνήθως ίδιο. Αυτές οι απαιτήσεις είναι [28]:

Χρόνος στην αγορά και ευελιξία (Time to Market and Flexibility): Στην Ευρώπη η ασφαλιστική αγορά έχει ρυθμιστεί από τα κυβερνητικά πρακτορεία. Με την εμφάνιση της άρσης των ελέγχων των παγκόσμιων αγορών και της ευρωπαϊκής αγοράς, οι ασφαλιστικές εταιρείες βρίσκονται σε μια νέα διάσταση του ανταγωνισμού. Οι νέοι ανταγωνιστές περιλαμβάνουν τις τράπεζες και άλλους παρόχους οικονομικών υπηρεσιών, ασφαλιστικών εταιρειών από τις γειτονικές χώρες και πολλοί άλλοι. Στον άγριο ανταγωνισμό ο Χρόνος στην αγορά διαδραματίζει έναν κυρίαρχο ρόλο. Οι επιχειρήσεις που είναι σε θέση να σχεδιάσουν και να εμπορευτούν περισσότερα προϊόντα γρηγορότερα είναι επίσης και ικανές να προσφέρουν μεμονωμένα προϊόντα και έχουν ένα ανταγωνιστικό πλεονέκτημα από τις επιχειρήσεις που χρειάζονται έτη για να φέρουν ένα νέο προϊόν στην αγορά.

Νέα και μεμονωμένα προϊόντα: Ο Michael Porter διακρίνει τις διάφορες ανταγωνιστικές στρατηγικές [18]. Μπορεί κάποιος να γίνει ηγέτης δαπανών, ποιοτικός ηγέτης ή φορέας θέσεων (niche player). Εάν θέλει να ανοίξει τμήματα καινούργιων αγορών, είναι χρήσιμο να είναι σε θέση να έχει πολλά μεμονωμένα προϊόντα για πολλές ομάδες πελατών. Σε περιόδους της πρόωρης εκβιομηχάνισης αυτός ήταν ένας στόχος αντίθετος με το στόχο της αποτελεσματικότητας δαπανών. Με την εμφάνιση των υπολογιστών, μεμονωμένα προϊόντα με χαμηλότερο κόστος γίνονται εφικτά. Πρέπει να συνδυαστούν οι στοιχειώδεις τυποποιημένες δομικές μονάδες για να ληφθούν μεμονωμένα προϊόντα. Οι επιχειρήσεις που κατορθώνουν να παρέχουν μεμονωμένα προϊόντα έχουν ένα σοβαρό ανταγωνιστικό πλεονέκτημα.

Σημείο της Υπηρεσίας και της Κοπής Δαπανών: Υπάρχουν διάφοροι τρόποι να αντιδράσουν σε περισσότερο ανταγωνισμό. Ένας τρόπος είναι να παρασχεθεί η υπηρεσία πιο κοντά στον πελάτη. Πολλά ασφαλιστικά συστήματα είναι σήμερα συστήματα γραφείων με πολύ μεγάλη ροή εγγράφων που ρέει μεταξύ διάφορων στρωμάτων των διαφόρων γραφείων. Οι στρατηγικές μείωσης δαπανών στοχεύουν σε αυτήν την ροή εγγράφων και στοχεύουν επίσης στη μείωση των βημάτων διαδικασιών. Τα ασφαλιστικά συστήματα επεκτείνονται πιο κοντά στον πελάτη, μέσω του Διαδικτύου και με τα lap-top των αντιπροσώπων πωλήσεων γίνεται πολύ πιο εύκολα η ενσωμάτωση στο κεντρικό σύστημα συναλλαγής.

Κόστος Ανάπτυξης: Οι ασφαλιστικές εργασίες παρέχουν καθαρές άυλες υπηρεσίες - κανένα υλικό (hardware) – που κανένα προϊόν δεν γίνεται να αγγιχτεί. Μια ασφαλιστική εταιρεία που μπορεί να κόψει δαπάνες δεδομένων επεξεργασίας είναι σαν μια κατασκευαστική επιχείρηση που μειώνει την εργασία και τις υλικές δαπάνες της. Το αποτέλεσμα είναι ένα ανταγωνιστικό πλεονέκτημα. Επομένως οι συνολικές σχετικές δαπάνες ανάπτυξης για τα ασφαλιστικά συστήματα πρέπει να μειωθούν παρέχοντας περισσότερη λειτουργία και ευελιξία συγχρόνως.

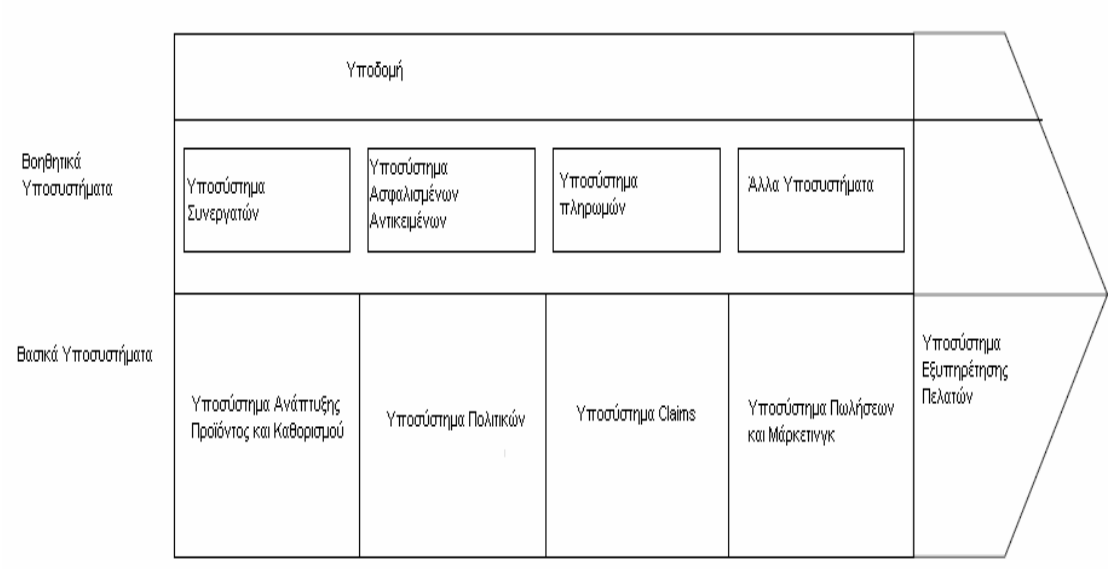
Τα πρότυπα που υπάρχουν σε αυτόν τον τομέα συνδέονται πολλές φορές μεταξύ τους και χωρίζονται σε ομάδες για να γίνουν πιο εύκολα κατανοητά [28].

1). Υψηλού Επιπέδου Δομή ενός Ασφαλιστικού Συστήματος: Εξετάζει τα υποσυστήματα υψηλού επιπέδου που υπάρχουν στα περισσότερα σύγχρονα ασφαλιστικά συστήματα. Τα πρότυπα είναι: Ασφαλιστική Αλυσίδα Τιμών (Insurance Value Chain), Κεντρικός Υπολογιστής Προϊόντων (Product Server) και Σύστημα

Κανόνα (Rule System) ως χαρακτηριστικό υποσύστημα ενός Κεντρικού Υπολογιστή Προϊόντων.
2). Πρότυπα Ασφαλιστικών Προϊόντων: Περιέχει μερικά πρότυπα που είναι χρήσιμα εάν χρειάζεται να διαμορφωθούν τα ασφαλιστικά προϊόντα για τη χρήση σε έναν Κεντρικό Υπολογιστή Προϊόντων. Τα πρότυπα που παρουσιάζονται είναι: Δέντρο Προϊόντος (Product Tree) και Αποζημίωση Γεγονότος Αντικειμένου (Object Event Indemnity).
3). Πολιτικές: Εδώ εξετάζεται πώς εφαρμόζονται οι πολιτικές ασφάλισης. Το μόνο πρότυπο είναι μέχρι τώρα η Πολιτική ως Στιγμιότυπο Προϊόντων.
4). Διανεμημένα Ασφαλιστικά Συστήματα: Μερικά υποσυστήματα που υπάρχουν σε ένα ασφαλιστικό σύστημα δεν θα ήταν απαραίτητα, εάν υπήρχε ένας εικονικός υπερυπολογιστής με μηδενικό χρόνο για την πρόσβαση δεδομένων και απεριόριστο εύρος ζώνης δικτύων ευρείας περιοχής με κανένα κόστος. Δυστυχώς τέτοια συστήματα ακόμα αναπτύσσονται. Μέχρι τότε θα εξεταστούν τα πρότυπα που εξετάζουν τη διανομή όπως το Σύστημα Ασφαλιστικών Πωλήσεων (Insurance Sales System) ή ένα Επιτραπέζιο Σύστημα (Table System).

9.4.1 Ασφαλιστική Αλυσίδα Τιμών (Insurance Value Chain)

Έστω ότι πρέπει να φτιαχτεί μια δομή για ένα νέο σύστημα γραφείων ασφάλειας. Διαπιστώνεται ότι υπάρχει πολλή περιττή λειτουργία στα υπάρχοντα συστήματα. Επίσης διαπιστώνεται ότι η προσθήκη των νέων προϊόντων είναι ένας κουραστικός στόχος που εκτελείται σε ένα ποσοστό δύο νέων προϊόντων το χρόνο και την κατηγορία προϊόντων. Αυτό προέρχεται από το γεγονός ότι η γνώση προϊόντων είναι πέρα από το σύστημα που διαχειρίζεται τις πολιτικές και το μέρος συστήματος που διαχειρίζεται την επεξεργασία .
Ποια είναι μια καλή αρχιτεκτονική για τη λειτουργία της επιχείρησης και των επιχειρησιακών αντικειμένων που εμφανίζεται σε ένα σύστημα ασφάλειας γραφείων;
Το σύστημα πρέπει να κατασκευαστεί με τέτοιο τρόπο που να είναι ικανό να υποστηρίξει την ανταγωνιστική στρατηγική. Πρέπει να υπάρχει σωστή διαμόρφωση, άρα καλή συντηρησιμότητα και διαχείριση της πολυπλοκότητας των μεγάλων ασφαλιστικών συστημάτων, που έχουν ένα χαρακτηριστικό μέγεθος πολύ περισσότερων από 10.000 σημείων λειτουργίας.
Η αρχιτεκτονική κτίζεται σύμφωνα με την Ασφαλιστική Αλυσίδα Τιμών [28].



Σχήμα 9.4.1.1 Απεικονίζεται η δομή ενός Συστήματος Ασφάλειας σύμφωνα με την Ασφαλιστική Αλυσίδα Τιμών.
Η αλυσίδα τιμών (value chain) διαμορφώνεται από τα υποσυστήματα [28]:
1). Υποσύστημα Ανάπτυξης Προϊόντος και Καθορισμού (Product Development and Definition Subsystem): Είναι αρμόδιο για την παροχή των άλλων μερών συστήματος με ορισμούς προϊόντων που ερμηνεύονται από τα άλλα μέρη του συστήματος. Πριν μπορέσουν να χρησιμοποιηθούν από άλλα υποσυστήματα, πρέπει να αναπτυχθούν και να καθοριστούν.
2). Υποσύστημα Πολιτικών (Policies Subsystem): Είναι αρμόδιο για να αποθηκεύσει τις πολιτικές και να υποστηρίξει όλες τις περιπτώσεις χρήσης που εκτελούν επιχειρησιακές ενέργειες σε αυτές.
3). Υποσύστημα Claims: Συγκεντρώνει απαιτήσεις επεξεργασίας για όλες τις κατηγορίες προϊόντων. Οι κύριοι επιχειρησιακοί στόχοι είναι γεγονότα, αξιώσεις (claims), περιληφθέντα συμβαλλόμενα μέρη και άλλα.
4). Υποσύστημα Πωλήσεων και Μάρκετινγκ (Sales and Marketing Subsystem): Εξετάζει πώς να πουλήσει τα προϊόντα στον πελάτη.
5). Υποσύστημα Εξυπηρέτησης Πελατών (Customer Service Subsystem): Έχει να κάνει με πρόσθετες υπηρεσίες που μπορεί να προσφερθούν στους πελάτες, εκτός από αυτό που είναι υποχρεωμένο να υποστηρίξει το σύστημα Claims.
Για να διαμορφωθεί κατάλληλα όλη την επιχειρησιακή λειτουργία χρειάζονται μερικά βοηθητικά υποσυστήματα [28]:
1). Υποσύστημα Συνεργατών (Partner (Party) Subsystem): Ενθυλακώνει τη γνώση για όλα τα συμβαλλόμενα μέρη, με τα οποία έχει να κάνει η επιχείρηση.
2). Υποσύστημα Ασφαλισμένων Αντικειμένων (Insured Objects Subsystem): Περιέχει τις πληροφορίες για ασφαλισμένα ή χαλασμένα αντικείμενα.
3). Υποσύστημα πληρωμών (Payments Subsystem): Χειρίζεται εισερχόμενες και εξερχόμενες ταμειακές ροές.
Υπάρχουν και άλλα υποσυστήματα που δεν είναι συγκεκριμένα για τις ασφαλιστικές εργασίες όπως η Γενική Λογιστική, τα Συστήματα Διοικητικών Πληροφοριών και άλλα τα οποία δεν εξετάζονται εδώ.
Η χρησιμοποίηση του προτύπου Ασφαλιστική Τιμή Αλυσίδας δεν υπονοεί ότι θα υπάρχει μια σίγουρη κατάληξη σε ένα εύκαμπτο σύστημα και ένα σύντομο χρόνο στην αγορά. Πρέπει να χρησιμοποιηθούν περισσότερα πρότυπα, για να υπάρξει ευελιξία προϊόντων.
Το πρότυπο επιτρέπει να χτιστεί ένα σύστημα για όλες τις κατηγορίες προϊόντων αλλά συνήθως χρησιμοποιούνται αρχικά μερικές κατηγορίες προϊόντων και μετά επεκτείνεται το σύστημα. Έτσι κόβονται οι δαπάνες ανάπτυξης με την αποφυγή της περιττής λειτουργίας.
Αυτή η δομή δεν υπονοεί ότι μικραίνουν τα στρώματα επικοινωνίας και κόβονται οι δαπάνες. Για να γίνει αυτό πρέπει να αρχίσει μια reengineering προσπάθεια επιχειρησιακής διαδικασίας και να χρησιμοποιηθεί ένα σύστημα (workflow) ροής της δουλειάς.

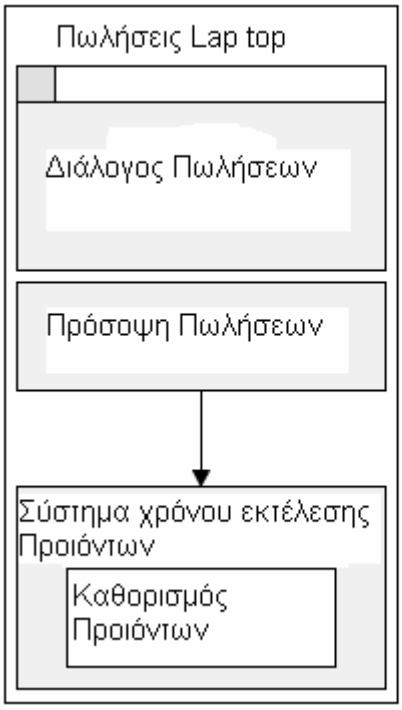
9.4.2 Κεντρικός Υπολογιστής Προϊόντων (Product Server)

Έστω ότι πρέπει να χτιστεί ένα προσανατολισμένο προς το προϊόν ασφαλιστικό σύστημα. Υπάρχει ένα σύστημα που επεξεργάζεται τις προτάσεις, διαχειρίζεται τις πολιτικές και χειρίζεται αξιώσεις (claims). Είναι απαραίτητοι οι ίδιοι ορισμοί προϊόντων για τα PC (προσωπικοί υπολογιστές) πωλήσεων και ίσως για άλλα

συστήματα όπως τις προσφορές Διαδικτύου ή τα συστήματα που υποστηρίζουν τους ασφαλιστικούς μεσίτες (insurance brokers) [28].

Πού καθορίζεται η γνώση προϊόντων και πώς διανέμεται;.
Εκτός από αυτά που αναφέρθηκαν, υπάρχουν και άλλα στοιχεία που λαμβάνονται υπόψη εδώ:
Ανεξαρτησία πλατφορμών (Platform indipendence): Συστήματα γραφείων (Back office systems) που οργανώνονται σε μηχανές OS/390. Τα PC πωλήσεων τρέχουν σε Win95 ή άλλα λειτουργικά συστήματα PC. Οι πελάτες Διαδικτύου μπορούν να τρέξουν σε Java ή σε HTML με έναν κεντρικό υπολογιστή σε κάποια γλώσσα. Οι ορισμοί προϊόντων πρέπει να είναι προσιτοί στις διαφορετικές πλατφόρμες.
Σχεδιασμός διεπαφών (Interface design): Οι ευρείες διεπαφές τείνουν να οδηγήσουν σε κακή συντηρησιμότητα και μονολιθικές δομές λογισμικού. Οι στενές διεπαφές είναι καλύτερες εάν είναι επιθυμητό ένα υποσύστημα για ένα μεγάλο αριθμό πελατών σε διαφορετικές πλατφόρμες [28].
Ενθυλάκωση της γνώσης προϊόντων εναντίον του βέλτιστου σχεδίου ενδιάμεσων με τον χρήστη: Αφ' ενός πρέπει να ενθυλακωθεί όλη γνώση προϊόντων. Το πλεονέκτημα αυτού θα ήταν ότι παραδείγματος χάριν όλοι οι διάλογοι ερμηνεύουν μόνο έναν καθορισμό προϊόντων και είναι σε θέση να προσαρμοστούν αυτόματα στους ορισμούς νέων προϊόντων. Το μειονέκτημα αυτής της προσέγγισης είναι, ότι το «αυτόματο» ενδιάμεσο με τον χρήστη είναι σπάνια όμορφο. Η προσέγγιση μπορεί να λειτουργήσει για τους διαλόγους γραφείων. Είναι πολύ απίθανο αυτό να δουλέψει για τις παρουσιάσεις προϊόντων διαλόγων .
Διαφορετικές Απαιτήσεις (Different Requirements): Ακόμα κι αν τα συστήματα γραφείων καθώς όπως και τα συστήματα πωλήσεων πρέπει να χρησιμοποιήσουν τους ίδιους ορισμούς προϊόντων, οι απόψεις τους σχετικά με τους ορισμούς προϊόντων μπορούν ελαφρώς να διαφέρουν. Για το σύστημα γραφείων που αποθηκεύει όλες τις πολιτικές χρειάζεται παλαιούς καθώς επίσης και πραγματικούς ορισμούς προϊόντων. Για το σύστημα πωλήσεων χρειάζεται μόνο το πρόγραμμα προϊόντων που πωλείται πραγματικά. Τα συστήματα καθορισμού προϊόντων ξέρουν περισσότερα προϊόντα. Εκείνοι που αναπτύσσονται ακόμα και δεν είναι στο πρόγραμμα πώλησης ακόμα. Πρέπει να παρέχονται διαφορετικές απόψεις για αυτά τα συστήματα.
Η γνώση προϊόντων πρέπει να ενθυλακωθεί σε έναν κεντρικό υπολογιστή προϊόντων. Στο χρόνο εκτέλεσης ενθυλακώνεται σε ένα φορητό σύστημα χρόνου εκτέλεσης προϊόντων και παρέχει απόψεις χρησιμοποιώντας προσόψεις (facades).
Σε έναν χαρακτηριστικό κεντρικό υπολογιστή προϊόντων υπάρχουν τα ακόλουθα συστατικά [28]:
1). Συντάκτης προϊόντων (Product editor): Παρέχει μια εύχρηστη διεπαφή για το πρόσωπο που καθορίζει τα προϊόντα. Οι ορισμοί προϊόντων μπορούν να συνεχιστούν σε ένα μοντέλο προϊόντων.
2). Επίμονοι ορισμοί προϊόντων (Persistent Product Definitions): Είναι ένα σύνολο επίμονων προϊόντων επιχείρησης που αναπαριστούν τα προϊόντα.
Αυτά τα πρώτα δύο συστατικά μπορούν να οργανωθούν σε μια συνηθισμένη αντικειμενοστρεφή αρχιτεκτονική τριών στρωμάτων.
3). Γεννήτρια (Generator): Αρμόδια για να παράγει κάποιο φορητού κώδικα από bytes, από τους επίμονους ορισμούς προϊόντων.
4). Σύστημα χρόνου εκτέλεσης Προϊόντων (Product Runtime System): Ερμηνεύει αυτόν τον κώδικα και είναι σε θέση να τρέξει στις πλατφόρμες OS/390 καθώς επίσης και Windows. Μερικά συστήματα χρησιμοποιούν το ANSI C για το σύστημα χρόνου εκτέλεσης προϊόντων για να εξασφαλίσουν αυτήν την φορητότητα.

Αφού πρέπει να παρέχεται μια πολύ στενή διεπαφή στο σύστημα χρόνου εκτέλεσης προϊόντων και δεδομένου ότι γίνεται να εξεταστούν οι αρχικές διαδικασίες μιας τέτοιας στενής διεπαφής, μπορεί να εφαρμοστεί:
5). Προσόψεις (Facades): Που παρέχουν τις διαφορετικές απόψεις σχετικά με τους ορισμούς προϊόντων.
Το γεγονός ότι ο κεντρικός υπολογιστής προϊόντων που καλείται περιέχει όλα τμήματα των συστατικών, δεν δείχνει ότι όλα αυτά τα συστατικά θα τρέξουν απαραίτητα σε μια ενιαία μηχανή ή ότι ο κεντρικός υπολογιστής προϊόντων είναι κεντρικός υπολογιστής από την άποψη ενός συστήματος πελατών/κεντρικών υπολογιστών [28].
Η διαμόρφωση για το σύστημα πωλήσεων φαίνεται στο παρακάτω σχήμα. Ο διάλογος πωλήσεων χρησιμοποιεί το σύστημα χρόνου εκτέλεσης προϊόντων μέσω της πρόσοψης. Το υπόλοιπο του αποκαλούμενου κεντρικού υπολογιστή προϊόντων, ο συντάκτης προϊόντων, τα επιχειρησιακά αντικείμενα προϊόντων, η γεννήτρια μπορούν να τρέξουν επάνω σε διαφορετικές μηχανές στο πλαίσιο των διαφορετικών λειτουργικών συστημάτων και των τεχνικών υποδομών [28].



Σχήμα 9.4.2.1 Απεικονίζονται οι πωλήσεις Lap top.

Ένα σχέδιο ασφαλιστικών συστημάτων που χρησιμοποιεί έναν κεντρικό υπολογιστή προϊόντων παρέχει την ευελιξία προϊόντων και την ανεξαρτησία πλατφορμών. Συγκρινόμενο με τις προσεγγίσεις που χρησιμοποιούν ένα ενεργό πρότυπο αντικειμένου (active object) μόνο, μπορούν να χρησιμοποιηθούν οι ίδιοι ορισμοί προϊόντων με διαφορετικές απόψεις, για να οδηγηθεί το σύστημα πωλήσεων και τις παρουσιάσεις Διαδικτύου.
Το μειονέκτημα είναι η διαδικασία παραγωγής. Τα συστήματα που χρησιμοποιούν μια γεννήτρια έχουν μερικά μειονεκτήματα, έναντι των συστημάτων που χρησιμοποιούν τα ενεργά πρότυπα και την αντανάκλαση (reflection) [19]. Τα μειονεκτήματα αντισταθμίζονται εν μέρει από το γεγονός ότι συνήθως δεν είναι επιθυμητό οι αλλαγές στο πρότυπο προϊόντων να είναι ορατές παντού, την ίδια στιγμή που εισάγονται σε έναν συντάκτη προϊόντων [28].

Ασφαλιστικά προϊόντα, όπως κώδικας πρέπει να εξεταστεί και versioned . Ένα σύστημα που χρησιμοποιεί ένα ενιαίο ενεργό πρότυπο αντικειμένου και όχι χωριστά διαστήματα για τη ανάπτυξη προϊόντος, η δοκιμή προϊόντων και το παραγωγικό προϊόν μπορούν εύκολα να φέρουν πρόβλημα.
Άλλες συνέπειες είναι [28]:
1). Ανεξαρτησία πλατφορμών (Platform independence): Επιτυγχάνεται με τη χρησιμοποίηση ενός φορητού συστήματος χρόνου εκτέλεσης προϊόντων.
2). Καλός σχεδιασμός διεπαφών (Good Interface design): Μπορεί να επιτευχθεί με τον καθορισμό μιας πολύ στενής διεπαφής για το σύστημα χρόνου εκτέλεσης προϊόντων.
3). Το βέλτιστο σχέδιο ενδιάμεσων με τον χρήστη (The Optimal User Interface Design): Περιέχει κάποιο προϊόν που γνωρίζει τον εαυτό του.
4). Διαφορετικές απαιτήσεις (Different Requirements): Μπορεί να ενσωματωθούν χρησιμοποιώντας διαφορετικές προσόψεις στο σύστημα χρόνου εκτέλεσης προϊόντων.

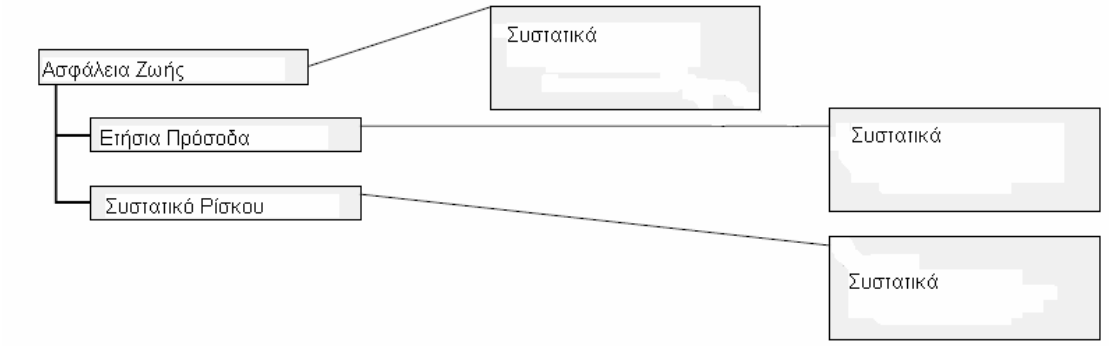
9.4.3 Σύστημα Κανόνα (Rule System)

Μόλις χτιστεί ένας συντάκτης προϊόντων, που επιτρέπει να καθοριστεί η δομή προϊόντος, χρειάζεται ένας μηχανισμός για να αντλήσει τις ιδιότητες των στοιχείων δέντρων από άλλες ιδιότητες προκειμένου να εκτιμηθεί παραδείγματος χάριν μια πολιτική [28].
Πώς κωδικοποιείται η λειτουργία που πρέπει να εκτελέσει τους ελέγχους αληθοφάνειας (plausibility) ή την πολιτική εκτίμησης στα δέντρα προϊόντων;
Πρέπει να ληφθούν υπόψη [28]:
Δυνατότητα χρησιμοποίησης από τους χειριστές εφαρμογής: Εάν πρέπει οι διοικητές εφαρμογής να είναι σε θέση να καθορίσουν νέα προϊόντα, χωρίς κωδικοποίηση, πρέπει να καθοριστεί μια γλώσσα (scripting language) που επιτρέπει τη σύνδεση των ιδιοτήτων των προϊόντων, που εκτελεί τους υπολογισμούς. Έτσι, προκύπτει εύκολα μια πλήρης νέα γλώσσα προγραμματισμού.
Κόστος (Cost): Η ανάπτυξη ενός συστήματος κανόνα (rule system) γίνεται εύκολα μια ακριβή περιπέτεια. Μπορεί συχνά να είναι φτηνότερο να εκπαιδεύσει ορισμένους διοικητές εφαρμογής αντί να αναπτύξει μια γλώσσα προγραμματισμού για αυτούς.
Πληρότητα του μηχανισμού παραγωγής (Completeness of derivation mechanism): Εάν δημιουργείται ένα σύστημα κανόνα, πρέπει να εκτελέσει όλες τις παραγωγές ιδιοτήτων που είναι αναγκαίες για τον καθορισμό προϊόντων. Διαφορετικά πρέπει οπωσδήποτε να γίνει προγραμματισμός.
Ευελιξία (Flexibility): Για τις γλώσσες με σύνταξη όπως C++, το Excel Sheet, είναι πιο εύκαμπτο για τους διοικητές εφαρμογής από το να γίνει έκδοση, σύνταξη και σύνδεση του κύκλου.
Προϊόντα δοκιμής και διόρθωσης (Testing and Debugging Products): Εάν χρησιμοποιείται ένας μηχανισμός παραγωγής ιδιοτήτων κώδικα σε ορισμούς προϊόντων που συμπεριφέρεται όπως μια γλώσσα σεναριογραφιών (script language), θα χρειαστούν πρόσθετα εργαλεία όπως σε οποιοδήποτε γλωσσικό περιβάλλον.
Πρέπει να χτιστεί ένας μηχανισμός παραγωγής ιδιοτήτων παρόμοιος με τις σημασιολογικές λειτουργίες ενός parse δέντρου [20] ή με τους υπολογισμούς κελιών σε ένα φύλλο Excel.
Όπως φαίνεται στο σχήμα 9.4.4.1 στο Ασφαλιστικό Προϊόν ως Δέντρο (Insurance Product as Tree): Όπως στο χτίσιμο μεταγλωττιστή, μπορούν να προγραμματιστούν οι σημασιολογικές λειτουργίες σε μια γλώσσα προγραμματισμού ή μπορεί επίσης να

καθοριστεί ένας κλειστό σύνολο εργαλείων με μια χωριστή γλώσσα σεναριογραφιών [28].
Αυτό δεν έχει να κάνει με Βασισμένα στον Κανόνα Συστήματα (Rule Base Systems). Εκεί βρίσκονται συχνά τα έμπειρα συστήματα (που εφαρμόζονται παραδείγματος χάριν σε Prolog). Το σύστημα κανόνα (Rule System) εφαρμόζει εδώ τον επιχειρησιακό κανόνα, αλλά σε μια αρκετά αιτιοκρατική και προβλέψιμη βάση.
Οι συνέπειες είναι τόσο πολλές όσο οι παραλλαγές εφαρμογής. Τα ζητήματα που περιλαμβάνονται είναι απολύτως ανάλογα με το σχέδιο γλώσσας προγραμματισμού. Εδώ είναι μερικά παραδείγματα που μπορούν να συμβούν και πραγματικά συνέβησαν στα συστήματα που είναι ήδη γνωστά.
Ελλιπής γλώσσα καθορισμού κανόνα (Incomplete rule definition language): Σε αυτήν την περίπτωση το σύστημα είναι πάρα πολύ ακριβό έναντι της ευελιξίας και της προσαρμοστικότητας που προσφέρει [28].
Υπερπλήρης Γλώσσα καθορισμού κανόνα (Overcomplete rule definition language): Μια τέτοια γλώσσα είναι σύνθετη και μπορεί να είναι επιρρεπής σε λάθη και να είναι ακριβή. Σε αυτήν την περίπτωση μπορεί να χρησιμοποιηθεί μια ερμηνευμένη γλώσσα προγραμματισμού (interpreted programming language) αντί ενός αποκαλούμενου συστήματος κανόνα.

9.4.4 Δέντρο Προϊόντος (Product Tree)

Μετά την προσέγγιση Ασφαλιστική Αλυσίδα Τιμών έχει αποφασιστεί να χτιστεί ένας Κεντρικός Υπολογιστής Προϊόντων. Πριν σχεδιαστεί ο Κεντρικός Υπολογιστής Προϊόντων, πρέπει να διαμορφωθούν τα Ασφαλιστικά Προϊόντα [28].
Ποια είναι μια καλή αναπαράσταση για τα Ασφαλιστικά Προϊόντα;
Πρέπει να ληφθούν υπόψη :
Συμμετοχή χρηστών εναντίον της ευελιξίας και της γενικότητας: Πρέπει να υπάρχει ένα πρότυπο προϊόντων, το οποίο οι χρήστες θα καταλάβουν. Όμως η βιομηχανία έχει δει τις πολύ εύκαμπτες προσεγγίσεις βασισμένες στα αφηρημένα πρότυπα δεδομένων που είναι πολύ δύσκολο να γίνουν κατανοητά.
Επαναχρησιμοποίηση και ευελιξία: Μια καλή προσέγγιση θα επέτρεπε την επαναχρησιμοποίηση σε επίπεδο δομικών μονάδων. Αυτό που θα ήταν επιθυμητό είναι βιβλιοθήκες των δομικών μονάδων προϊόντων που μπορούν να επαναχρησιμοποιηθούν και να επανασυνδυαστούν για να διαμορφώσουν τα νέα προϊόντα όπως στη βιομηχανία κατασκευής.
Η λύση και δομή του προτύπου φαίνεται στο σχήμα 9.4.4.1 [28].



Σχήμα 9.4.4.1 Απεικονίζεται το Προϊόν Ασφάλειας ως Δέντρο.

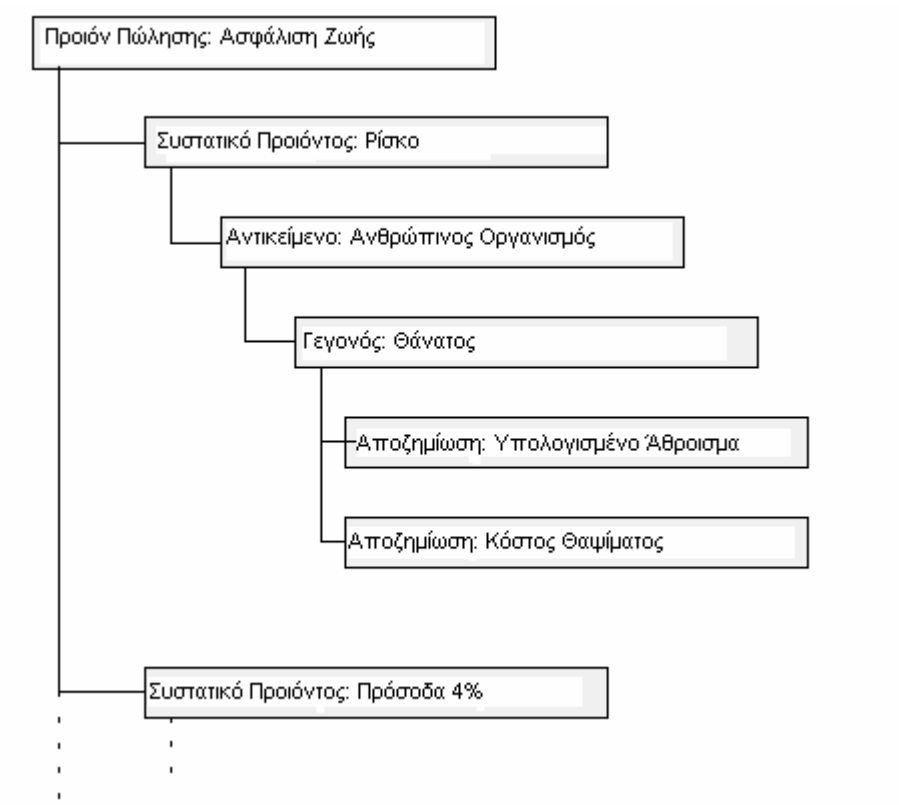
· Διαμορφώνονται τα προϊόντα και τα συστατικά τους ως κόμβοι ενός δέντρου.
· Προστίθενται οι ιδιότητες σε κάθε κόμβο περιγράφοντας τον.

· Προστίθεται η λειτουργία όπως ο υπολογισμός ενός ασφαλίστρου, ως κανόνες παραγωγής, ανάλογοι με τις σημασιολογικές λειτουργίες σε ένα parse δέντρο μεταγλωττιστών ή μπορεί να δημιουργηθεί ένα νέο σύστημα κανόνα.
Οι συνέπειες του προτύπου είναι [28]:
Ευελιξία προϊόντων: Μόλις προσδιοριστούν οι στοιχειώδεις δομικές μονάδες των ασφαλιστικών προϊόντων, ο σχεδιασμός και η διαμόρφωση των νέων προϊόντων γίνεται πολύ ευκολότερα και απλά.
Συμμετοχή χρηστών: Η πρακτική εργασία με τους κεντρικούς υπολογιστές προϊόντων δείχνει ότι οι εμπειρογνώμονες περιοχών, με κάποια πρόσθετη εκπαίδευση είναι σε θέση να διαμορφώσουν τα προϊόντα ως δέντρα.
Επαναχρησιμοποίηση: Μόλις καθοριστούν οι δομικές μονάδες μπορεί να επαναχρησιμοποιηθούν όλες στα νέα και υπάρχοντα προϊόντα.

9.4.5 Αποζημίωση Γεγονότος Αντικειμένου (Object Event Indemnity)

Έχει αποφασιστεί να διαμορφωθούν τα ασφαλιστικά προϊόντα ως δέντρα και αρχίζουν να καθορίζονται τα προϊόντα. Έστω ότι γίνεται προσπάθεια να ενωθούν δύο πρότυπα που χτίστηκαν από διαφορετικούς ανθρώπους, και το παράδειγμα δέντρων δεν επιτρέπει να επανασυνδυαστούν αυθαίρετα οι δομικές μονάδες [28].
Ποιος είναι ένας καλός τρόπος να διαμορφωθούν τα ασφαλιστικά προϊόντα, που χρησιμοποιούν μια δομή δέντρων; Ποιες είναι καλές αφαιρέσεις για να διαμορφώσουν τα ασφαλιστικά προϊόντα;
Ποιότητα και δυνατότητα κατανόησης: Όπως με πολλές γλώσσες προγραμματισμού, το μόνο παράδειγμα για να εφαρμόσει τα ασφαλιστικά προϊόντα ως δέντρο προϊόντων επιτρέπει πολλά πρότυπα που είναι κακοδομημένα, δεν είναι εύκολα κατανοητά και θα προκαλέσουν αργότερα έναν πονοκέφαλο συντήρησης. Οι παρόμοιες παρατηρήσεις ισχύουν για πολλές περιοχές με ένα πολύ υψηλού επιπέδου και αφηρημένο παράδειγμα διαμόρφωσης, όπως τη διαμόρφωση αντικειμένου και τη διαμόρφωση σχέσης οντοτήτων. Χρειάζονται μερικοί κανόνες ή πρότυπα ανάλυσης περιοχών [21], για να μειωθούν οι πιθανές λύσεις και για να διευκολυνθεί η διαμόρφωση .
Επαναχρησιμοποίηση: Όσο πιο ομοιόμορφα διαμορφώνονται τα τμήματα ασφαλιστικών προϊόντων, τόσο πιο εύκολα μπορούν να επαναχρησιμοποιηθούν σε άλλα προϊόντα.
Η λύση είναι μοντελοποίηση των ασφαλιστικών προϊόντων σε όρους OEI: Αντικείμενο - Γεγονός – Αποζημίωση (Object- Event- Indemnity). Κάθε ασφαλιστικό προϊόν ασφαλίζει ένα ή περισσότερα ασφαλισμένα αντικείμενα. Εάν ένα ορισμένο γεγονός (π.χ. ζημία) συμβαίνει σε εκείνο το αντικείμενο, ο πελάτης έχει το δικαίωμα να απαιτήσει μια ορισμένη αποζημίωση.

Τα δέντρα προϊόντων μοιάζουν με το σχήμα 9.4.5.1 [28].



Σχήμα 9.4.5.1 Απεικονίζεται το Δέντρο του Προϊόντος Ασφάλισης Ζωής.

Κάθε όρος μπορεί να χρησιμοποιηθεί κατ' επανάληψη: Τα αντικείμενα μπορούν να περιέχουν άλλα αντικείμενα, προϊόντα μπορεί να περιέχουν άλλα προϊόντα και ούτω καθεξής. Αλλά η βασική ιδέα πίσω από τη δομή είναι αρκετά σταθερή.
Οι συνέπειες είναι παραγωγικότητα και επαναχρησιμοποίηση: Εάν καθένας εμμένει σε αυτό ή ένα παρόμοιο πρότυπο διαμόρφωσης θα υπάρχει μια κοινή κατανόηση για τις δομές προϊόντων. Θα υπάρχει υψηλότερη παραγωγικότητα και καλύτερη επαναχρησιμοποίηση.

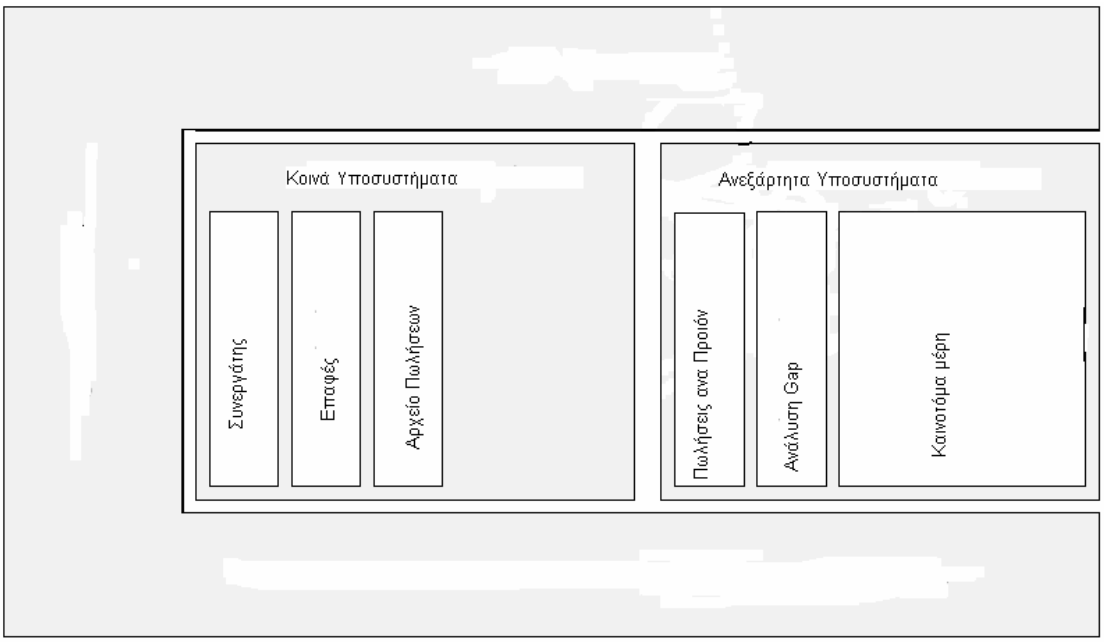
9.4.6 Πολιτική ως Στιγμιότυπο Προϊόντων (Policy as Product Instance)

Έστω ότι έχει δημιουργηθεί ο συντάκτης προϊόντων, χρησιμοποιώντας τη σύνθεση για το πρότυπο και κάποιο ιεραρχικό παράθυρο καταλόγων για το GUI (Graphic User Interface). Έχουν γραφεί επίσης οι κόμβοι δέντρων ως προϊόντα, ή τμήματα προϊόντων και η αποζημίωση γεγονότος αντικειμένου.
Πώς μπορεί να αναπαρασταθεί μια πολιτική ασφάλειας;
Η λέξη κλειδί είναι η ευελιξία. Ο καλύτερος κεντρικός υπολογιστής προϊόντων είναι πιθανό να είναι άχρηστος, εάν δεν μπορεί ο χρήστης να χειριστεί τα νέα προϊόντα στο σύστημα γραφείων. Επομένως το σύστημα που χειρίζεται τις πολιτικές, πρέπει να είναι τόσο εύκαμπτο όσο ένα εργαλείο καθορισμού προϊόντων.
Η λύση είναι δημιουργία στιγμιότυπων πολιτικών από προϊόντα. Χρησιμοποίηση του προτύπου αντικειμένου τύπου για να γίνει αυτό. Η πολιτική αντανακλά έπειτα τους δέντρο - δομημένους ορισμούς προϊόντων. Μερικά στιγμιότυπα προϊόντων διατηρούνται ως πρωτότυπα σε ένα εργοστάσιο πολιτικών [22].
Οι συνέπειες του προτύπου είναι [28]:

Ευελιξία: Όπως τα περισσότερα αντανακλαστικά συστήματα [19] θα δημιουργηθεί ένα πολύ εύκαμπτο σύστημα που μπορεί να διατηρηθεί συγχρονισμένο με τους ορισμούς προϊόντων.
Φορητότητα: Πρέπει να υπάρχουν πρόσθετες προφυλάξεις για να διατηρηθεί το σύστημα φορητό. Εάν χρησιμοποιηθεί το πρότυπο για να εφαρμοστεί ένα ενεργό πρότυπο αντικειμένου σε έναν host υπολογιστή, μπορεί να υπάρξει μια δυσκολία να εφαρμοστεί το ίδιο πράγμα στους φορητούς προσωπικούς υπολογιστές των πωλητών και αντίστροφα.
Απόδοση: Όπως με τα περισσότερα αντανακλαστικά συστήματα η απόδοση μπορεί να γίνει κακή εάν δεν παρακολουθείται συνεχώς. Για να αποτραπεί αυτό, πρέπει να δημιουργηθεί μια έξυπνη χαρτογράφηση βάσεων δεδομένων [23].

9.4.7 Σύστημα πωλήσεων ασφαλειών (Insurance Sales System)

Αφού υπάρχει ένα εύκαμπτο σύστημα γραφείων είναι επιθυμητό ένα σύστημα κεντρικών γραφείων lap-top για τους αντιπροσώπους πωλήσεων που βοηθά να πουληθούν τα προϊόντα που έχουν καθοριστεί χρησιμοποιώντας τον κεντρικό υπολογιστή προϊόντων [28].
Πώς θα δομηθεί το σύστημα πωλήσεων στα lap-top των αντιπροσώπων;
Διανομή συστήματος (System Distribution): Το ιδανικό ασφαλιστικό σύστημα θα ήταν ένας διανεμημένος εικονικός κεντρικός υπολογιστής με ένα GUI που παρέχει δύναμη στις πωλήσεις, όλες τις διαθέσιμες πληροφορίες για την επιχείρησή. Αυτό είναι απαγορευτικά ακριβό ακόμα δεδομένου ότι υπάρχουν διαφορές στην απόδοση μεταξύ της πρόσβασης μνήμης του αρχείου ή της βάσης δεδομένων I/O και το δικτυωμένο (networked) I/O.
Χαμηλότερο κόστος έναντι της μεμονωμένης εμφάνισης αγοράς: Σήμερα μπορεί να αγοραστούν τα συστήματα απομίμησης χαμηλότερου κόστους που πωλούν τα τυποποιημένα οικονομικά προϊόντα σε χαμηλές τιμές. Αυτό όμως δεν είναι το ζητούμενο, αλλά μια μεμονωμένη εμφάνιση αγοράς, τα μη τυποποιημένα προϊόντα και γρήγορα.
Ευελιξία: Σύντομη κλήση κύκλων προϊόντων για ένα πολύ εύκαμπτο σύστημα.
Επαναχρησιμοποίηση και ενιαία πηγή: Για να κρατηθούν οι δαπάνες περιληφθείσες και για να αποφευχθούν οι πηγές προβλημάτων, όπως ο πλεονασμός, πρέπει να χρησιμοποιηθεί μια ενιαία βάση κώδικα και μια μέγιστη επαναχρησιμοποίηση και για τα συστήματά κεντρικών γραφείων και γραφείων.
Καινοτόμες πτυχές: Χρειάζεται μια θέση για να συνδεθούν τα καινοτόμα μέρη στην εφαρμογή.
Καταρχήν ορίζονται εκείνα τα πράγματα στο lap-top που πρέπει να πουληθούν, τα προϊόντα ή η παρεχόμενη υπηρεσία. Κατόπιν διαιρείται η λειτουργία σε ένα μέρος, που δεν είναι συγκεκριμένο για οποιαδήποτε ασφαλιστική εταιρεία, ένα υποσύστημα συνεργατών , ένα αρχείο πωλήσεων, ένα υποσύστημα επαφών και το μέρος του πυρήνα πωλήσεων, που μετατρέπει τις προτάσεις σε πολιτικές. Προστίθεται ένα μέρος πωλήσεων πολυμέσων στην προτίμηση του χρήστη και γίνεται σύνδεση του συστήματος μέσω μιας μη απευθείας σύνδεση με τη μονάδα κεντρικής επεξεργασίας προκειμένου να υποβληθούν σε επεξεργασία οι προτάσεις και τα δεδομένα ανταλλαγής των πελατών [28].



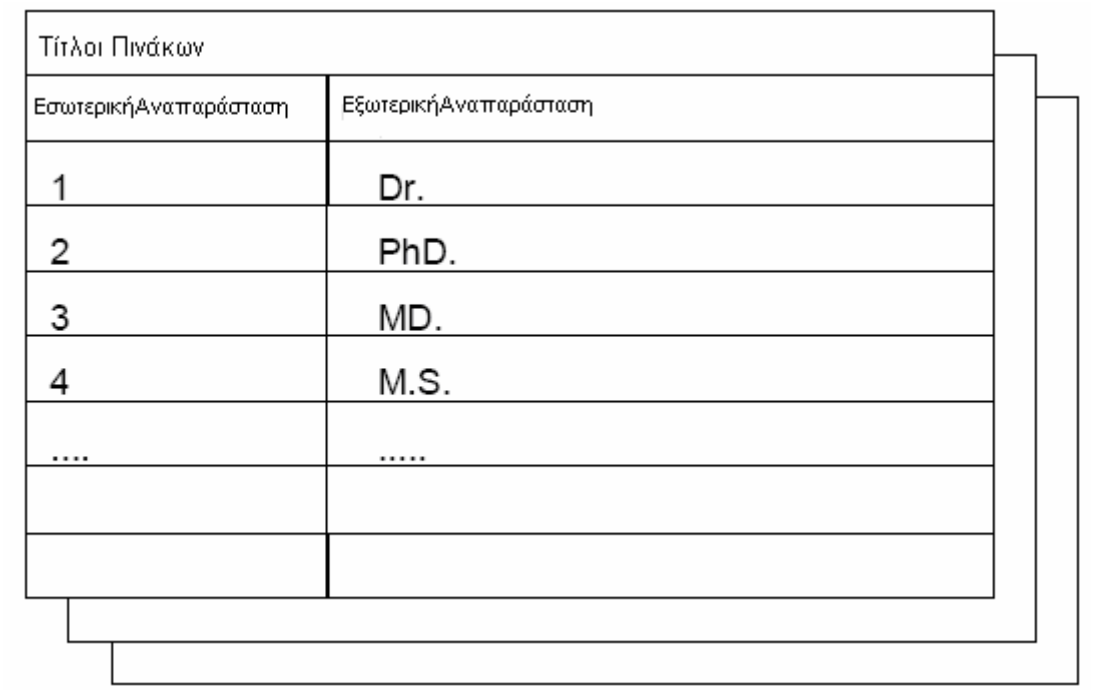
Σχήμα 9.4.7.1 Απεικονίζεται το Σύστημα Πωλήσεων Ασφαλειών

Τα συστατικά του συστήματός έχουν τις ακόλουθες ευθύνες [28]:
1). Κοινό Τμήμα (Common Shell): Παρέχει όλη την τεχνική υποδομή που είναι απαραίτητη για μια βασισμένη στο PC αντικειμενοστρεφή λύση, όπως τα MVC πλαίσια, πλαίσια εμμονής (persistence frameworks), και του γενικού οδηγού εφαρμογής .
2). Υποσύστημα συνεργατών (Partner Subsystem): Επιτρέπει στις πωλήσεις αντιπροσωπευτικά να συλλέξουν τις πληροφορίες για τους πελάτες της, τις οικονομικές περιστάσεις τους, τα χόμπι και άλλα. Όλα τα γεγονότα που απαλύνουν τις πωλήσεις και επιτρέπουν την επιλογή των ομάδων-στόχων για τη δράση πωλήσεων. Ένα υποσύστημα συνεργατών για ένα πρόσωπο πωλήσεων προσφέρει χαρακτηριστικά πολύ περισσότερη λειτουργία από ένα υποσύστημα συνεργατών γραφείων.
3). Αρχείο Πωλήσεων (Sales Record): Επιτρέπει να παρακολουθούνται τα προϊόντα, που ο συνεργάτης έχει αγοράσει ήδη από τους πωλητές.
4). Υποσύστημα επαφών (Contact Subsystem): Είναι ένα συστατικό που επιτρέπει να παρακολουθεί τα ραντεβού για να κάνει τον προγραμματισμό.
5). Πωλήσεις (Sales): Προσφέρει διαλόγους που επιτρέπουν να διατυπωθούν και να υπολογιστούν οι προτάσεις μαζί με τον πελάτη . Αυτοί οι διάλογοι είναι προϊόν προσανατολισμένο, χρησιμοποιώντας χαρακτηριστικά τον κεντρικό υπολογιστή προϊόντων .
6). Ανάλυση Gap (Gap Analysis): Είναι ένα μεμονωμένο μέρος που συγκεντρώνει τα στοιχεία για τον πελάτη και συγκρίνει τα οικονομικά προϊόντα που έχει, με τη συνολική ανάγκη οικονομικών προϊόντων, με συνέπεια έναν κατάλογο προϊόντων που μπορεί να αγοράσει.
7). Καινοτόμα μέρη (Innovative Parts) : Οτιδήποτε που βοηθά να πουληθούν τα προϊόντα, όπως πρόσθετες πληροφορίες προϊόντων πολυμέσων.
Οι συνέπειες είναι οι εξής:
Απόδοση στο δίκτυο: Με μια μη απευθείας σύνδεση σύστημα είστε ασφαλής σχετικά με την απόδοση, αλλά όχι και σε σχέση με την πρόσβαση σε όλα τα επιχειρηματικά

δεδομένα. Αυτό που χρειάζεται για να πουληθούν τα προϊόντα είναι όχι μόνο πωλήσεις, αλλά και καλή εξυπηρέτηση [28].
Λειτουργία: Το πρότυπο περιγράφει το είδος συστημάτων που υπάρχουν σήμερα όσο αφορά την υπηρεσία. Το πρότυπο δεν περιγράφει τα συστήματα που θα δούμε στο κοντινό μέλλον - θα δούμε πολύ περισσότερες λειτουργίες που κινούνται προς POS (Point Of Service).
Χαμηλότερο κόστος έναντι της μεμονωμένης εμφάνισης αγοράς: Το πρότυπο επιτρέπει μια μεμονωμένη εμφάνιση αγοράς (προϊόντα και καινοτόμα μέρη) με αναλογικά χαμηλότερο κόστος. Το πρότυπο αφήνει επίσης αρκετό χώρο για τις καινοτόμες πτυχές.

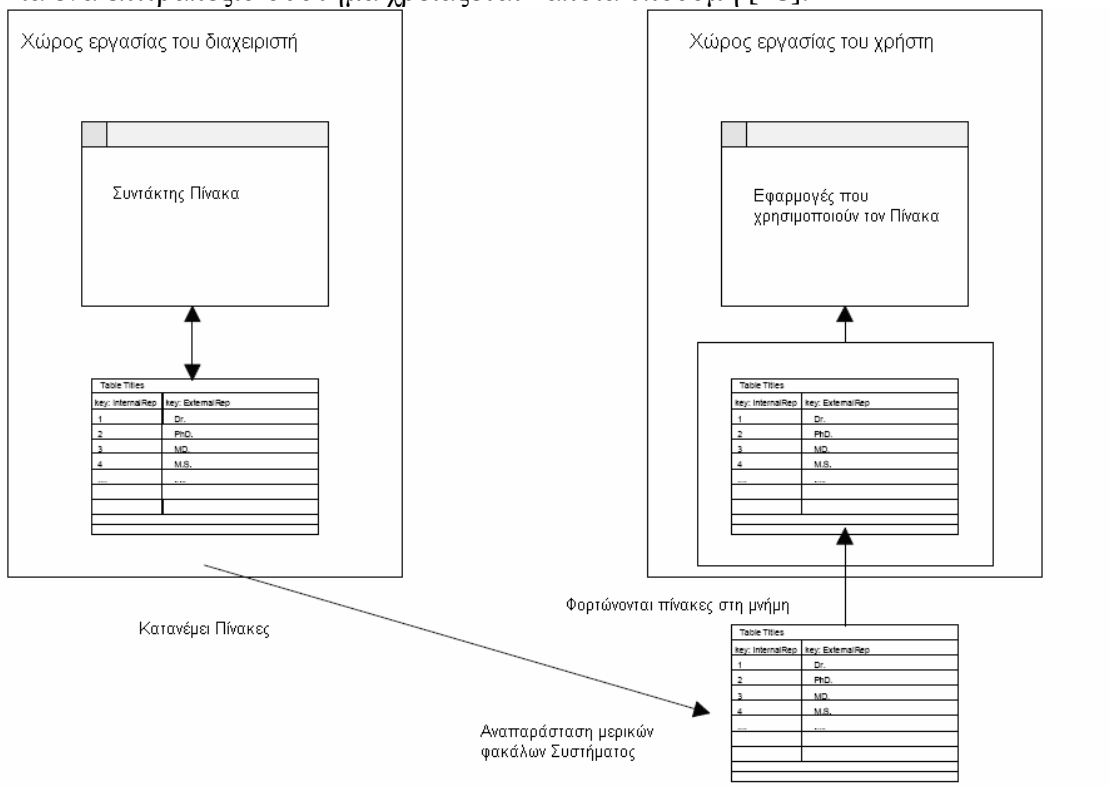
9.4.8 Επιτραπέζιο σύστημα (Table System)

Τα ασφαλιστικά συστήματα, ειδικά εκείνα που εξετάζουν προϊόντα, χρειάζονται πολλούς πίνακες, για να παρέχουν τις έγκυρες τιμές για τα κλειδιά, για να παρέχουν τις χαρτογραφήσεις από π.χ. τις περιοχές σε δεδομένα σχετικά για εκτίμηση αυτόματων πολιτικές ασφαλίστρων [28].
Πώς παρέχεται εύκαμπτη και έγκαιρη πρόσβαση στα δεδομένα που πρέπει να προσεγγιστούν, συνήθως μόνο για ανάγνωση, μπορεί να οργανωθεί με μορφή πίνακα και να πρέπει να ενημερώνεται τακτικά από τους ειδικούς περιοχών; [28]
Απόδοση έναντι της περιττής λειτουργίας: Εφαρμόζεται ένα περιττό σύστημα εάν εφαρμόζεται ένα δεύτερο σύστημα βάσεων δεδομένων με καλύτερη απόδοση από ένα σύστημα σχεσιακής βάσης δεδομένων, αλλά με τη λιγότερη λειτουργία. Μια δεύτερη τοπική βάση δεδομένων μόνο για ανάγνωση είναι τόσο γρήγορη, ώστε η τιμή που καταβάλλεται από την άποψη συντήρησης αξίζει την τιμή ενός διπλάσιου συστατικού.
Δαπάνες και προσπάθεια διανομής δεδομένων: Σε μια κεντρική βάση δεδομένων δεν υπάρχει κανένα replication πρόβλημα στοιχείων. Εάν υπάρχει μια δεύτερη βάση δεδομένων που χρησιμοποιεί τα αρχεία πίνακα σε παθητικό χρόνο και την κύρια αποθήκευση στο χρόνο λειτουργίας της εφαρμογής, τότε η εφαρμογή έχει πρόβλημα replication. Επομένως έχετε μια ανταλλαγή μεταξύ της διανομής δεδομένων και της απόδοσης πρόσβασης δεδομένων.
Δοκιμή και παράδοση των νέων προϊόντων: Τα δεδομένα καθορισμού συμπεριφέρονται σαν κώδικας. Τα δεδομένα του πίνακα σε μια ασφαλιστική εφαρμογή είναι δεδομένα καθορισμού και επομένως πρέπει να αντιμετωπιστούν σαν κώδικας. Χρειάζονται δοκιμή και απελευθέρωση των διαδικασιών. Εάν είναι επιθυμητό να παρέχεται αυτή η λειτουργία σε ένα κεντρικό σύστημα βάσεων δεδομένων, πρέπει να εφευρεθούν μερικές πρόσθετες διαδικασίες.
Ανάγκη για τα ιστορικά δεδομένα: Τα ασφαλιστικά συστήματα στηρίζονται σε μεγάλο ποσοστό στα ιστορικά δεδομένα. Για λόγους διαδικασιών εσωτερικού λογιστικού ελέγχου πρέπει να υπάρχει δυνατότητα αναπαραγωγής οποιασδήποτε παλαιάς κατάστασης μιας πολιτικής. Όμως οι πολιτικές μπορούν να αλλάξουν και πρέπει να είναι δυνατόν να αναπαραχθούν οι πραγματικές και ιστορικές καταστάσεις οποιασδήποτε σχετικής ασφαλιστικής σύμβασης οποιαδήποτε στιγμή.
Χρησιμοποιήθηκε ένα σύστημα πίνακα που παρέχει τους πίνακες ως μόνη αφαίρεση. Αυτοί οι πίνακες βρίσκονται στην κύρια μνήμη στο χρόνο εκτέλεσης. Σε έναν πελάτη ένα σύστημα πίνακα παρέχει ένα σύνολο πινάκων ως κεντρική αφαίρεση [28].



Σχήμα 9.4.8.1 Απεικονίζονται οι πίνακες που περιέχουν τίτλους για μια εφαρμογή ασφάλισης.

Για ένα επιτραπέζιο σύστημα χρειάζεται κάποια υποδομή [28].



Ένας συντάκτης πίνακα επιτρέπει στους διοικητές εφαρμογής να εισάγουν τα έγκυρα δεδομένα για τους πίνακες. Οι πίνακες πρέπει έπειτα να διανεμηθούν σε όλες τις περιοχές πελατών. Θα κατοικήσουν χαρακτηριστικά σε ένα ή περισσότερα επίπεδα αρχεία ή κάποια άλλη αναπαράσταση συστημάτων αρχείων. Όταν το σύστημα πελατών ξεκινά, μπορεί να φορτώσει όλους τους πίνακες στη μνήμη. Εάν το σύστημα πελατών δεν έχει αρκετή μνήμη μπορεί επίσης να χρησιμοποιήσει έναν

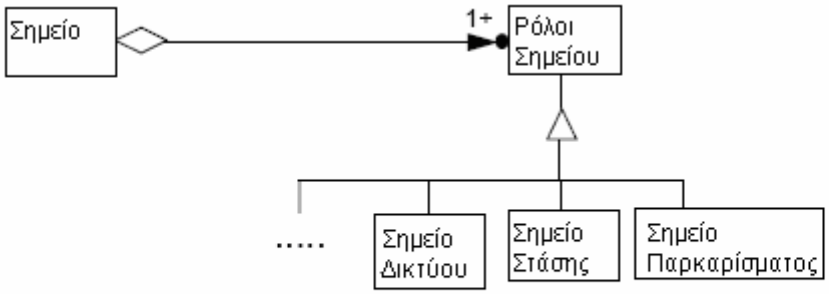
αποθηκευτικό αλγόριθμο για να διαβάσει τους πίνακες και να αποβάλλει τους αχρησιμοποίητους πίνακες.
Συνέπειες που έχει το πρότυπο αυτό είναι [28]:
Απόδοση: Η πρόσβαση ενός συστήματος πίνακα είναι γρηγορότερη από μια πρόσβαση βάσεων δεδομένων (νανοδευτερόλεπτα αντί των χιλιοστών του δευτερολέπτου).
Πλεονασμός και κόστος: Το μειονέκτημα είναι ότι εγκαθίσταται μια δεύτερη άδεια βάση δεδομένων στην κύρια μνήμη και ότι πρέπει να αναπτυχθούν οι διαδικασίες για τη διανομή δεδομένων, τη δοκιμή δεδομένων και την έκδοση δεδομένων. Ένα επιτραπέζιο σύστημα δεν είναι φτηνό.

9.5 Πρότυπα Αντικειμένων Μεταφορών (Transport objects patterns TOPs)

Τα σχεδιαστικά πρότυπα παρέχουν έναν φορμαλισμό για τη σύλληψη συστατικών που εμφανίζονται συχνά μέσα στα πρότυπα αντικειμένου δημόσιων συγκοινωνιών. Έδωσαν μια γλώσσα για την περιγραφή αυτών των συστατικών σε άλλα. Παρείχαν επίσης τα συστατικά οικοδόμησης και το πλαίσιο για νέες λύσεις σε προβλήματα που έχουν απασχολήσει πολλές φορές. Εδώ χρησιμοποιούνται δυο υπάρχοντα σχεδιαστικά πρότυπα, δηλαδή το Πρότυπο Κατάστασης (State Pattern) και το Πρότυπο Στρατηγικής (Strategy Pattern). Αυτά θα διαμορφώσουν μερικά αντικείμενα μεταφορών που διαδραματίζουν πολλαπλούς ταυτόχρονους ρόλους και περιέχουν πολλαπλάσιες λύσεις.
Ένα δημόσιο σύστημα μεταφοράς παρουσιάζει πολλά σύνθετα προβλήματα προγραμματισμού και σχεδιασμού που απαιτούν εύκαμπτες λύσεις λογισμικού που έχουμε κατασκευάσει από τα αντικείμενα μεταφορών.
Τα συστήματα μεταφοράς έχουν τα ακόλουθα χαρακτηριστικά γνωρίσματα [27].
Πολλά αντικείμενα μεταφορών διαδραματίζουν πολλούς ρόλους συγχρόνως. Παραδείγματος χάριν, ένα δίκτυο μεταφορών μπορεί να κατασκευαστεί από δύο βασικά συστατικά: σημεία και συνδέσεις μεταξύ αυτών των σημείων.
Ένα σημείο μπορεί να είναι ένα ΣημείοΣτάσης(StopPoint) όπου οι επιβάτες επιβιβάζονται και αποβιβάζονται από τα οχήματα, ένα ΣημείοΣταθμού (ReliefPoint) όπου οι οδηγοί μπορούν να αρχίσουν ή να τελειώσουν τις βάρδιές τους σε ένα ξεχωριστό όχημα, ή ένα ΣημείοΠληροφορίας (TimingPoint) στα οποία δίνονται πληροφορίες για τους χρόνους ταξιδιών. Στην πραγματικότητα, αυτοί οι ρόλοι πολλαπλασιάζονται καθώς όλο και περισσότερες εφαρμογές που περιλαμβάνουν αυτά τα σημεία υποστηρίζονται. Οι διαφορετικές αναπαραστάσεις αυτών των ρόλων μπορούν να οδηγήσουν στις διαφορετικές λύσεις στη δομή συστημάτων [27].
Υπάρχουν διάφορες μέθοδοι για μια λύση σε ένα πρόβλημα, καμία από τις οποίο δεν είναι συνήθως ικανοποιητική. Παραδείγματος χάριν, υπάρχουν διάφοροι τρόποι να υπολογιστεί ο τρέχοντας χρόνος από τη σύνδεση μεταξύ δύο σημείων, ανάλογα με παράγοντες όπως ο τύπος ημέρας και η ώρα της ημέρας στους οποίους πραγματοποιείται ένα ταξίδι, κανόνες κυκλοφορίας και τύπος οχήματος. Στα σημεία και στις συνδέσεις ορίζονται συχνά διαφορετικοί ρόλοι μέσα σε κάθε μέθοδο λύσης. Οι διαφορετικές αναπαραστάσεις αυτών των αλγορίθμων μπορούν επίσης να οδηγήσουν σε διαφορετικές λύσεις στη δομή συστημάτων [27].
Επιπλέον, οι διαδικασίες δημόσιων συγκοινωνιών έχουν πολλούς σύνθετους, αλληλοεξαρτώμενους περιορισμούς και κανόνες [24]. Είναι επίσης ένας τομέας σημαντικού εμπορικού συμφέροντος και για τον οποίο υπάρχει ιδιαίτερο ενδιαφέρον

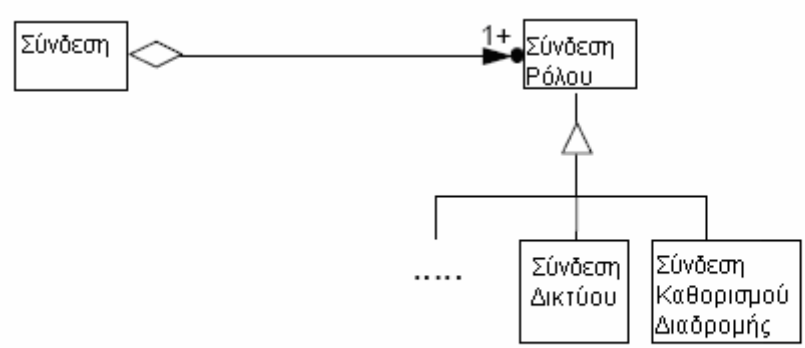
για την εύρεση των νέων αρχιτεκτονικών για την ενσωμάτωση των διαφορετικών εφαρμογών [25].
Θα εξεταστούν μερικά αντικείμενα μεταφορών, και πιο συγκεκριμένα σημεία, συνδέσεις, και ΜεταβλητέςΤαξιδιού (JourneyVariants). Όλα αυτά τα αντικείμενα εκθέτουν τα ανωτέρω χαρακτηριστικά γνωρίσματα των πολλών ταυτόχρονων ρόλων και των πολλών μεθόδων λύσης. Θα παρουσιαστεί πώς μερικά από τα πρότυπα της 'Συμμορίας των Τεσσάρων' (Gang of Four) [26], δηλαδή το Πρότυπο Κατάστασης και το Πρότυπο Στρατηγικής, έχουν προσαρμοστεί για τη διαμόρφωση αυτών των αντικειμένων. Το πρότυπο Κατάστασης αναφέρεται εδώ και ως Πρότυπο Ρόλου.

Σχεδιασμός των αντικειμένων μεταφοράς [27].
Ένα σημείο (point) στο δίκτυο μεταφορών παίζει πολλούς ρόλους. Ένα αντικείμενο σημείου αναπαρίσταται ως συνάθροιση ενός ή περισσότερων ρόλων σημείου, όπως φαίνεται στο σχήμα 9.5.1 Δεδομένου ότι ένα σημείο μπορεί να διαδραματίσει διαφορετικούς ρόλους συγχρόνως, τροποποιήθηκε το Πρότυπο Ρόλου για να προσαρμοστεί η πολλαπλότητα των ρόλων. Όλοι οι άλλοι ρόλοι που μπορεί να διαδραματίσει ένα σημείο θα οριστούν ως οι ειδικεύσεις της αφηρημένης κλάσης ΣημείοΡόλου. Ένα σημείο θα κρατήσει τις αναφορές σε όλες τις περιπτώσεις συγκεκριμένων κλάσεων ρόλου του. Αυτό το σημείο θα εξουσιοδοτήσει σε αυτές τις συγκεκριμένες κλάσεις ρόλου [27].



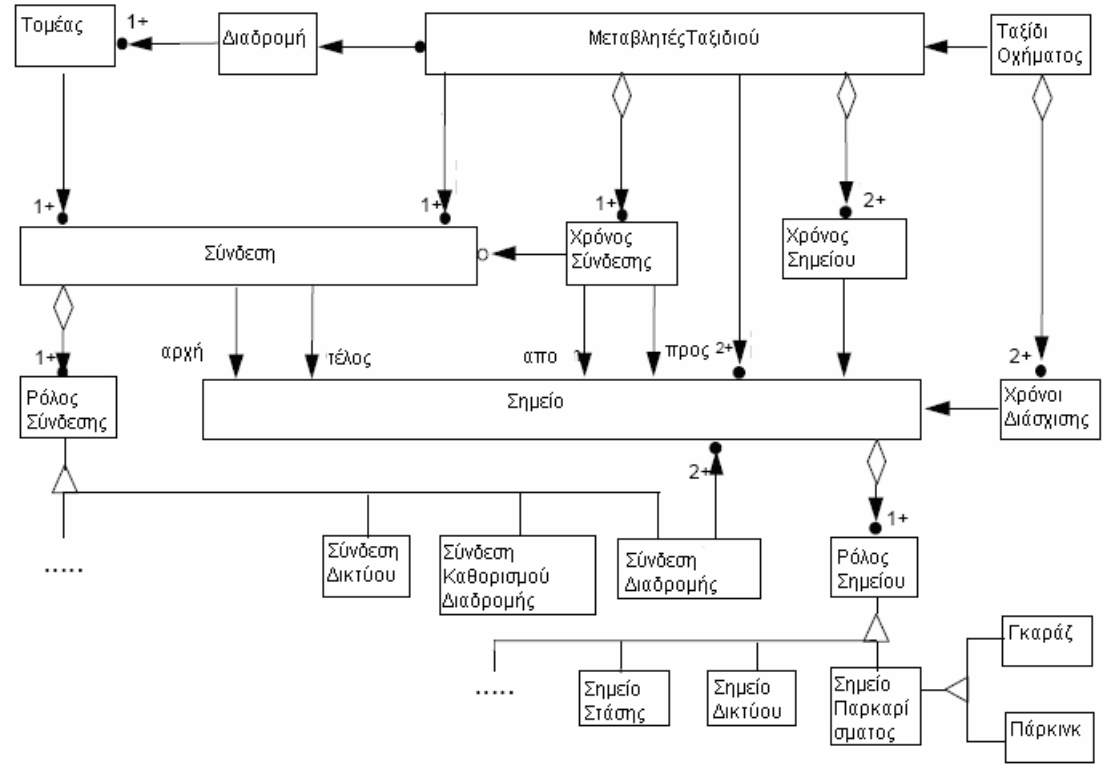
Σχήμα 9.5.1 Απεικονίζει ένα Σημείο που έχει ένα ή πολλούς ρόλους.

Ο ρόλος ενός σημείου στο δίκτυο είναι μεταβλητός. Ένα σημείο μπορεί να έχει πολλούς ευδιάκριτους ρόλους. Οτιδήποτε που είναι μεταβλητό, ή δεν είναι ένα άμεσο συστατικό ενός αντικειμένου, πρέπει πάντα να προτείνει μια έμμεση σύνθεση. Το σημείο πρέπει να έχει μια σχέση με μια χωριστή κλάση από την οποία προέρχεται κάθε συγκεκριμένος ρόλος. Με αυτόν τον τρόπο τα διάφορα γνωρίσματα συμπεριφοράς που παρέχονται από κάθε έναν από τους ρόλους μπορούν να προσεγγιστούν μέσω του σημείου από την εξουσιοδότηση.
Μια σύνδεση στο δίκτυο μεταφορών διαδραματίζει επίσης πολλούς ρόλους και επομένως αυτό αναπαρίσταται ως συνάθροιση ενός ή περισσότερων ρόλων συνδέσεων, όπως δείχνει το σχήμα 9.5.2. Όλοι οι άλλοι ρόλοι που μπορεί να παίξει μια σύνδεση θα οριστούν ως οι ειδικεύσεις της αφηρημένης κλάσης ΣύνδεσηΡόλου. Μια σύνδεση θα κρατήσει τις αναφορές σε όλες τις περιπτώσεις συγκεκριμένων κλάσεων ρόλου της. Αυτή η σύνδεση θα εξουσιοδοτήσει σε αυτές τις συγκεκριμένες κλάσεις ρόλου. Με άλλα λόγια, χρησιμοποιείται ακριβώς το ίδιο πρότυπο, για να αναπαρασταθούν και οι συνδέσεις και τα σημεία [27].



Σχήμα 9.5.2 Απεικονίζει μια Σύνδεση, η οποία μπορεί να έχει έναν ή περισσότερους ρόλους.

Ένας ρόλος που παίζεται από μια σύνδεση και οι ρόλοι που διαδραματίζονται από τα σημεία που συνδέει σε ένα δίκτυο μεταφορών, είναι πολύ κοντά συνδεδεμένα μαζί. Οι απόψεις για το πιο είναι το πιο βασικό αντικείμενο, διχάζονται. Οι διαδρομές εκφράζονται μερικές φορές ως διατεταγμένη ακολουθία τμημάτων, όπως φαίνεται στο παρακάτω σχήμα, τα οποία είναι η ίδια διατεταγμένη ακολουθία συνδέσεων ή και σημείων [27].



Σχήμα 9.5.3 Απεικονίζεται οι διαδρομές, που εκφράζονται μερικές φορές ως διατεταγμένη ακολουθία τμημάτων.

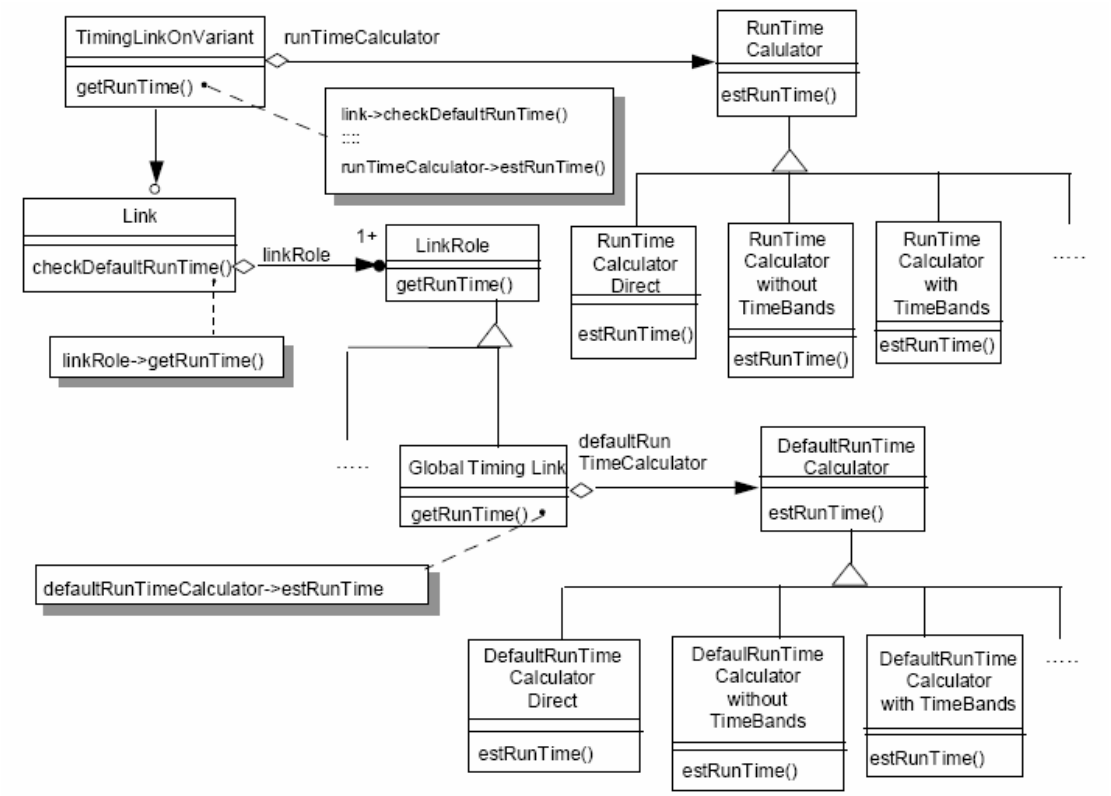
Τα ΣημείαΣυνένωσης είναι εκείνα τα σημεία στα οποία οι διαδρομές ενώνονται και διαχωρίζονται. Υπάρχουν πολλοί τρόποι να χωριστεί ένα δίκτυο μεταφορών σε μια ιεραρχία των μικρότερων ενδιάμεσων και βασικών συστατικών, κανένα από τα οποία δεν είναι καλύτερο από άλλο, υπό οποιαδήποτε έννοια [27].

Πρέπει να διαμορφωθεί μια έννοια που είναι ικανή να μεταφέρει τους χειριστές στον πραγματικό κόσμο και να παρέχει την ευελιξία στην αναπαράσταση λογισμικού, ώστε να ικανοποιήσουν οι διαφορετικοί τρόποι τη σχέση μεταξύ αυτών των εννοιών. Περισσότερες από μια μέθοδοι αναπαράστασης απαιτούνται, ακόμη και μέσα σε μια ενιαία εκτέλεση του συστήματος. Τα σχεδιαστικά πρότυπα βοηθούν για να επιτευχθεί ο στόχος.

Μια ΜεταβλητήΤαξιδιού είναι μια βασική έννοια μέσα σε ένα σύστημα μεταφοράς, επειδή περιγράφει τον τρόπο με τον οποίο ένα ή περισσότερα ταξίδια λειτουργούν κατά μήκος των διαδρομών μέσω του φυσικού δικτύου. Περιγράφει μια διατεταγμένη ακολουθία σημείων (ή συνδέσεων) μέσω της οποίας περνά ένα ή περισσότερα ταξίδια. Μια ΜεταβλητήΤαξιδιού περιγράφει ακριβώς τις πληροφορίες διαδρομών, διαμορφώνει επίσης τη γέφυρα για την περιγραφή της συμπεριφοράς των υπηρεσιών κατά μήκος της διαδρομής (π.χ. στάσεις οχημάτων, τους χρόνους που περιμένουν στα σημεία, τους χρόνους διάσχισης συνδέσεων). Μια ΜεταβλητήΤαξιδιού μπορεί να είναι αρμόδια για τη γνώση όλων των σημείων μέσω των οποίων περνά, αλλά ένα σημείο μπορεί να μην πρέπει να ξέρει σε ποια ΜεταβλητήΤαξιδιού βρίσκεται [27].
Το σχήμα 9.5.3 παρουσιάζει μια ιδιαίτερη διαμόρφωση των κύριων εννοιών και των ενώσεών τους. Οι ενώσεις σε αυτό το σχήμα δείχνονται ως ομοιοκατευθυνόμενες για να υπογραμμίσουν τις σχέσεις του ενός αντικειμένου με το άλλο. Ένα ΜεταβλητήΤαξιδιού μπορεί να είναι αρμόδιο για τη γνώση όλων των σημείων μέσω των οποίων περνά, αλλά ένα σημείο μπορεί να μην ξέρει σε ποιο ΜεταβλητήΤαξιδιού βρίσκεται. Μπορεί ένα σημείο να κρατήσει απλά μια αρίθμηση του τρέχοντος αριθμού ΜεταβλητήΤαξιδιού στον οποίο βρίσκεται μια ευθύνη του συγκεκριμένου ρόλου ΣημείοΚαθορισμούΔιαδρομής. Οι ομοιοκατευθυνόμενες ενώσεις έχουν το ιδιαίτερο προσόν της μείωσης της σύζευξης μεταξύ των αντικειμένων, και αυτό μπορεί μακροπρόθεσμα να ξεπεράσει οποιοδήποτε μειονέκτημα λόγω της πιο αργής εκτέλεσης μερικών ελέγχων συνέπειας [27].
Η ΣύνδεσηΡόλου είναι η αφηρημένη κλάση ρόλου που παρέχει τη διεπαφή μεταξύ της Σύνδεσης και των συγκεκριμένων ρόλων του. Η ΣύνδεσηΡόλου κάνει το ίδιο πράγμα για το Σημείο.
Η ΧρόνοςΣύνδεσης είναι μια ενδιαφέρουσα έννοια. Είναι μια πολλαπλή σχέση μεταξύ της ΜεταβλητήςΤαξιδιού και της Σύνδεσης. Είναι απαραίτητο, επειδή χρησιμοποιείται ως κάτοχος θέσεων για τους τρέχοντας χρόνους οχημάτων μεταξύ μιας Σύνδεσης στα πλαίσια ενός συγκεκριμένου ΠαράμετροςΤαξιδιού. Ο τρέχοντας χρόνος στην σύνδεση μπορεί να ποικίλει από ένα ΠαράμετροςΤαξιδιού σε άλλο. Ο τρέχοντας χρόνος σε μια σύνδεση μπορεί να διαδραματίσει το ρόλο ενός GlobalTimingLink (δηλ. που δεν συνδέεται με ένα συγκεκριμένο ΠαράμετροςΤαξιδιού) μπορεί μόνο να κρατήσει τους τρέχοντας χρόνους που μπορούν να θεωρηθούν αρκετά ακριβείς για μερικές εφαρμογές, αλλά όχι για άλλες. Ένα ΧρόνοςΣημείου είναι μια πολλαπλή σχέση, μεταξύ ΠαράμετροςΤαξιδιού και του Σημείου και χρησιμοποιείται ως κάτοχος θέσεων για τους χρόνους αναμονής των οχημάτων [27].
Η Σύνδεση και το Σημείο υπάρχουν στο δίκτυο ανεξάρτητα από τις ΠαράμετροςΤαξιδιού. Όχι όμως και οι ΧρόνοςΣημείου και ΧρόνοςΣύνδεσης. Όταν μια ΠαράμετροςΤαξιδιού καταστρέφεται, καταστρέφονται και όλες οι σχετικές ΧρόνοςΣημείου και ΧρόνοςΣύνδεσης. Το σύμβολο συνάθροισης στο διάγραμμα δείχνει αυτήν την εξάρτηση.
Δεδομένου ότι διάφοροι ΠαράμετροςΤαξιδιού μπορεί να περάσουν μέσω του ίδιου Σημείου, αυτό σημαίνει ότι διάφορα στιγμιότυπα όχι ΧρόνοςΣημείου μπορούν να

μοιραστούν ένα στιγμιότυπο Σημείου. Μόλις ένα ΠαράμετροςΤαξιδιού ορίζεται να περάσει μέσω ενός σημείου, τότε εκείνο το σημείο παίρνει το ρόλο ενός ΣημείοΚαθορισμούΤαξιδιού. Παύει να διαδραματίζει εκείνο τον ρόλο μόνο όταν διαγραφούν όλα τα ΠαράμετροςΤαξιδιού, που περνούν μέσω του [27].

Μεταξύ τα τέλη δεκαετίας του '80 και αρχές της δεκαετίας του '90 όταν άρχισε η σχεδίαση των προτύπων αντικειμένων μεταφοράς, η κληρονομικότητα παρά η αντιπροσώπευση ήταν πιο προεξέχουσα τεχνική και για τη γνώση και για την τεχνολογία λογισμικού. Η κληρονομικότητα κλάσεων ήταν το ουσιαστικό κατασκεύασμα για την αναπαράσταση γνώσης και υποστηρίχθηκε από τα περισσότερα εργαλεία ανάπτυξης βασισμένων στη γνώση συστημάτων. Οι περισσότερες πρόσφατα αναπτυγμένες μεθοδολογίες τεχνολογίας λογισμικού προώθησαν τη χρήση της κληρονομικότητας για το σχεδιασμό και την εφαρμογή. Οι γλώσσες προγραμματισμού που αναπτύχθηκαν έκτοτε επίσης υποστήριξαν την κωδικοποίηση κληρονομικότητας. Η αρχική διαμόρφωσή των αντικειμένων μεταφοράς επηρεάστηκε πάρα πολύ από την έμφαση στην κληρονομικότητα. Το βιβλίο σχεδιαστικών προτύπων Gang of Four [22] είναι μια σημαντική βοήθεια στην επίδειξη των πλεονεκτημάτων που αποκομίζονται με τη χρησιμοποίηση της αντιπροσώπευσης καθώς επίσης και της κληρονομικότητας για την επίλυση πολλών από τα προβλήματα μεταφορών.
Το παράδειγμά τους του προτύπου Κατάστασης έπεισε ότι η Κατάσταση είναι μια φυσική και καλύτερη λύση για τα Σημεία και τις Συνδέσεις. Έχει αλλάξει το όνομα Κατάσταση στο Ρόλος για να απεικονιστεί η πρόθεσή μας καλύτερα. Με τη χρησιμοποίηση του προτύπου Ρόλου, δημιουργήθηκε ένας πολύ απλό και πιο επεκτάσιμο πρότυπο για τους ρόλους σημείων και συνδέσεων. Για παράδειγμα, ένας νέος ρόλος για ένα σημείο μπορεί να είναι ένα ΣημείοΕλέγχουΚίνησης (TrafficControlPoint) και αυτό μπορεί να προστεθεί στο πρότυπό χωρίς οποιαδήποτε αλλαγή στη συμπεριφορά ενός σημείου.
Στο πρώτο σχήμα χρησιμοποιήθηκε μια ιεραρχία κλάσεων δύο επιπέδων για να αναπαρασταθεί η ειδίκευση ΣημείοΠαρκαρίσματος. Ο λόγος για αυτό είναι ότι η κλάση ΣημείοΠαρκαρίσματος είναι μια αφηρημένη κλάση που ειδικεύεται περαιτέρω στις κλάσεις Γκαράζ και Πάρκινκ. Με αυτόν τον τρόπο, έχουν συντηρηθεί τα επίπεδα αφαίρεσης. Είναι προτιμότερο κάθε ενδιάμεση κλάση επιπέδων σε μια ιεραρχία να γίνεται αφηρημένη κλάση [27].
Η εξάπλωση των αντικειμένων ρόλου μέσα στο σύστημά περιλαμβάνει κάποια γενικά έξοδα. Πρέπει να εξεταστεί εάν η αυξανόμενη ευελιξία που παρέχεται από το πρότυπο Ρόλου δικαιολογεί την απομάκρυνση από άλλους τρόπους αναπαράστασης ρόλων. Αρκετά συχνά οι χειριστές δημόσιων συγκοινωνιών θέλουν μια περιγραφή των ρόλων των αντικειμένων μεταφοράς μόνο για λόγους παρουσίασης.
Η ευελιξία προστέθηκε χρησιμοποιώντας το πρότυπο Στρατηγικής. Παραδείγματος χάριν, στο παρόν πρότυπο, η ευθύνη υπολογισμού διαφορετικής πληροφορίας χρόνου εξουσιοδοτήθηκε σε ένα Κατασκευαστή Χρονοδιαγράμματος. Ο Κατασκευαστής Χρονοδιαγράμματος αποτελείται από δύο πρότυπα στρατηγικής, που λειτουργούν σε διάφορα επίπεδα. Σε κάθε επίπεδο, καθορίζονται οι διαφορετικές μέθοδοι υπολογισμού. Με αυτόν τον τρόπο, δημιουργήθηκε μια πιο εύκαμπτη δομή [27].



Σχήμα 9.5.4 Απεικονίζονται πολλά πρότυπα Στρατηγικής για (runtime) υπολογισμούς σε τρέχοντα χρόνο.

Έχουμε προσδιορίσει τρία επίπεδα στο σχήμα 9.5.4 στο οποίο φυλάσσονται οι τρέχοντας πληροφορίες.
Αυτά είναι:
1). GlobalTimingLink. Αυτό δεν επιτρέπει τις παραλλαγές πέρα από την ίδια σύνδεση σε διαφορετική ΜεταβλητήΤαξιδιού (JourneyVariants).
2). TimingLinkOnVariant. Αυτό επιτρέπει τις παραλλαγές από ΜεταβλητήΤαξιδιού (JourneyVariant) και εκτοπίζει πάντα οποιαδήποτε προκαθορισμένη τιμή που εισάγεται σε ένα GlobalTimingLink.
3). TimingLinkOnJourney. Αυτό είναι συγκεκριμένο για ένα μεμονωμένο ταξίδι και εκτοπίζει πάντα οποιουσδήποτε χρόνους που εκφράζονται στα άλλα δύο επίπεδα. Δεν παρουσιάζεται ρητά στα διαγράμματα. Είναι αυτονόητο μέσα σε ένα JourneyRunTime.
Στα επίπεδα 1 και 2, υπάρχουν περαιτέρω παραλλαγές λόγω των διαφορετικών μεθόδων τέτοιους παράγοντες όπως ΤύποςΗμέρας (DayType) και ΩραΗμέρας (TimeOfDay). Υπάρχουν διάφοροι τρέχοντας χρόνοι που εκφράζονται για κάθε σύνδεση που αντιστοιχεί στους διαφορετικούς χρόνους της ημέρας. Αυτό το σύνολο χρόνων εκτέλεσης μπορεί να διαφέρει στους διαφορετικούς τύπους ημερών (δηλ., οι Δευτέρες διαφέρουν από τις Κυριακές) ή DayTypeGroups (δηλ., εργάσιμες μέρες, Σαββατοκύριακα).
Τρία επιπλέον επίπεδα παραλλαγής παρουσιάζονται:
1). DirectRunTimes. Αυτοί είναι κλάσεις τομής μεταξύ τριών άλλων κλάσεων, η κλάση Σύνδεσης, DayTypeGroup, και TimeOfDay.
2). Run times without TimeBands. Σε αυτές τις περιπτώσεις, ένα κλειδί διευκρινίζει ποιος χρόνος εκτέλεσης στη χρήση συνδέεται με κάθε μεμονωμένο ταξίδι οχημάτων.
3). Run times with TimeBands. Σε αυτές τις περιπτώσεις, ένα κλειδί διευκρινίζει ποιος χρόνος εκτέλεσης στη χρήση συνδέεται με ένα προκαθορισμένο TimeBand και

αυτό που επιλέγεται υπολογίζεται αυτόματα για κάθε ταξίδι ανάλογα με το χρόνο της ημέρας που αρχίζει να διαβαίνει κάθε σύνδεση.
Αναπαριστώντας διαφορετικούς τρέχοντος χρόνου υπολογισμούς ως πρότυπα στρατηγικής, επίσης απεικονίζει το πραγματικό κόσμο, όπου οι εμπειρογνώμονες συνδυάζουν συνήθως περισσότερες από μια μεθόδους στην τελική λύση τους στο πρόβλημα.
Ο συνδυασμός των προτύπων Ρόλου και Στρατηγικής παρέχει μια αφηρημένη διεπαφή και για τους ρόλους και για τις στρατηγικές. Τα αντικείμενά σημείου και συνδέσεων διαμορφώνονται με έναν τέτοιο συνδυασμό. Το πρότυπο Ρόλου εφαρμόζει τους ρόλους που ένα αντικείμενο διαδραματίζει , ενώ το πρότυπο Στρατηγικής εφαρμόζει αλγοριθμικές παραλλαγές. Μερικές από αυτήν την παραλλαγή μπορούν να αφορούν τους διαφορετικούς ρόλους, οπότε σε αυτή την περίπτωση οι διαδικασίες μπορούν ακόμα να ενσωματωθούν μέσα στις συγκεκριμένες κλάσεις ρόλου [27].

Αναφορές:

[1]: Κωνσταντίνος Αντώνης, (2003),Slides: *Ασφάλεια Υπολογιστικών Συστημάτων, Προστασία επικοινωνιών- Κρυπτογραφία.*
[2]: *An Introduction to Computer Security: the NIST handbook*, 1995, National Institute of Standards and Technology (NIST), Special Publication 800-12.
[3]: N.Asokan, Philippe A. Janson, Michael Steiner and Michael Waidner, (2007), *The State of the Art in Electronic Payment Systems*. IEEE Computer, pages 28-35, September 1997.
[4]: ΣΤΕΦΑΝΟΣ ΓΚΡΙΤΖΑΛΗΣ-ΣΩΚΡΑΤΗΣ Κ. ΚΑΤΣΙΚΑΣ, ΔΗΜΗΤΡΗΣ ΓΚΡΙΖΑΛΗΣ *ΑΣΦΑΛΕΙΑ ΔΙΚΤΥΩΝ ΥΠΟΛΟΓΙΣΤΩΝ*, ΕΚΔΟΣΕΙΣ ΠΑΠΑΣΩΤΗΡΙΟΥ 2003, σελ. 70-140.
[5]: Pau-Chen Cheng, Juan A. Garay, Amir Herzberg and Hugo Krawczyk, (1998), *A Security Architecture for the Internet Protocol*. IBM Systems Journal 1998.
[6]: Amir Herzeberg and Hilik Yochai, (1997), *Minipay: Charging per Click on the Web*. Computer Networks and ISDN Systems, 1997.
[7]: Amir Herzberg and Dalit Naor, (1998), *Surf'N'Sign: Client Signatures on Web Documents*, IBM Systems Journal, 1998.
[8]: Ε. Σακκόπουλος, (2008), Slides: *Τεχνολογίες Εφαρμογών Διαδικτύου Ασφάλεια στο Web*, Πανεπιστήμιο Στερεάς Ελλάδος Τμήμα Πληροφορικής με εφαρμογές στη Βιοιατρική.
[9]: Tom Davis, *Cryptography* http://www.geometer.org/mathcircles , February 7, 2000.
[10]: *Authentication*, U.S. Government Printing Office, Office of Information Dissemination Program Development Service, Washington, D.C., October 13, 2005.
[11]: Martín Abadi, Cédric Fournet, (2004), *Private authentication*. Theor. Comput. Sci. 322 (3): 427-476 (2004).
[12]: Leslie Ikemoto, (2004), CS276 Class Notes for Lecture 8 (second half) *Message Integrity* February 18, 2004.
[13]: http://code.google.com/intl/el-GR/edu/submissions/daswani/index.html
[14]: Manfred Lange, (1998), *Time Patterns*, Hewlett-Packard GmbH, NSL (Network Support Lab).
[15]: Robert Martin et al., (1998), "*Pattern Languages of Program Design 3*", Reading Massachusetts, 1998, Addison-Wesley.
[16]: Christian August Crusius, (1995), "*Entwurf der nothwendigen Vernunft-Wahrheiten, wiefern sie den zufälligen entgegen gesetzet werden.*", Die philosophischen Hauptwerke. Hrsg. Von Giorgio Tonelli. Bd. 2. Hildesheim 1964.
[17]: Paul Schönsleben, Ruth Leuzinger, (1996), *Innovative Gestaltung von Versicherungsprodukten.; Flexible Industriekonzepte in der Assekuranz*, Gabler.
[18]: Michael E. Porter, (1985), *Competitive Advantage*; The Free Press 1985.
[19]: Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, (1996), *Pattern Oriented Software Architecture - A System of Patterns*, Wiley 1996.
[20]: Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, (1986), *Compilers: Principles, Techniques, and Tools*,Addison-Wesley 1986.
[21]: Martin Fowler, (1997), *Analysis Patterns*; Addison Wesley Longman, 1997.

[22]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, (1995), *Design Patterns, Elements of Reusable Object-oriented Software*, Addison-Wesley 1995.
[23]: Wolfgang Keller, (1997), *Mapping Objects to Tables: A Pattern Language*, Proceedings of the 1997 European Pattern Languages of Programming Conference, Irrsee, Germany, Siemens Technical Report 120/SW1/FB 1997.
[24]: Zhao, L and E. Foster (1994). *ROO: rules and object-orientation*. In C. Mingins and B. Meyer (Eds.), Technology of Object-Oriented Language and Systems: TOOLS15, pp. 31-44. Prentice-Hall.
[25]: Foster, E. (1991). *The Cassiope project*. In International Workshop: Data Management for Public Transport, East Magdeburg, Germany.
[26]: Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley.
[27]: Ted Foster and Liping Zhao, (1996), *TOPs: transport object patterns – timetable constructor.*
[28]: Wolfgang Keller, (1998), *Some Patterns for Insurance Systems* , Conference Plop 1998.
[29]: Alexandre M. Braga ,Cecilia M. F. Rubira, Ricardo Dahab, *Tropyc: A Pattern Language for Cryptographic Software,* State University of Campinas Institute of Computing, Conference Plop 1998.
[30]: http://msdn.microsoft.com/en-us/library/ms994921.aspx

.

10. ΑΝΑΖΗΤΗΣΗ ΣΥΝΔΥΑΣΜΩΝ ΚΑΙ ΑΛΛΩΝ ΠΡΟΤΥΠΩΝ ΓΙΑ ΤΟ ΣΧΕΔΙΑΣΜΟ ΣΥΣΤΗΜΑΤΩΝ ΛΟΓΙΣΜΙΚΟΥ ΥΨΗΛΩΝ ΕΠΙΔΟΣΕΩΝ

10.1 Εισαγωγικά στοιχεία

Για να διευκολυνθεί ο σχεδιασμός συστημάτων υψηλών επιδόσεων επιβάλλεται η αναζήτηση νέων προτύπων η οποία μπορεί να προκύψει και από συνδυασμό των ήδη υπαρχόντων. Τα συστήματα υψηλών επιδόσεων λειτουργούν σε περιβάλλοντα πολλών επεξεργαστών (multiprocessor) ή συστάδων υπολογιστών (clusters).
Όπως έχει ήδη αναφερθεί στα προηγούμενα κεφάλαια, τα σχεδιαστικά πρότυπα είναι γενικές επαναχρησιμοποιήσιμες λύσεις στα συνεχόμενα εμφανιζόμενα προβλήματα του σχεδιασμού λογισμικού [1]. Τα περισσότερα από αυτά έχουν παρουσιαστεί στο [2]. Τα σχεδιαστικά πρότυπα στοχεύουν να βοηθήσουν τους αναλυτές και τους σχεδιαστές λογισμικού να λύσουν τα προβλήματα προγραμματισμού και σχεδιασμού, τα οποία συχνά παρουσιάζονται κατά τη διάρκεια του κύκλου της ζωής της ανάπτυξης λογισμικού. Η εφαρμογή τους μπορεί να επιταχύνει και να διευκολύνει τις διαδικασίες του προγραμματισμού και του σχεδιασμού, της ανάπτυξης και της εγκατάστασης του λογισμικού με τη χρήση των ήδη δοκιμασμένων και καλά-αποδεδειγμένων μεθόδων. Η επαναχρησιμοποίηση των σχεδιαστικών προτύπων αποτρέπει την εμφάνιση των σημαντικών προβλημάτων στο λογισμικό και διευκολύνει την αναγνωσιμότητα, τη συντήρηση και την κατανόηση του κώδικα λογισμικού.

Τα σχεδιαστικά πρότυπα χρησιμοποιούνται σε διάφορους τομείς και περιοχές της επιστήμης, όπως στη βιολογία, βιοϊατρικής, επικοινωνιών και εφαρμογές Ιστού. Παραδείγματος χάριν, στο έγγραφο [3] ένα σχεδιαστικό πρότυπο οντολογίας παρουσιάζεται προκειμένου να παρασχεθούν οι πρωταρχικές απαιτήσεις για τις βιοϊατρικές οντολογίες. Επιπλέον, τα σχεδιαστικά πρότυπα μπορούν να χρησιμοποιηθούν για να βελτιώσουν την ποιότητα των συστημάτων λογισμικού επικοινωνίας [4]. Τα πρότυπα λογισμικού έχουν προταθεί επίσης στον τομέα του υπολογισμού υψηλής επίδοσης (High Performance Computing) [5] και γενικότερα για τα ταυτόχρονα προγράμματα και τις διανεμημένες εφαρμογές [6] [7]. Τα σχεδιαστικά πρότυπα λογισμικού μπορούν να εφαρμοστούν για να λύσουν τα ταυτόχρονα προβλήματα λογισμικού [8]. Η εφαρμογή των σχεδιαστικών προτύπων στα ταυτόχρονα συστήματα λογισμικού μπορεί να μειώσει το χρόνο ανάπτυξης λογισμικού, να βελτιώσει τη συντηρησιμότητα κώδικα, και να αυξήσει την επαναχρησιμοποίηση κώδικα πέρα από τις παραδοσιακές τεχνικές τεχνολογίας λογισμικού [9]. Τα πρότυπα μπορούν επίσης να εφαρμοστούν στο σχεδιασμό αντικειμενοστρεφών συστημάτων που τρέχουν σε πολυεπεξεργαστές ή περιβάλλοντα συστάδων.

Τα ταυτόχρονα και διανεμημένα συστήματα γίνονται δημοφιλή στα πλαίσια των σχεδιαστικών προτύπων λογισμικού [10]. Εντούτοις, τα υπάρχοντα πρότυπα για το σχεδιασμό των αντικειμενοστρεφών συστημάτων για υπολογισμό υψηλής επίδοσης [11] δεν μπορούν να καλύψουν εξ ολοκλήρου τις ανάγκες των σχεδιαστών. Αρκετοί διαμορφώνουν τα βασισμένα συστήματα έχουν αναπτυχθεί προκειμένου να διευκολυνθεί η γρηγορότερη παράλληλη ανάπτυξη εφαρμογών. Τα παράλληλα σχεδιαστικά πρότυπα υπολογισμού έχουν χρησιμοποιηθεί για την ανάπτυξη των προ-εφαρμοσμένων και επαναχρησιμοποιήσιμων συστατικών. Εντούτοις, τα περισσότερα από αυτά τα συστήματα αντιμετωπίζουν διάφορους σοβαρούς περιορισμούς όπως, περιορισμένη ευελιξία και μηδενική επεκτασιμότητα [12].

Ο υπολογισμός υψηλών επιδόσεων αντιμετωπίζει πολλά σύνθετα προβλήματα, τα οποία τα υπάρχοντα σχεδιαστικά πρότυπα και τεχνικές δεν μπορούν να εξετάσουν επιτυχώς [11] [13] [12]. Ο στόχος είναι να προταθεί ο συνδυασμός προτύπων (υβριδικά πρότυπα), τα οποία στοχεύουν άμεσα να βοηθήσουν τους σχεδιαστές για να κατασκευάσουν καλύτερα συστήματα υψηλών επιδόσεων. Τα προτεινόμενα πρότυπα, συνδυάζουν τα πρότυπα συγχρονισμού με άλλα πρότυπα από διαφορετικές κατηγορίες, προκειμένου να λυθούν συγκεκριμένα επανεμφανιζόμενα προβλήματα και να δοθεί η ευκαιρία στα συστήματα υψηλής επίδοσης να γίνουν γρηγορότερα και πιό αξιόπιστα.

Πιο κάτω καθορίζονται δύο υβριδικά πρότυπα υψηλής επίδοσης που συνδυάζουν τα πρότυπα συγχρονισμού με τα σχεδιαστικά πρότυπα. Το πρώτο πρότυπο στοχεύει να μειώσει τους πόρους που απαιτούνται για να ολοκληρώσει το σύστημα μια εφαρμογή και συνδυάζει το πρότυπο Χρονοπρογραμματιστή και το πρότυπο Πληρεξούσιου. Το δεύτερο υβριδικό πρότυπο εξασφαλίζει ότι πολλοί χρήστες μπορούν να έχουν πρόσβαση σε διαφορετικά στοιχεία, χωρίς λάθη και συνδυάζει το πρότυπο Ανάγνωση-γραφής κλειδώματος και το πρότυπο Στρατηγικής.

10.2 Συνδυασμός προτύπων Χρονοπρογραμματιστή και Πληρεξουσίου

Αυτός ο συνδυασμός προτείνει μια λύση στο πρόβλημα της ανεξαρτησίας του πελάτη και των κλάσεων που χρησιμοποιεί, από τη διαδικασία της πρόσβασης στα αντικείμενα αυτών των κλάσεων.

Ο πελάτης δεν πέχει αναφορά στο πραγματικό αντικείμενο. Εντούτοις αναφορά σε ένα αντικείμενο (πληρεξούσιο) που δημιουργεί τη διεπαφή. Όταν το αντικείμενο πρέπει πραγματικά να χρησιμοποιηθεί, κατόπιν το πρότυπο Χρονοπρογραμματιστή χρησιμοποιεί ένα αντικείμενο προκειμένου να σχεδιαστούν τα ταυτόχρονα αιτήματα από τα νήματα για την μη συγχρονισμένη διαδικασία. Κάθε νήμα που προσπαθεί να εκτελέσει μια χρονοπρογραμματισμένη μέθοδο αποτρέπεται έως ότου ξυπνιέται.

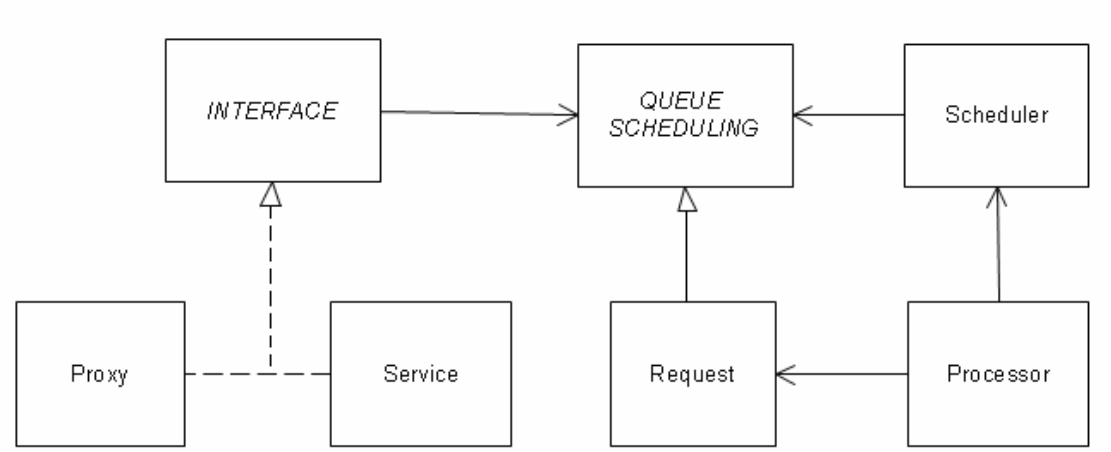
Ο πελάτης παρά το γεγονός που δεν έχει καμία αναφορά στο πραγματικό αντικείμενο, αυτό έχει μια αναφορά σε ένα αντικείμενο πληρεξούσιου από τη διεπαφή. Ο πελάτης χειρίζεται το πληρεξούσιο με τον ίδιο τρόπο που θα χειριζόταν την υπηρεσία και επικαλείται μια μέθοδο εφαρμογής. Ο πελάτης μπορεί τώρα να δημιουργήσει ένα αντικείμενο υπηρεσια και να καλέσει μια μέθοδο εφαρμογής από το αντικείμενο.

Όπως φαίνεται και στο σχήμα 10.2.1, το αντικείμενο Αίτηση ενθυλακώνει ένα αίτημα για ένα αντικείμενο Επεξεργαστή. Τα στιγμιότυπα των κλάσεων Επεξεργαστή υπολογίζουν έναν υπολογισμό, ο οποίος περιγράφεται από το αντικείμενο Αίτημα. Αυτά τα στιγμιότυπα μπορούν να αντιπροσωπευθούν από περισσότερα από ένα αντικείμενα Αίτηση, που επεξεργάζονται εκείνη τη στιγμή, αλλά μπορούν να επεξεργάζονται ένα κάθε φορά. Το αντικείμενο Επεξεργαστή αντιπροσωπεύει στο αντικειμένου Χρονοπρογραμματιστή την ευθύνη χρονοπρογραμματισμού ενός αντικειμένου Αιτήματος τη φορά.
Τα στιγμιότυπα των κλάσεων Χρονοπρογραμματιστή, χρονοπραμματίζουν τα αντικείμενα Αίτησης. Μια κλάση Χρονοπρογραμματιστή δεν γνωρίζει τίποτα για την κλάση Αίτησης που δημιουργεί. Έχει πρόσβαση στα αντικείμενα Αίτησης μέσω της διεπαφής Προγραμματισμό Σειράς, το οποίο εφαρμόζουν. Μια κλάση Χρονοπρογραμματιστή αποφασίζει πότε το επόμενο αίτημα θα τρέξει. Δεν είναι αρμόδιο για τη σειρά με την οποία τα αιτήματα θα εκτελεστούν.

Τα αντικείμενα Αίτηση εφαρμόζουν τη διεπαφή Προγραμματισμός Σειράς. Αυτή η διεπαφή εξυπηρετεί δύο στόχους. Αρχικά, έχοντας αναφορά σε μια διεπαφή Προγραμματισμός Σειράς, οι κλάσεις Επεξεργαστή αποφεύγουν την εξάρτηση σε μια κλάση Αίτηση. Επίσης, καλώντας τις μεθόδους που καθορίζονται από τη διεπαφή Προγραμματισμός Σειράς, οι κλάσεις Χρονοπρογραμματιστή είναι σε θέση να αποφασίσουν ποιο αντικείμενο Αίτημα θα υποβληθεί σε επεξεργασία μετά. Αυτό αυξάνει την ικανότητα επαναχρησιμοποίησης των κλάσεων Χρονοπρογραμματιστή.

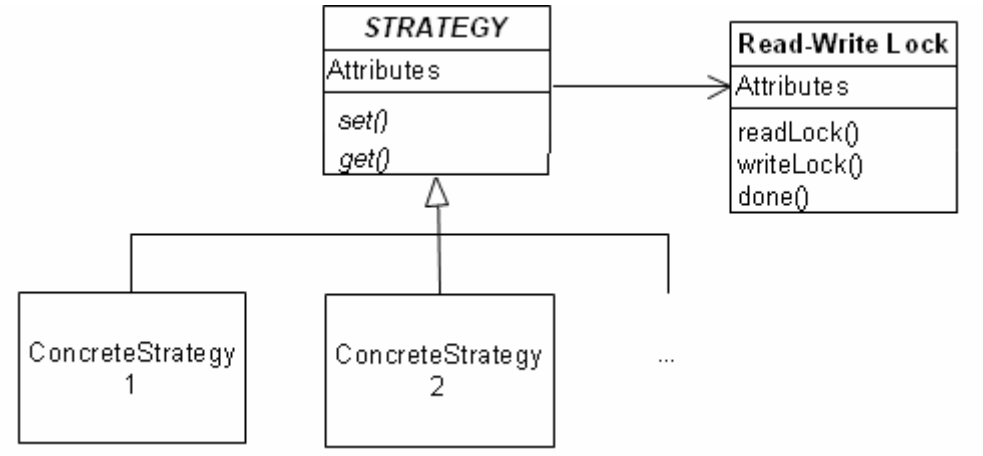
Με αυτόν τον συνδυασμό, τα οφέλη από το πρότυπο Χρονοπρογραμματιστή και το πρότυπο Πληρεξούσιου υπάρχουν μαζί και βοηθούν τη υψηλή επίδοση σε συγκεκριμένες περιοχές. Συγκεκριμένα, επιτυγχάνεται η ανάγκη για τη μείωση χρονικής διάρκειας της επεξεργασίας. Εάν ο χρήστης χρειαστεί την υπηρεσία που κατασκευάζεται, τότε το πρότυπο Χρονοπρογραμματιστή θα εκτελέσει το στόχο του. Εάν ένα μέρος της υπηρεσίας απαιτείται και όχι ολόκληρη η υπηρεσία, τότε εφαρμόζεται το πρότυπο Πληρεξούσιου με αποτέλεσμα να εξοικονομείται πολύτιμος χρόνος.

Χωρίς αυτόν τον συνδυασμό προτύπων, το πρότυπο Χρονοπρογραμματιστή θα είχε περιττή επεξεργασία. Ένα άλλο όφελος που είναι πολύ σημαντικό στα συστήματα υψηλής επίδοσης είναι ότι, οι πόροι που χρησιμοποιούνταν εάν η υπηρεσία δεν έπρεπε να κατασκευαστεί (το πρότυπο Χρονοπρογραμματιστή ενεργούσε μόνο) θα ήταν πολύ περισσότεροι.



Σχήμα 10.2.1 Απεικονίζεται το διάγραμμα κλάσεων του συνδυασμού των προτύπων Χρονοπρογραμματιστή και Πληρεξουσίου.

10.3 Συνδυασμός προτύπων Κλειδώματος Ανάγνωσης-Γραφής και Στρατηγικής



Σχήμα 10.3.1 Απεικονίζεται το διάγραμμα κλάσεων του συνδυασμού των προτύπων Στρατηγικής και Κλειδώματος Ανάγνωσης-Γραφής.

Όπως φαίνεται και στο σχήμα 10.3.1, οι διεπαφές στρατηγικής έχουν μεθόδους ορισμού και λήψης (set and get methods) για τις πληροφορίες των στιγμιοτύπων τους. Οποιοσδήποτε αριθμός νημάτων μπορεί ταυτόχρονα να πάρει τις πληροφορίες από ένα αντικείμενο Στρατηγική, μόνο εάν κανένα νήμα δεν θέτει πληροφορίες την ίδια στιγμή. Οι μέθοδοι ορισμού πρέπει να εμφανιστούν μια κάθε στιγμή, ενώ δεν

βρίσκεται σε εξέλιξη καμία λειτουργία. Τα αντικείμενα Στρατηγική πρέπει να συντονίσουν τις μεθόδους λήψης και ορισμού, έτσι ώστε να υπακούνε στους περιορισμούς. Οι κλάσεις ConcreteStrategies παρέχουν διαφορετικές συμπεριφορές και έχουν επίσης τις μεθόδους λήψης και ορισμού της διεπαφής.

Ο πελάτης αλληλεπιδρά με τη διεπαφή Στρατηγική, αλλά δεν ξέρει την πραγματική κλάση του αντικειμένου ή πώς η κλάση εφαρμόζει τη λειτουργία. Η διεπαφή είναι κοινή για κάθε στρατηγική. Μερικές συνέπειες είναι ότι αυτό το πρότυπο επιτρέπει σε μια από τις στρατηγικές για να επιλεχτεί δυναμικά και υπάρχει περισσότερη ευελιξία. Επιπλέον, το αποκλειστικό κλείδωμα ανάγνωσης-γραφής εξασφαλίζει ότι δεν θα υπάρξει οποιοσδήποτε ανταγωνισμός εάν ένας χρήστης γράφει και ένας άλλος χρήστης διαβάζει ή γράφει τις πληροφορίες. Πολλοί αναγνώστες μπορούν να χρησιμοποιήσουν άφοβα τις πληροφορίες και δεν υπάρχει σύγκρουση μεταξύ των πόρων [14] [15].

Οι μέθοδοι λήψης ενός αντικειμένου δεν παίρνουν πληροφορίες έως ότου πάρουν μια κλειδαριά ανάγνωσης. Κάθε αντικείμενο Κλειδώματος ανάγνωσης-γραφής συνδέεται με ένα αντικείμενο Στρατηγική. Προτού λάβει οτιδήποτε μια μέθοδος λήψης, καλεί τη μέθοδο κλείδωμαΑνάγνωσης (readLock) από το αντικείμενο Κλείδωμα ανάγνωσης-γραφής, το οποίο δίνει ένα κλείδωμα ανάγνωσης στο πραγματικό νήμα που τρέχει. Ενώ το νήμα έχει ένα κλείδωμα ανάγνωσης, η μέθοδος λήψης είναι βέβαιο ότι είναι ασφαλής να πάρει δεδομένα από το αντικείμενο. Αυτό συμβαίνει επειδή, όσο υπάρχουν κλειδώματα ανάγνωσης, το αντικείμενο Κλειδώματος γραφής δεν θα δημοσιεύσει κλείδωμα γραφής. Εάν οι κλειδαριές γραφής δεν έχουν τελειώσει την εργασία τους, τότε καλείται η μέθοδος κλείδωμαΑνάγνωσης (readLock) του αντικειμένου Κλειδώματος ανάγνωσης-γραφής, τίποτα δεν επιστρέφεται έως ότου όλες οι μέθοδοι κλείδωμαΓραφής έχουν ολοκληρώσει τις εργασίες τους. Οι κλήσεις στη μέθοδο κλείδωμαΑνάγνωσης (readLock) επιστρέφουν τα δεδομένα [14].

Όταν η μέθοδος λήψης έχει τελειώσει την εργασία της λαμβάνοντας δεδομένα από το αντικείμενο, καλεί τη μέθοδο ολοκλήρωση (done) από το αντικείμενο Κλειδώματος ανάγνωσης-γραφής. Αυτή η κλήση αναγκάζει το τρέχον νήμα για να σταματήσει τη κλειδαριά ανάγνωσης [16].

Αυτός ο δεύτερος συνδυασμός έχει μερικά πολύ κρίσιμα πλεονεκτήματα στο σχεδιασμό υπολογισμού υψηλής επίδοσης. Το πρώτο όφελος είναι ότι πολλοί πελάτες μπορούν να χρησιμοποιήσουν τη διεπαφή Στρατηγική και να έχουν πρόσβαση στα δεδομένα. Αυτή η πρόσβαση περιλαμβάνει τον ορισμό και τη λήψη πληροφοριών. Το στοιχείο περιέχει διαφορετικές πληροφορίες λόγω των διαφορετικών στρατηγικών. Το δεύτερο πλεονέκτημα είναι ότι, με αυτόν τον συνδυασμό τα λάθη που θα εμφανίζονταν, έχουν ελαχιστοποιηθεί μέσω του κλειδώματος. Οι πελάτες μπορούν ταυτόχρονα να έχουν πρόσβαση χωρίς να πρέπει να εξεταστούν τα λάθη που θα εμφανίζονταν, αν δύο πελάτες έγραφαν ταυτόχρονα.

Αναφορές:

[1] http://en.wikipedia.org/wiki/Design_patterns_(computer_science)
[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, Massachusetts, 1995.

[3] Aldo Gangemi , Carol Catenacci, Massimo Battagli, (2004), *Inflammation Ontology Design Pattern: an exercise in building a core biomedical ontology with Descriptions and Situations*, Proceedings of the 10th Bio-Ontologies Special Interest Group Workshop 2007.
[4] Douglas C. Schmidt Paul Stephenson , *Experience Using Design Patterns to Evolve Communication Software Across Diverse OS Platforms*, Department of Computer Science Ericsson, Inc. Washington University, St. Louis, MO 63130 Cypress, CA 90630
[5] Hoang M. Song, (2006), *Modelling, Simulation and Optimization of Complex Processes*, Proceedings of the Third International Conference on High Performance Scientific Computing, March 6–10, 2006, Hanoi, Vietnam.
[6] D. Lea, (2000), *Concurrent Programming in Java: Design Principles and Patterns*. Addison-Wesley, Reading, Massachusetts, second edition, 2000.
[7] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, (2000) *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*. John Wiley & Sons, *Chichester*, England, 2000.
[8] Douglas C. Schmidt, (1997), *Applying Patterns to Meet the Challenges of Concurrent Software*, Invited contribution to the IEEE Concurrency to the IEEE Concurrency Special Edition on Software Engineering for Parallel and Distributed Systems, Vol. 5, No. 3, Fall 1997.
[9] James Hu, Irfan Pyarali, Douglas C. Schmidt, (1998), *APPLYING THE PROACTOR PATTERN TO HIGH-PERFORMANCE WEB SERVERS*, In Proceedings of the 10th International Conference on Parallel and Distributed Computing and Systems.
[10] Ninja Shi and Ronald A. Olsson, (2005), *Reverse Engineering of Design Patterns for High Performance Computing*, 28th International Conference on Software Engineering, Shangai, 2006.
[11] Douglas C. Schmidt, (1995), *Active Object An Object Behavioral Pattern for Concurrent Programming*, Proc.Pattern Languages of Programs.
[12] Dhrubajyoti Goswami, Ajit Singh, and Bruno R. Preiss, (2002), *From Design Patterns to Parallel Architectural Skeletons*, Journal of Parallel and Distributed Computing, Volume 62 , Issue 4 (April 2002), pp. 669 – 695.
[13] Weiguo Liu and Bertil Schmidt, (2008), *A Case Study on Pattern-based Systems for High Performance Computational Biology*, Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International.
[14] Mark Grand, 2002. *Patterns in Java, Volume 1: A Catalog of Reusable Design Patterns Illustrated with UML*, Second Edition, John Wiley & Sons © 2002.
[15] Eric Freeman & Elisabeth Freeman With Kathy Sierra & Bert Bates, (2004), *Strategy Pattern Head First Design Pattern O'Reilly*, First Ed. Oct 2004.
[16] Cleidson R. B. de Souza and Roberto S. Silva Filho, (1995), *Checking Java Concurrency Design Patterns Using Bandera*, Department of Information and Computer Science, University of California, Irvine

11. ΕΦΑΡΜΟΓΕΣ ΤΩΝ ΣΧΕΔΙΑΣΤΙΚΩΝ ΠΡΟΤΥΠΩΝ ΣΤΟ ΣΧΕΔΙΑΣΜΟ ΠΡΟΤΥΠΟΥ ΙΑΤΡΙΚΟΥ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΤΗ ΔΙΑΓΝΩΣΗ ΚΑΙ ΘΕΡΑΠΕΙΑ ΚΑΡΔΙΑΚΩΝ ΠΑΘΗΣΕΩΝ

Στο κεφάλαιο αυτό παρουσιάζεται τμήμα πληροφοριακού συστήματος που ασχολείται με την συγκοπή καρδιάς. Είναι πολύ σημαντικό ότι, το προτεινόμενο σύστημα εξετάζει τις περιπτώσεις που μπορούν να συμβούν όταν ένας ασθενής νοσεί από συγκοπή καρδιάς ή είναι πιθανό να εμφανίσει. Πιο συγκεκριμένα, εξετάζονται οι πιθανές ασθένειες τα συμπτώματα, οι συμβουλές των καρδιολόγων, οι εξετάσεις και η διάγνωση, καθώς και η εισαγωγή του ασθενή και η χειρουργική επέμβαση.

Όπως έχει ήδη αναφερθεί μερικά από τα πλεονεκτήματα της αντκειμενοστρεφής σχεδίασης είναι η εύκολη κατανόηση σύνθετων συστημάτων και η επαναχρησιμοποίηση [12]. Τα σχεδιαστικά πρότυπα προτείνουν λύσεις σε επαναλαμβανόμενα προβλήματα και χρησιμοποιούνται για ανάπτυξη αντικειμενοστρεφούς λογισμικού [13][14]. Σε ένα σύστημα υγείας αυτά τα πρότυπα μπορούν να χρησιμοποιηθούν για να κατασκευαστεί ένα οργανωμένο σύστημα [9]. Τα συγκεκριμένα πρότυπα λογισμικού που χρησιμοποιήθηκαν στο πληροφοριακό σύστημα υγείας είναι το πρότυπο Πρόσοψης, πρότυπο Εικονικού Πληρεξουσίου, πρότυπο Αφηρημένης Υπέρ-κλάσης, το πρότυπο Singleton και το πρότυπο Ανάγνωση Μόνο Διεπαφής.

11.1 Πληροφορική στο χώρο της υγείας

Η δύναμη της πληροφορικής επανάστασης πολλαπλασιάζεται, ενώ το κόστος της πέφτει κατακόρυφα. Η τεχνολογία της πληροφορικής μετασχηματίζει τα σπίτια μας, τη ζωή μας στο χώρο εργασίας, τον κοινωνικό μας ιστό, τις κοινωνίες μας, το περιβάλλον μας και αναπόφευκτα και τις υπηρεσίες υγείας.
Τα τελευταία χρόνια η ανάγκη για μείωση του κόστους, οδήγησε στην ανάπτυξη της υγείας και της πληροφορικής στην υγεία. Ο χώρος της υγείας είναι ιδιαίτερα ευαίσθητος και οι υπολογιστές βρίσκουν σημαντικότατες εφαρμογές σε όλους τους τομείς της υγείας, όπως και στην καρδιολογία. Η αποτελεσματική αντιμετώπιση των περιστατικών είναι ζήτημα υψίστης σημασίας.
Οι υπολογιστές βρίσκουν σημαντικότατες εφαρμογές σε όλους τους τομείς της υγείας και της Ιατρικής επιστήμης. Τα διάφορα είδη των υπολογιστών και οι ειδικοί τύποι αυτών, χρησιμοποιούνται τόσο στην διαγνωστική , όσο και στην προληπτική Ιατρική. Τα χειρουργεία, οι χώροι ανάνηψης, οι διάφορες εξετάσεις και η παρακολούθηση των ασθενών δεν νοούνται στην σύγχρονη Ιατρική χωρίς την υποστήριξη ιατρικών μηχανημάτων συνδεδεμένα και υποστηριζόμενα από ηλεκτρονικούς υπολογιστές [8].

Τα συστήματα υγείας έχουν τέσσερις βασικούς στόχους.
• Να βελτιώνουν την υγεία του πληθυσμού που υπηρετούν
• Να ανταποκρίνονται στις προσδοκίες των πολιτών
• Να παρέχουν οικονομική προστασία στον πολίτη για τις δαπάνες
υγείας και
• Να θεραπεύουν την ασθένεια με ποιοτική ιατροφαρμακευτική φροντίδα [7].
Η ανακάλυψη νέων φαρμάκων, η πρόοδος στην τεχνολογία, η πρόοδος της ίδιας της ιατρικής επιστήμης και ο αυξανόμενος μέσος όρος ηλικίας, οδηγούν στη δημιουργία συστημάτων, τα οποία να μπορούν να βοηθήσουν αποτελεσματικά την περίθαλψη των ασθενών. Επίσης οι ιατρικοί φάκελοι των ασθενών, έχουν μεγάλη χρησιμότητα.

Η εφαρμογή της σύγχρονης τεχνολογίας της πληροφορικής στην υγεία
δύναται να φέρει τα πιο κάτω θετικά αποτελέσματα.
• Παρέχει εύκολη και γρήγορη πρόσβαση σε αναζητούμενες πληροφορίες.
• Η επεξεργασία των δεδομένων γίνεται πιο γρήγορα και χωρίς λάθη.
• Τα δεδομένα αποθηκεύονται , καταλαμβάνοντας μικρό χώρο (cd, dvd, σκληροί δίσκοι).
• Παρέχει μεγαλύτερο βάθος και ουσία στην πληροφόρηση
• Πληροφορεί και υποστηρίζει τις κλινικές αποφάσεις.
• Πληροφορεί και υποστηρίζει τις διαχειριστικές και διοικητικές αποφάσεις.

• Διευκολύνει την ανταλλαγή πληροφοριών μεταξύ των διαφόρων προμηθευτών υπηρεσιών υγείας.
• Μειώνει το χρόνο νοσηλείας.
• Μειώνει το κόστος και το μέσο όρο ημερών παραμονής στο νοσοκομείο.
• Μειώνει τον αριθμό των επισκέψεων στα νοσοκομεία
• Παρέχει στον ασθενή βελτιωμένες πληροφορίες για την υγεία του και υποβοηθά στην προαγωγή της υγείας. Αυξάνει την αποδοτικότητα των πόρων στην υγεία.
• Μεταβάλλεται σε εργαλείο και μέσο παροχής υπηρεσιών υγείας [7].

11.2 Σχεδίαση με UML

Η πιο διαδεδομένη γλώσσα περιγραφής συστημάτων λογισμικού είναι η Ενοποιημένη Γλώσσα Μοντελοποίησης (Unified Modeling Language- UML). Η UML είναι μια γραφηματική (graphical) γλώσσα μοντελοποίησης. Χρησιμοποιείται για προδιαγραφές, αναπαράσταση με οπτικό τρόπο (visualizing), δημιουργία και τεκμηρίωση των τμημάτων λογισμικού καθώς και για μοντελοποίηση εταιρικών και άλλων συστημάτων που δεν αφορούν λογισμικό.
Ένα σύστημα κατά την UML θεωρείται ότι αποτελείται από αντικείμενα τα οποία συνεργάζονται μεταξύ τους και μέσα από συνεργασίες επιτυγχάνεται ο σκοπός που αυτό το σύστημα πρέπει να υπηρετεί. Μια συνεργασία υλοποιείται με την ανταλλαγή μηνυμάτων μεταξύ των αντικειμένων που συμμετέχουν σε αυτήν. Η ανταλλαγή των μηνυμάτων γίνεται με ένα κατάλληλο τρόπο ανάλογο των αναγκών που υπάρχουν.
Περιέχει διάφορων ειδών διαγράμματα . Τα διαγράμματα περιπτώσεως χρήσης (use case diagrams) μπορούν να περιγράψουν τη λειτουργικότητα, ενώ τα διαγράμματα κλάσεων (class diagrams) μπορούν να περιγράψουν τη δομή μιας διαδικασίας. Ένα διάγραμμα καταστάσεων (state diagram) μπορεί να αναπαραστήσει τη συμπεριφορά και τις καταστάσεις στις οποίες μπορεί να βρεθεί μια δραστηριότητα σε μια φάση της διαδικασίας (ανενεργή, υπό έναρξη, υπό διακοπή, σε αναμονή, υπό εκτέλεση, περατωμένη κτλ). Τα διαγράμματα ακολουθίας (sequential diagrams), δραστηριοτήτων (activity diagrams) και συνεργασίας (collaboration diagrams) περιγράφουν την αλληλεπίδραση μεταξύ των αντικειμένων και των δρώντων στοιχείων (actors) που εμπλέκονται σε μια διαδικασία [10].
Μερικά πλεονεκτήματα που παρέχει η αντικειμενοστρεφής σχεδίαση είναι η αφαιρετική αναπαράσταση, η κατανόηση σύνθετων συστημάτων, η επαναχρησιμοποίηση και η εύκολη προσαρμογή των συστημάτων και των διαδικασιών. Δηλαδή εστιάζει στα σημαντικά στοιχεία και κρύβει λεπτομέρειες και άσχετα στοιχεία. Μοντελοποιώντας περιορίζουμε το πρόβλημα εστιάζοντας σε επιμέρους πλευρές του συστήματος (διαίρει και βασίλευε) και κλίμακες αφαίρεσης [11].

11.3 Συγκοπή καρδιάς

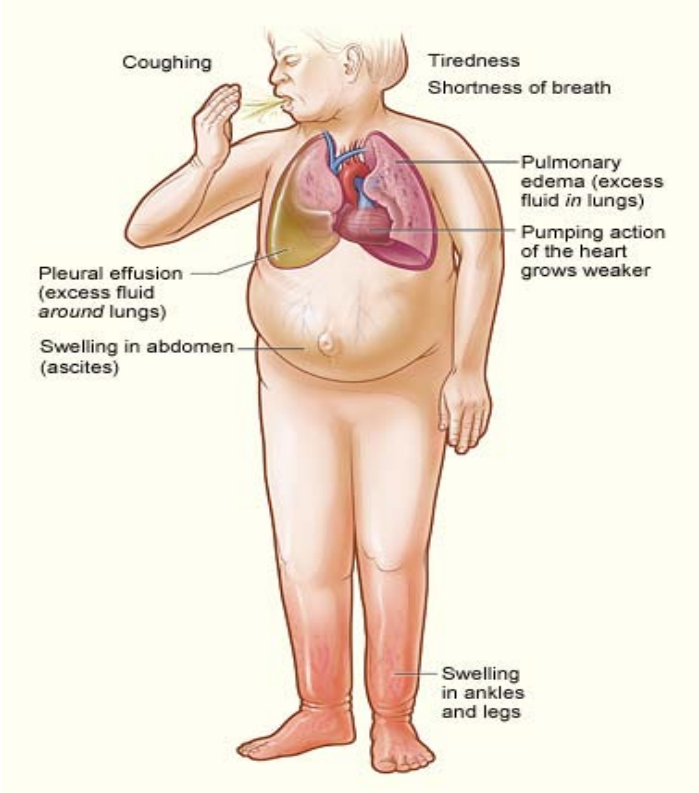
Η συγκοπή καρδιάς είναι ένα σύνθετο σύνδρομο που μπορεί να προκύψει από οποιαδήποτε δομική ή λειτουργική καρδιακή αναταραχή που εξασθενίζει τη δυνατότητα της καρδιάς να λειτουργήσει ως αντλία για να υποστηρίξει μια φυσιολογική κυκλοφορία. Το σύνδρομο της συγκοπής καρδιάς χαρακτηρίζεται από συμπτώματα όπως δυσκολία στην αναπνοή και κούραση. Δεν υπάρχει καμία διαγνωστική εξέταση για τη συγκοπή καρδιάς, και η διάγνωση στηρίζεται στην

κλινική κρίση βασισμένη σε έναν συνδυασμό ιστορίας, τη φυσική εξέταση και τις κατάλληλες έρευνες [1].
Είναι το τελευταίο στάδιο όλων των ασθενειών της καρδιάς και είναι μια σημαντική αιτία της νοσηρότητας και της θνησιμότητας.
Δυστυχώς, η συγκοπή καρδιάς είναι δύσκολο να εντοπιστεί κλινικά, δεδομένου ότι πολλά χαρακτηριστικά γνωρίσματα του όρου δεν είναι συγκεκριμένα και μπορούν να υπάρξουν λίγα κλινικά χαρακτηριστικά γνωρίσματα στα πρώτα στάδια της ασθένειας.
Οι πρόσφατες πρόοδοι έχουν καταστήσει την πρόωρη αναγνώριση της συγκοπής καρδιάς όλο και περισσότερο σημαντική, δεδομένου ότι η σύγχρονη θεραπεία φαρμάκων έχει τη δυνατότητα να βελτιώσει τα συμπτώματα και την ποιότητα ζωής, να μειώσει τα ποσοστά εισαγωγής σε νοσοκομεία, να επιβραδύνει το ποσοστό προόδου ασθενειών, και να βελτιώσει την επιβίωση [3].
Επίσης, είναι μια σχετικά κοινή χρόνια αναταραχή και θεωρείται παράδειγμα των καρδιακών χρόνιων ασθενειών. Αυτή η αναταραχή έχει επιπτώσεις κυρίως στους ηλικιωμένους από 65 και άνω. Η αύξηση στην υπολογιζόμενη διάρκεια ζωής στις αναπτυγμένες χώρες έχει οδηγήσει σε μια αύξηση στον αριθμό εισαγωγών σε νοσοκομείο λόγω των χρόνιων αρρωστιών, καθώς επίσης και σε μια πιθανή μείωση στην ποιότητα ζωής του γηράσκοντος πληθυσμού [2].
Η συγκοπή καρδιάς είναι ένας καρδιακός όρος που εμφανίζεται όταν εξασθενίζει ένα πρόβλημα με τη δομή ή τη λειτουργία της καρδιάς τη δυνατότητά του να παρέχει την ικανοποιητική ροή αίματος για να ικανοποιήσει τις ανάγκες του σώματος. Με την κατάλληλη θεραπεία, η συγκοπή καρδιάς μπορεί να ρυθμιστεί στην πλειοψηφία των ασθενών, αλλά συνδέεται και με μια ετήσια θνησιμότητα 10%.
Η ρευστή υπερφόρτωση (fluid overload) είναι ένα κοινό πρόβλημα για τους ανθρώπους με συγκοπή καρδιάς, αλλά δεν είναι συνώνυμη με αυτή. Οι ασθενείς με την αντιμετωπισμένη συγκοπή καρδιάς θα είναι συχνά euvolaemic (ένας όρος για την κανονική ρευστή θέση), ή σπανιότερα, αφυδατωμένοι [4].
Οι γιατροί χρησιμοποιούν τον όρο «οξεία» για γρήγορη αρχή, και «χρόνια» για μακροχρόνια διάρκεια. Η χρόνια συγκοπή καρδιάς είναι επομένως μια μακροπρόθεσμη κατάσταση, συνήθως με σταθερή αντιμετωπίσιμη συμπτωματολογία. Η οξεία συγκοπή καρδιάς, περιγράφει την ξαφνική αρχή HF (Heart Failure), ή συγκοπή καρδιάς, που αναφέρεται στα επεισόδια στα οποία ένας ασθενής με τη γνωστή χρόνια συγκοπή καρδιάς αναπτύσσει απότομα τα συμπτώματα [4].

Ασθένειες που σχετίζονται με τη συγκοπή της καρδιάς

Οι ασθένειες που οδηγούν στην συγκοπή της καρδιάς είναι πολλές και οι παράγοντες που επηρεάζουν την συγκοπή είναι πολλοί και διαφορετικοί.
Οι στεφανιαίες καρδιακές παθήσεις (Coronary artery diseases) είναι η πιο συχνή αιτία της συγκοπής καρδιάς στις δυτικές χώρες. Οι στεφανιαίοι παράγοντες κινδύνου, όπως το κάπνισμα και ο διαβήτης mellitus, είναι δείκτες κινδύνου της ανάπτυξης της συγκοπής καρδιάς. Το κάπνισμα είναι ένας ανεξάρτητος και ισχυρός παράγοντας κινδύνου για την ανάπτυξη της συγκοπής καρδιάς.
Η υπέρταση (Hypertension) είναι συνδεδεμένη με αυξανόμενο κίνδυνο συγκοπής καρδιάς, είναι μια περισσότερο κοινή αιτία της συγκοπής καρδιάς σε επιλεγμένες ομάδες ασθενών, όπως των γυναικών και των μαύρων πληθυσμών.
Οι καρδιομυοπάθειες (Cardiomyopathies) ορίζονται ως οι ασθένειες του καρδιακού μυός, που δεν είναι δευτερεύουσες στη στεφανιαία ασθένεια, την υπέρταση, ή σε άλλες ασθένειες. Οι καρδιομυοπάθειες είναι λιγότερο κοινές αιτίες της συγκοπής

καρδιάς. Είναι χωρισμένες σε τέσσερις λειτουργικές κατηγορίες: διεσταλμένη (dilated), υπερτροφική (hypertrophic), περιοριστική (restrictive) και obliterative. Η διεσταλμένη καρδιομυοπάθεια είναι μια περισσότερο κοινή αιτία της συγκοπής καρδιάς από τις υπόλοιπες τρεις και περιγράφει την ασθένεια μυών καρδιών στην οποία η κυρίαρχη ανωμαλία είναι θερμική μεταβολή διαστάσεων της αριστερής κοιλίας, με ή χωρίς σωστή κοιλιακή θερμική μεταβολή διαστάσεων. Η υπερτροφική χαρακτηρίζεται από τις ανωμαλίες των μυοκαρδιακών ινών, οι οποίες δημιουργούν διαφορετικές πιέσεις και αρρυθμίες με αποτέλεσμα τον αιφνίδιο θάνατο.
Οι ασθένειες βαλβίδων (Valvar diseases), ανάλογα με τη φύση (στένωση ή χάλαση) και την έκταση το προβλήματος, οδηγούν και σε κίνδυνο για συγκοπή.
Οι καρδιακές αρρυθμίες είναι κοινές στους ασθενείς με συγκοπή καρδιάς. Ο ενδοκολπικός ινιδισμός και η συγκοπή καρδιάς συνυπάρχουν συχνά, και αυτό έχει επιβεβαιωθεί σε δοκιμές μεγάλων κλιμάκων και μικρότερες βασισμένες σε νοσοκομειακές μελέτες. Οι κοιλιακές αρρυθμίες είναι επίσης κοινές στη συγκοπή καρδιάς, που οδηγεί σε μια ξαφνική επιδείνωση σε ορισμένους ασθενείς [3].



Heart Failure [4]

Το Αλκοόλ και τα Ναρκωτικά (Alcohol and Drugs) έχουν μια άμεση τοξική επίδραση στην καρδιά, η οποία μπορεί να οδηγήσει στην οξεία συγκοπή καρδιάς ή τη συγκοπή καρδιάς ως αποτέλεσμα των αρρυθμιών.

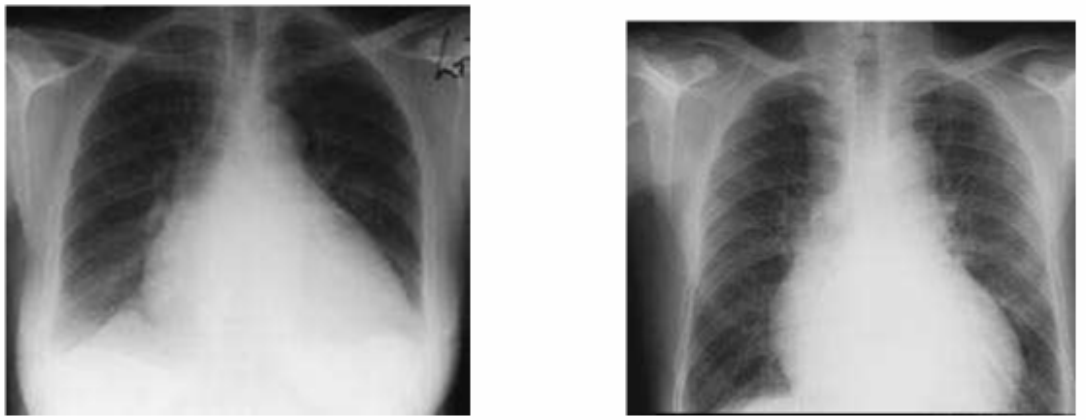
Συμπτώματα της συγκοπής καρδιάς

Ασθενείς με συγκοπή καρδιάς παρουσιάζουν ποικίλα συμπτώματα, τα περισσότερα από τα οποία είναι μη συγκεκριμένα. Τα κοινά συμπτώματα είναι η δύσπνοια

(dyspnea) , η κούραση και ο λήθαργος (tiredness and lethargy) , το οίδημα (oedema) κυρίως στα πόδια και τους αστραγάλους. Σε μερικές περιπτώσεις μπορεί να υπάρχει δυσκολία ύπνου, ανορεξία (anorexia), αύξηση βάρους (increase of weight), βήχας (cough) και μειωμένη ουρίνη (decreased ourin). Τα παραπάνω συμπτώματα χωρίζονται από τους γιατρούς και σε συμπτώματα αριστερής και δεξιάς πλευράς της καρδιάς.

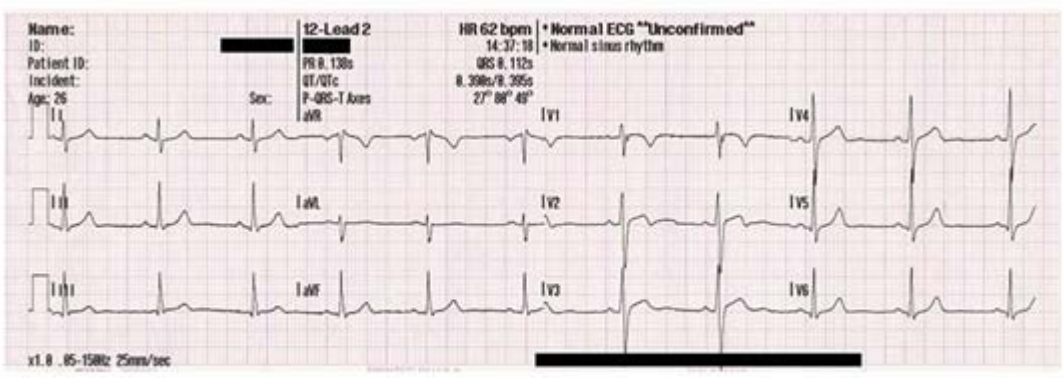
Διάγνωση της συγκοπής καρδιάς

Εκτός από την εξέταση που κάνει ο γιατρός στον ασθενή, είναι απαραίτητες και άλλες αντικειμενικότερες εξετάσεις για να διαγνωστεί η πιθανότητα συγκοπής καρδιάς.
Η θωρακική των ακτινών Χ εξέταση (Chest X ray examination) έχει έναν σημαντικό ρόλο στη στερεότυπη έρευνα για τους ασθενείς με πιθανή συγκοπή καρδιάς, και μπορεί επίσης να είναι χρήσιμη στον έλεγχο της προόδου της θεραπείας. Ανάλογα με το είδος των ακτινογραφιών και σε συνδυασμό με άλλες εξετάσεις, μπορεί να γίνει σωστή διάγνωση.



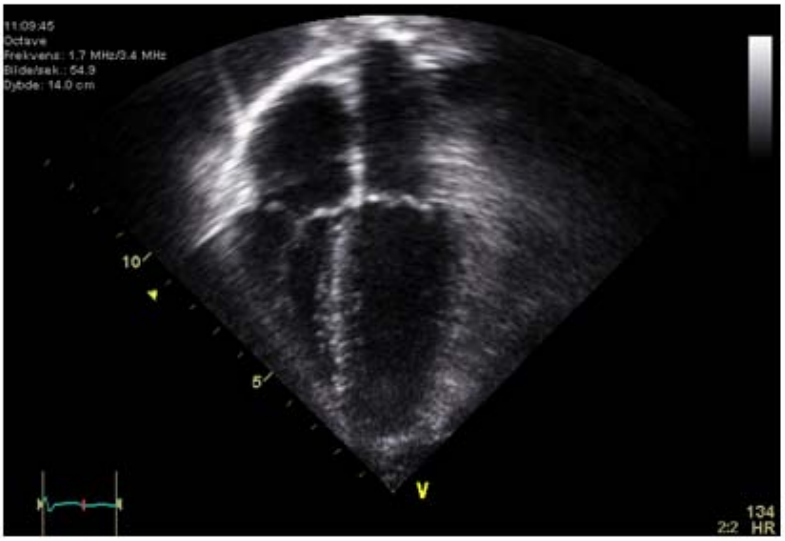
Ακτινογραφίες θώρακος που παρουσιάζουν ασθενή με διεσταλμένη καρδιομυοπάθεια και πνευμονική συμφόρηση

Οι εξετάσεις ηλεκτροκαρδιογραφήματος 12 σημείων (Electrocardiography 12 lead) δεν είναι φυσιολογικές στους περισσότερους ασθενείς με συγκοπή καρδιάς, αλλά μπορεί να είναι φυσιολογικές στο 10% των περιπτώσεων. Οι ανωμαλίες περιλαμβάνουν τα κύματα του Q, τις ανωμαλίες στο κύμα T και το τμήμα του ST, την αριστερή κοιλιακή υπερτροφία, το φραγμό κλάδων δεσμών, και τον ενδοκολπικό ινιδισμό [3] [5].



12 σημείων ηλεκτροκαρδιογράφημα ενός φυσιολογικού ατόμου.

Η ηχοκαρδιογραφία (Echocardiography) χρησιμοποιείται για να εντοπίσει τις καρδιαγγειακές παθήσεις. Στην πραγματικότητα, είναι μια από τις ευρύτατα χρησιμοποιημένες διαγνωστικές δοκιμές για τις καρδιακές παθήσεις. Μπορεί να παρέχει έναν πλούτο χρήσιμων πληροφοριών, συμπεριλαμβανομένου του μεγέθους και της μορφής της καρδιάς, της ικανότητας άντλησής της και της θέσης και της έκτασης οποιασδήποτε ζημίας στους ιστούς της. Είναι ιδιαίτερα χρήσιμο για τις ασθένειες των βαλβίδων καρδιών. Όχι μόνο επιτρέπει στους γιατρούς να αξιολογήσουν τις βαλβίδες καρδιάς, αλλά μπορούν να ανιχνεύσουν τις ανωμαλίες στο σχέδιο της ροής αίματος, όπως η οπίσθια ροή του αίματος μέσω των εν μέρει κλειστών βαλβίδων καρδιάς. Με την αξιολόγηση της κίνησης του τοιχώματος της καρδιάς, η ηχοκαρδιογραφία μπορεί να βοηθήσει να ανιχνεύσει την παρουσία και να αξιολογήσει τη δριμύτητα της ασθένειας στεφανιαίων αρτηριών, καθώς επίσης να καθορίσει εάν οποιοσδήποτε θωρακικός πόνος συσχετίζεται με τις καρδιακές παθήσεις. Το μεγαλύτερο πλεονέκτημα είναι ότι δεν απαιτεί το σπάσιμο του δέρματος ή την είσοδο των κοιλοτήτων σωμάτων και δεν έχει καμία παρενέργεια [6].

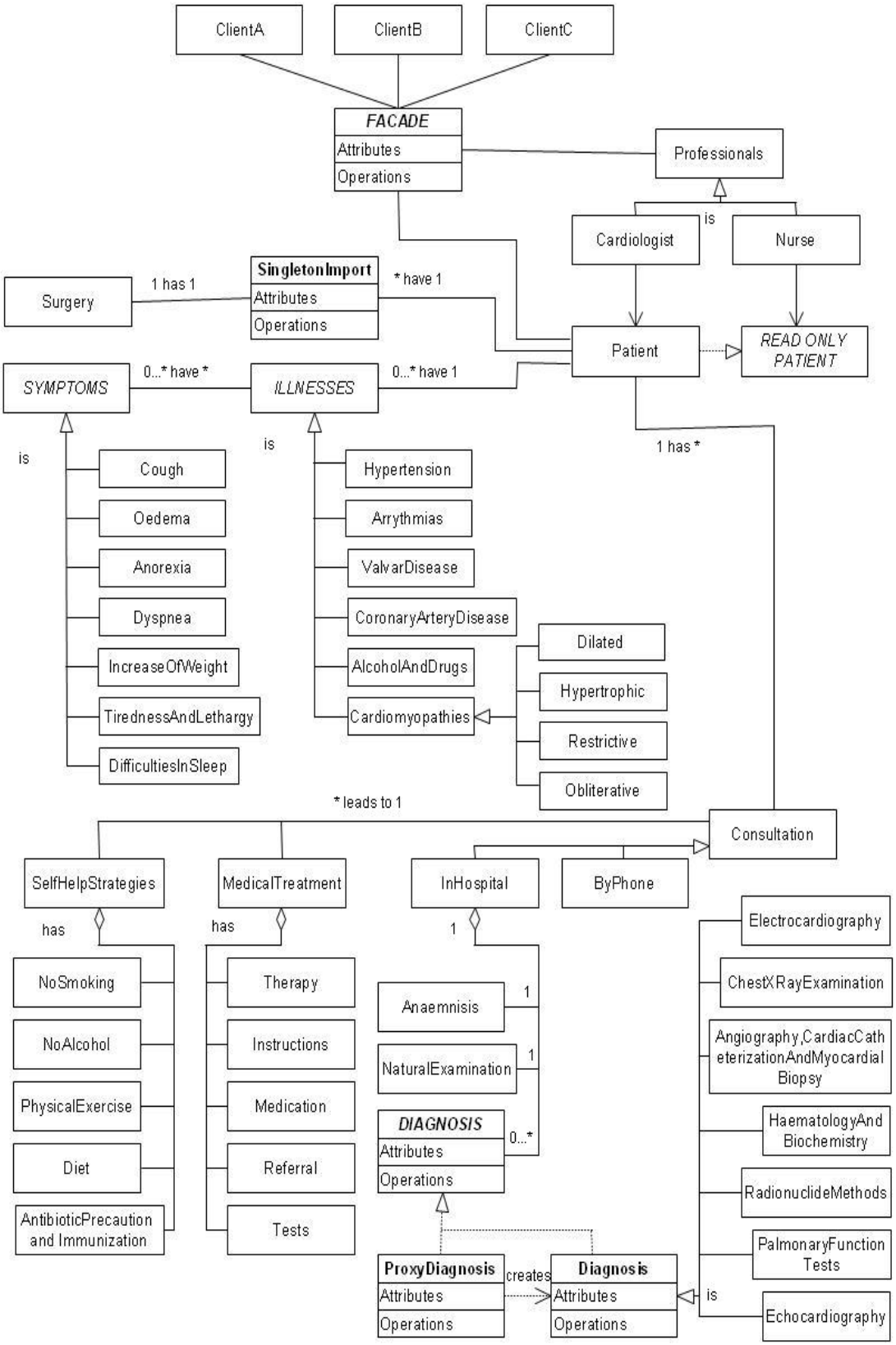


Ηχοκαρδιογραμμα, που δείχνει οτι η ανθρώπινη καρδιά έχει 4 κοιλότητες.

Οι αιματολογικές και οι βιοχημικές εξετάσεις (Hematology and biochemistry) γίνονται για να εξεταστεί πόσο η αναιμία (αν υπάρχει) επηρεάζει ή επηρεάζεται από την συγκοπή καρδιάς και από τη δυσκολία να αναπνεύσει ο ασθενής . Οι συγκεντρώσεις ορισμένων ουσιών στον οργανισμό του ασθενούς, δίνουν πολύ χρήσιμα συμπεράσματα για την πρόοδο των θεραπειών.
Η απεικόνιση των ραδιονουκλειδίων (Radionuclide Methods) χρησιμοποιείται όταν δεν είναι δυνατό να ληφθούν εικόνες με ηχοκαρδιογραφία και επιτρέπει την αξιολόγηση της αριστερής και δεξιάς κοιλίας την καρδιάς. Οι μελέτες ραδιονουκλεϊδίου είναι επίσης πολύτιμες για την αξιολόγηση του μυοκαρδιακού ραντίσματος και της παρουσίας ή της έκτασης της στεφανιαίας ισχαιμίας.
Η αγγειογραφία (Angiography) πρέπει να εξεταστεί στους ασθενείς με επαναλαμβανόμενο ισχαιμικό θωρακικό πόνο που συνδέεται με τη συγκοπή καρδιάς και σε εκείνους με τα στοιχεία της αυστηρής αντιστρέψιμης ισχαιμίας. Ο καρδιακός καθετηριασμός (Cardiac catheterization) με τη μυοκαρδιακή βιοψία (Myocardial biopsy) μπορεί να είναι χρήσιμος σε δυσκολότερες περιπτώσεις όπου υπάρχει διαγνωστική αμφιβολία [3].
Η αντικειμενική μέτρηση της λειτουργίας πνευμόνων (Pulmonary function tests) είναι χρήσιμη στον αποκλεισμό των αναπνευστικών αιτιών, αν και η αναπνευστική και καρδιακή ασθένεια συνήθως συνυπάρχει . Το μέγιστο εκπνευστικό ποσοστό ροής και ο αναγκασμένος εκπνευστικός όγκος ανά λεπτό μειώνονται στη συγκοπή καρδιάς [3].

11.4 Το σύστημα συγκοπής καρδιάς σε UML

Είναι πολύ σημαντικό να δημιουργηθεί ένα σύστημα, το οποίο θα εξετάζει όλες τις καταστάσεις που έχουν να κάνουν, όταν ένας ασθενής έχει πιθανότητα να εμφανίσει συγκοπή καρδιάς. Δηλαδή τις ασθένειες και τα συμπτώματα, τις συμβουλές των γιατρών, τις εξετάσεις και τη διάγνωση , ακόμα και την εισαγωγή του ασθενή και τη χειρουργική επέμβαση. Το σύστημα αυτό θα υλοποιηθεί χρησιμοποιώντας αντικειμενοστρεφή σχεδίαση, επειδή έτσι οι έννοιες αποδίδονται εύστοχα. Το σύστημα αυτό μπορεί να εφαρμοστεί σε τμήμα νοσοκομείου που εξειδικεύεται στη συγκοπή καρδιάς. Θα μπορεί να συλλέγει πληροφορίες για στατιστικούς σκοπούς αλλά κυρίως για ερευνητικούς σκοπούς. Οι πελάτες του συστήματος μπορεί να είναι ο διευθυντής του τμήματος αυτού ή κάποιος διαχειριστής του συστήματος. Έτσι δημιουργήθηκε το επόμενο διάγραμμα κλάσεων.



Σχήμα 11.4.1 Απεικονίζεται το διάγραμμα κλάσεων του συστήματος

Παρακάτω γίνεται εξήγηση του προηγουμένου διαγράμματος κλάσεων.

Η συμβούλευση που λαμβάνει ο ασθενής είναι δυο ειδών. Τηλεφωνική συμβούλευση και συμβούλευση στο νοσοκομείο. Η κλάση Συμβούλευση (Consultation) χωρίζεται σε δυο κλάσεις με γενίκευση ή κληρονομικότητα (Generalization- Inheritance). Αυτές είναι η Τηλεφωνική (ByPhone) και Σε νοσοκομείο (InHospital). Επομένως, οι δυο αυτές κλάσεις είναι παιδιά της κλάσης Συμβούλεσης. Οι κλάσεις παιδιά έχουν όλα τα χαρακτηριστικά της κλάσης από την οποία προέρχονται, δηλαδή την Συμβούλευση και έχουν κάποια επιπλέον χαρακτηριστικά..

Το κάθε είδος συμβούλευσης οδηγεί σε ιατρική θεραπεία (Medical treatment) και σε οδηγίες για να μπορεί να βοηθήσει τον εαυτό του ο ίδιος ο ασθενής (Self help strategies). Δηλαδή υπάρχει συσχέτιση της Συμβούλευσης (Consultation) με τις κλάσεις Medical Treatment και Self Help Strategies με πολλαπλότητα (1-*) ένα προς πολλά και στις δυο. Αυτό γίνεται επειδή κάθε μια συμβούλευση οδηγεί σε μια ή περισσότερες ιατρικές θεραπείες και οδηγίες για να μπορεί να βοηθήσει τον εαυτό του ο ίδιος ο ασθενής.

Η ιατρική θεραπεία έχει πέντε συστατικά μέρη, τα οποία μπορούν να χρησιμοποιηθούν μαζί η χωριστά ή σε συνδυασμούς. Για αυτόν το λόγο ακριβώς η κλάση Medical Treatment συνδέεται με τις πέντε κλάσεις – τα πέντε μέρη με συνάθροιση (aggregation). Τα μέρη αυτά είναι η αναφορά (Referral) του ασθενή σε κάποιον ειδικό καρδιολόγο, η θεραπεία που θα ακολουθήσει ο ασθενής (Therapy), καθώς και η συνταγή για τα φάρμακα που θα παίρνει (Medication). Ακόμη, είναι κάποιες οδηγίες που δίνουν οι ιατροί και οι νοσοκόμες στους ασθενείς ανάλογα με την περίπτωση (Instructions). Τέλος, υπάρχουν και τα τεστ- εξετάσεις (Tests) που ίσως χρειαστεί να κάνει επιπλέον ο ασθενής.

Οι οδηγίες που δίνονται, έτσι ώστε να μπορέσει να βοηθηθεί μόνος του ο ασθενής, είναι συνήθως μέχρι πέντε. Αυτές, όπως και τα συστατικά της ιατρικής θεραπείας μπορούν να χρησιμοποιηθούν και σε συνδυασμούς. Επομένως, οι οδηγίες αφορούν την αντιβιοτική προφύλαξη του ασθενούς και ανοσοποίηση (Antibiotic precaution and Immunization), για να προστατέψει τον οργανισμό του, που πιθανώς να είναι ευαίσθητος , να κόψει το κάπνισμα (NoSmoking) και να μειώσει την κατανάλωση αλκοόλ (NoAlcohol). Επίσης, ο ασθενής πρέπει να ασκείται σωματικά (PhysicalExercise), όπως συστηματικό περπάτημα και να προσέχει την διατροφή του (Diet), η οποία πρέπει να περιέχει φρούτα και λαχανικά και όχι πολλά κρέατα και λιπαρά.

Το επαγγελματικό προσωπικό (Professionals) που υπάρχει είναι οι ιατροί (Cardiologist) και οι νοσοκόμες (Nurse) που είναι εκπαιδευμένες για καρδιολογικά θέματα. Επίσης η κλάση Professionals με τις κλάσεις Nurse και Cardiologist συνδέεται με κληρονομικότητα, αφού ανήκουν στην κλάση Professionals και κληρονομούν τα χαρακτηριστικά της.

Όταν ο ασθενής πηγαίνει στο νοσοκομείο ο ιατρός – καρδιολόγος, τον εξετάζει, με φυσική εξέταση (NaturalExamination), δηλαδή βλέπει την εμφάνισή του και ακούει τους χτύπους της καρδιάς του ασθενούς. Επίσης ρωτάει τον ασθενή για προηγούμενα περιστατικά που του έχουν συμβεί και τις συμβουλές που του είχαν δοθεί

(Anaemnesis). Συμβουλεύεται το ιστορικό του ασθενούς , χρησιμοποιώντας τον ιατρικό φάκελο του ασθενή. Γίνονται μια σειρά από εξετάσεις ανάλογα με την κάθε περίπτωση, έτσι ώστε να μπορέσει ο ιατρός να βγάλει σωστά συμπεράσματα και να κάνει σωστή διάγνωση (Diagnosis). Επομένως, ένας ασθενής μπορεί να έχει πολλές συμβουλεύσεις, όπως φαίνεται και στο σχήμα. Η κλάση Ασθενής (Patient) συσχετίζεται με την Consultation με πολλαπλότητα (1-*). Η κλάση InHospital συνδέεται με τις Anamnesis, Diagnosis και Natural Examination με συνάθροιση, καθώς και οι τρεις συμβάλλουν, ώστε να γίνει η συμβούλευση των ασθενών. Κάποιες φορές μπορεί να μη χρειαστεί κάποια από αυτές. Επίσης, οι πολλαπλότητες είναι 1-1, 1-0…* και 1-1 αντίστοιχα για κάθε μια από τις τρεις κλάσεις.

Η κλάση Διάγνωση (Diagnosis) είναι συνδεδεμένη με τις κλάσεις Chest Xray examination, Electrocardiography 12 lead, Echocardiography, Haematology and biochemistry, Radionuclide methods, Angiography cardiac catheterization and myocardial biopsy και Palmonary function tests με κληρονομικότητα- γενίκευση. Αυτό γίνεται επειδή, όλες αυτές οι κλάσεις που είναι συνδεδεμένες με την Diagnosis έχουν όχι μόνο τα ίδια χαρακτηριστικά που έχει αυτή, αλλά έχουν επιπλέον στοιχεία .

Σε περίπτωση που η κατάσταση του ασθενούς είναι σοβαρή, γίνεται εισαγωγή στο νοσοκομείο. Ακολουθείται η κατάλληλη θεραπεία . Αν όμως η κατάσταση είναι πολύ κρίσιμη , τότε ο ασθενής εισάγεται στο χειρουργείο. Η κλάση Import συνδέεται με την Surgery με συσχέτιση και πολλαπλότητα 1-1.

Η κλάση Illnesses συνδέεται με την κλάση Patient με συσχέτιση και πολλαπλότητα (0…*-1). Αυτό προκύπτει από το γεγονός ότι ένας ασθενής μπορεί να έχει από καμιά έως πολλές ασθένειες. Ακόμα, η κλάση Illnesses συνδέεται με την κλάση Symptoms με συσχέτιση και πολλαπλότητα (*-0…*). Αυτό συμβαίνει γιατί μια ασθένεια είναι πιθανόν να έχει διάφορα συμπτώματα ,από κανένα μέχρι πολλά. Επιπρόσθετα η κλάση Illnesses συνδέεται με τις κλάσεις Hypertension, Coronary artery disease, Arrythmias, Valvar disease, Alcohol and drugs και Cardiomyopathies. Η σύνδεση είναι κληρονομικότητα- γενίκευση αφού κάθε κλάση από αυτές ανήκει στις ασθένειες, αλλά είναι συγκεκριμένη ασθένεια με συγκεκριμένα χαρακτηριστικά.

Η κλάση Cardomyopathies συνδέεται με κληρονομικότητα με τις κλάσεις Dilated, Hypertrophic, Obliterative and restrictive αφού κάθε κλάση από αυτές ανήκει στις καρδιομυοπάθειες, αλλά προσδιορίζει μια συγκεκριμένη καρδιομυοπάθεια κάθε φορά. Για τον ίδιο λόγο η κλάση Symptoms συνδέεται με τις κλάσεις Cough, Increase of weight, Anorexia, Oedema, Tiredeness and lethargy, Dyspnea και Difficulties in sleep με κληρονομικότητα.

11.5 Ερμηνεία των προτύπων που χρησιμοποιήθηκαν στο σχήμα.

Το σύστημα καθώς αυξάνεται, τροποποιείται και εξελίσσεται και αυξάνεται ο αριθμός των κλάσεων. Επομένως αυξάνεται η πολυπλοκότητα και είναι δύσκολο να γίνει κατανοητό το σύστημα. Για να γίνει πιο απλό το σύστημα χρησιμοποιείται το πρότυπο Πρόσοψης (Façade Pattern). Οι Πελάτες (Clients) αντί να αλληλεπιδρούν με την κλάση Επαγγελματίες (Professionals) και την κλάση Ασθενής (Patient), δημιουργείται η διεπαφή Πρόσοψη (Facade). Η Πρόσοψη αλληλεπιδρά αυτή με τις κλάσεις Επαγγελματίες και Ασθενής. Επομένως οι Πελάτες αλληλεπιδρούν έμμεσα

με τους Επαγγελματίες και τον Ασθενή μέσω της Πρόσοψης. Η Πρόσοψη εκπροσωπεί τα αιτήματα του Πελάτη στα κατάλληλα αντικείμενα των κλάσεων υποσυστήματος.

Η Αφηρημένη Υπέρ-κλάση Συμπτώματα - Symptoms που χρησιμοποιείται, ορίζει τη διεπαφή που μοιράζονται οι κλάσεις Βήχας (Cough), Κούραση και Λήθαργος (Tiredness and Lethargy), Οίδημα (Oedema), Ανορεξία (Anorexia), Αύξηση Βάρους (Increase of Weight), Δυσκολίες Ύπνου (Difficulties in Sleep), Δύσπνοια (Dyspnea). Αυτές οι κλάσεις κληρονομούν και ορισμένα χαρακτηριστικά από την Αφηρημένη Υπέρ-κλάση τους. Ορισμένες συναρτήσεις- μέθοδοι είναι κοινές και για την Αφηρημένη Υπέρ-κλάση και για τις υπόλοιπες κλάσεις και καταγράφονται στην Αφηρημένη Υπέρ-κλάση. Κάποιες άλλες συναρτήσεις είναι διαφορετικές για τις υπόλοιπες κλάσεις. Οι συναρτήσεις αυτές θα υλοποιηθούν σε αυτές τις κλάσεις, αλλά η κάθε μια θα δώσει το δικό της αποτέλεσμα. Στην Αφηρημένη Υπέρ-κλάση καταγράφονται ως αφηρημένες συναρτήσεις.
Αφηρημένη Υπέρ-κλάση χρησιμοποιείται και στις ασθένειες. Δημιουργείται δηλαδή η Αφηρημένη Υπέρ-κλάση Ασθένειες –Illnesses που τη μοιράζονται οι κλάσεις Υπέρταση (Hypertension), Στεφανιαίες καρδιακές παθήσεις (Coronary artery disease), Arrythmias (Αρρυθμία), Ασθένειες βαλβίδων (Valvar disease), Alcohol and drugs (Αλκοόλ και ναρκωτικά), Καρδιομυοπάθειες (Cardiomyopathies).
Αν είναι επιθυμητό η αφηρημένη Υπέρ-κλάση μπορεί να εφαρμοστεί και στην κλάση Καρδιομυοπάθειες (Cardiomyopathies) και να οριστούν σαν υποκλάσεις της οι κλάσεις Διεσταλμένη (Dilated), Υπερτροφική (Hypertrophic), Περιοριστική (Restrictive) και Εξαλειπτική (Obliterative).

Η κλάση Ασθενής (Patient) δεν πρέπει να τροποποιείται από την κλάση Νοσοκόμα (Nurse), ενώ επιτρέπεται να τροποποιηθεί από την κλάση Καρδιολόγος (Cardiologist). Για αυτόν το λόγο χρησιμοποιείται το πρότυπο Read-Only-Interface. Δημιουργείται η διεπαφή Ασθενής Μόνο Ανάγνωση (Read Only Patient). Έτσι παρέχεται πρόσβαση ανάγνωσης μόνο της κλάσης Νοσοκόμα στο αντικείμενο Ασθενής μέσω της διεπαφής αυτής που δεν περιλαμβάνει μεθόδους για το αντικείμενο. Η κλάση Ασθενής εφαρμόζει τη διεπαφή Ασθενής Μόνο Ανάγνωση, η οποία δεν περιλαμβάνει μεθόδους που θα ανάγκαζαν ένα αντικείμενο Ασθενή να τροποποιήσει το περιεχόμενό του.
Η κλάση Καρδιολόγος χρησιμοποιεί την κλάση Ασθενής άμεσα και μπορεί να καλέσει τις μεθόδους που τροποποιούν την κατάσταση ενός Ασθενή. Η κλάση Νοσοκόμα έχει πρόσβαση στην κλάση Ασθενής μέσω της διεπαφής Ασθενής μόνο Ανάγνωση. Δεν είναι ικανή να έχει πρόσβαση στις μεθόδους που τροποποιούν τα αντικείμενα του Ασθενή.

Είναι πολύ πιθανό ο ασθενής στο Νοσοκομείο να μη χρειαστεί να κάνει τις εξετάσεις για διάγνωση. Αυτό σημαίνει πως οι υποκλάσεις της κλάσης Διάγνωση μπορεί να μη χρειαστούν ποτέ ή να χρειαστούν μετά από ένα χρονικό διάστημα. Οι υποκλάσεις αυτές είναι οι Θωρακική των ακτινών Χ εξέταση (Chest Xray Examination), Ηλεκτροκαρδιογράφημα 12 σημείων (Electrocardiography 12lead), Ηχοκαρδιογραφία (Echocardiography), Αιματολογικές και οι Βιοχημικές εξετάσεις (Haematology and biochemistry), Απεικόνιση των ραδιονουκλειδίων (Radionuclide methods), Μέτρηση της λειτουργίας πνευμόνων (Palmonary function tests), Αγγειογραφία καρδιακός καθετηριασμός και βιοψία μυοκαρδίου (Angiography Cardiac catheterization and myocardial biopsy). Το πρότυπο Εικονικού Πληρεξουσίου κρύβει από τους Πελάτες

του, το γεγονός ότι ένα αντικείμενο μπορεί να μην υπάρχει ακόμα, έχοντας πρόσβαση στο αντικείμενο έμμεσα, μέσω ενός αντικειμένου ΠληρεξούσιοΔιάγνωσης (ProxyDiagnosis). Το αντικείμενο αυτό εφαρμόζει την ίδια διεπαφή Διάγνωση – Diagnosis με το αντικείμενο που μπορεί να μην υπάρχει. Εδώ γίνεται προσπάθεια να αποφευχθεί η δημιουργία του αντικειμένου για όσο το δυνατό περισσότερο χρονικό διάστημα. Όταν πρέπει να δημιουργηθεί ένα στιγμιότυπο Diagnosis, τότε η κλάση Diagnosis δημιουργεί ό,τι χρειάζεται. Το πληρεξούσιοΔιάγνωσης – ProxyDiagnosis καθυστερεί τη δημιουργία των αντικειμένων της κλάσης Diagnosis έως ότου απαιτούνται πραγματικά. Το πληρεξούσιο αυτό παρέχει εμμεσότητα μεταξύ των κλάσεων που χρειάζονται τη κλάση Diagnosis. Ένα αντικείμενο ProxyDiagnosis είναι αρμόδιο για τη δημιουργία του αντίστοιχου αντικειμένου Diagnosis.

Το σημείο εισαγωγής (Import) του ασθενή στο Νοσοκομείο είναι ιδιαίτερης σημασίας. Πρέπει να υπάρχει ένα στιγμιότυπο της κλάσης Import. Δεν είναι επιθυμητό να υπάρχουν δύο ταυτόχρονα στιγμιότυπα της κλάσης αυτής. Ο ασθενής όταν έχει μια εισαγωγή δεν μπορεί να έχει και δεύτερη. Πρέπει να εκτελεστεί η πρώτη και μετά αν χρειαστεί να γίνει και η επόμενη. Για αυτό χρησιμοποιείται το πρότυπο Singleton. Το πρότυπο αυτό αποτελείται από μια κλάση την SingletonImport, η οποία είναι υπεύθυνη να δημιουργεί το μοναδικό της στιγμιότυπο. Είναι στατική για αυτό το λόγο. Η μέθοδος στιγμιότυπο - instance επιστρέφει ένα μοναδικό στιγμιότυπο όταν ζητηθεί , ενώ η πρόσβαση στην κλάση γίνεται μέσω της μεθόδου λειτουργία Singleton- Singletonfunction.

Το σύστημα που παρουσιάζεται εδώ εφαρμόζει αντικειμενοστρεφή σχεδίαση και μπορεί να χρησιμοποιηθεί σε τμήμα νοσοκομείου που ειδικεύεται στην συγκοπή καρδιάς. Οι πελάτες του συστήματος μπορεί να είναι ο διευθυντής ή ο διαχειριστής του συστήματος. Το σύστημα μπορεί να χρησιμοποιηθεί για να συγκεντρώσει πληροφορίες για στατιστικούς σκοπούς, αλλά ο κύριος στόχος του είναι για ιατρικούς ερευνητικούς σκοπούς.

Αναφορές:

[1]: CHRONIC HEART FAILURE National clinical guideline for diagnosis and management in primary and secondary care Full version of NICE Guideline No. 5 Developed by The National Collaborating Centre for Chronic Conditions 2003.

[2]: E. Villalba, M.T. Arredondo, M. Ottaviano, D. Salvi, E. Hoyo-Barbolla ,*Heart Failure monitoring system based on Wearable and Information Technologies* ,Life Supporting Technologies, Technical University of Madrid, Spain from JOURNAL OF COMMUNICATIONS, VOL. 2, NO. 2, MARCH 2007.

[3]: R C Davis, F D R Hobbs, G Y H Lip *,(2000), ABC of heart failure History and epidemiology* .

[4]: http://en.wikipedia.org/wiki/Heart_failure

[5]: http://en.wikipedia.org/wiki/Image:12leadECG.jpg

[6]: http://en.wikipedia.org/wiki/Echocardiography

[7]: Δρ Α. Πολυνείκης, *ΕΘΝΙΚΟ ΣΥΣΤΗΜΑ ΥΓΕΙΑΣ.*

[8]: *Κεφάλαιο 1 Εισαγωγή στην Ιατρική Πληροφορική.* Ιατρική Σχολή, Α.Π.Θ.,Εργαστήριο Ιατρικής Πληροφορικής Ιατρική Πληροφορική Ι, 2006.

[9]: Douglas Schmidt, (1995), *Using Design Patterns to Develop Reusable Object-Oriented Communication Software*. COMMUNICATIONS OF THE ACM October 1995/Vol. 38, No. 10, pp. 65-74.

[10]: Βασίλης Χ. Γερογιάννης. *Διαχείριση Κινδύνων*, Πρακτικά 17ου Συνεδρίου Ε.Ε.Ε.Ε. Πανεπιστήμιο Πατρών, 16-18 Ιουνίου 2005, Σελ. 391-402 Αντικειμενοστραφής μοντελοποίηση των διαδικασιών της ανάπτυξης λογισμικού.

[11]: Yannis Tzitzikas, (2005). Slides: HY 351: *Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων CS 351: Information Systems Analysis and Design* Lecture : 8 Date : 20-10-2005 Functional Modeling University of Crete, Fall 2005.
[12]: Yannis Tzitzikas, (2005). *Introduction to Object-Oriented Analysis and Design*, CS 351: Information Systems Analysis and Design.
[13]: Wolfgang Pree, Hermann Sikora, (1997), *Design Patterns for Object-Oriented Software Development*, International Conference on Software Engineering, Proceedings of the 19th international conference on Software engineering, Boston, Massachusetts, United States 1997, pp. 663 – 664.
[14]: Oleg Sokolsky, (2005), *Enhancing Dependability of Medical Software Systems*, High Confidence Medical Device Software and Systems (HCMDSS) Workshop , June 2-3, 2005 Philadelphia,PA.

12. ΣΥΜΠΕΡΑΣΜΑΤΑ

Για τον σχεδιασμό αντικειμενοστρεφών λογισμικών συστημάτων μπορεί να ακολουθηθεί μεθοδολογία υπαγαρορευομένη από πρότυπα σχεδιασμού τα οποία έχουν προκύψει ως αποδεδειγμένες λύσεις σε επαναληπτικά εμφανιζόμενα σχεδιαστικά πρόβλήματα.
Τα πρότυπα σχεδιασμού λογισμικού μπορούν να χωριστούν σε κατηγορίες. Οι δυο κύριες κατηγορίες είναι τα Αρχιτεκτονικά και τα Σχεδιαστικά. Τα Σχεδιαστικά πρότυπα χωρίζονται περαιτέρω σε υποκατηγορίες, οι κυριώτερες των οποίων είναι: Θεμελιώδη, Δημιουργικά, Δομικά, Συμπεριφοράς, Διαμέρισης και Ταυτοχρονισμού.
Ο συνδιασμός γνωστών προτύπων δημιουργεί σύνθετα πρότυπα που μπορούν να αποτελέσουν σχεδιαστικές λύσεις για σύνθετα προβλήματα.
Στην παρούσα εργασία, αναζητήθηκαν συνδιασμοί προτύπων για την επίλυση προβλημάτων σχεδίασης συστημάτων λογισμικού υψηλών επιδόσεων.
Επίσης η μελετηθείσα σε αυτήν την εργασία μεθοδολογία προτύπων σχεδιασμού, εφαρμόστηκε στον σχεδιασμό προτύπου Ιατρικού Πληροφορικού συστήματος που αφορά στην διάγνωση και στην θεραπεία καρδιακών παθήσεων.