ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Αυτόματη διαπραγμάτευση συμβολαίων με διεπαφή την φυσική γλώσσα πάνω σε υπηρεσίες διεπαφών του σημασιολογικού ιστού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΠΕΤΥΧΑΚΗ ΒΑΣΙΛΙΚΗ-ΑΓΓΕΛΙΚΗ

Επιβλέπων : Ασκούνης Δημήτριος
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβρης 2017

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Αυτόματη διαπραγμάτευση συμβολαίων με διεπαφή την φυσική γλώσσα πάνω σε υπηρεσίες διεπαφών του σημασιολογικού ιστού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΠΕΤΥΧΑΚΗ ΒΑΣΙΛΙΚΗ-ΑΓΓΕΛΙΚΗ

Επιβλέπων : Ασκούνης Δημήτρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Οκτωβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Ασκούνης Δημήτριος	Ψαράς Ιωάννης	Δούκας Χαράλαμπος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβρης 2017

(Υπογραφή)

.....

ΠΕΤΥΧΑΚΗ ΒΑΣΙΛΙΚΗ-ΑΓΓΕΛΙΚΗ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας εργασίας είναι η ένταξη της δυνατότητας διαπραγμάτευσης σε αυτοματοποιημένους πράκτορες που δραστηριοποιούνται στα πλαίσια του DeepGraphs specification.

Η διαπραγμάτευση είναι μία φυσική δραστηριότητα για τους ανθρώπους στην οποία εμπλέκονται καθημερινά όταν καλούνται να διεκδικήσουν κάτι. Συναντάται από τα πιο ασήμαντα σενάρια, όπως το παζάρι για μία τιμή, έως τα πιο σημαντικά, όπως οι διπλωματικές συναντήσεις κρατών. Ένα τόσο συχνό φαινόμενο δεν θα ήταν λογικό να απουσιάζει τελείως από τον ηλεκτρονικό χώρο του internet που, ειδικά τα τελευταία χρόνια, έχει γνωρίσει τεράστια ανάπτυξη και οι άνθρωποι αντικαθιστούν πολλά από τα πράγματα που προηγουμένως κάνανε διαπροσωπικά και χειροκίνητα με αντίστοιχες ηλεκτρονικές υπηρεσίες. Παρά τις προσπάθειες που έχουν γίνει κατά καιρούς για αυτοματοποιημένη διαπραγμάτευση, καμία δεν είχε ευρεία αποδοχή στο χώρο του διαδικτύου, και αυτό γιατί η επικοινωνία μεταξύ των συστημάτων είναι περίπλοκη. Κάθε ιστοσελίδα είναι χτισμένη με διαφορετική αρχιτεκτονική, συνδυάζοντας διαφορετικές τεχνολογίες. Για να επιτύχουμε λοιπόν μία καθολική λύση στην διαπραγμάτευση επιλέξαμε το DeepGraphs specification σαν το πρωτόκολλο που θα αποτελέσει την κοινή βάση επικοινωνίας των υπηρεσιών λογισμικού στο διαδίκτυο. Το πρότυπο αυτό αποτελεί μία επέκταση του Hydra specification και χρησιμοποιεί τεχνολογίες και πρωτόκολλα του internet, και κυρίως του σημασιολογικού ιστού, όπως το Shema.org, SWRL, REST, και άλλες αντίστοιχες, εξίσου ώριμες και ευρέως χρησιμοποιούμενες.

Στα πλαίσια της παρούσας εργασίας επιχειρήθηκε και μία υλοποίηση, ένα παράδειγμα που συνοδεύει την μελέτη που έγινε σαν μία μορφή επικύρωσης. Το συγκεκριμένο παράδειγμα είναι διαθέσιμο στην κοινότητα ανοιχτού λογισμικού και είναι μία συμβολή σε όσους ασχολούνται ή σκοπεύουν να ασχοληθούν μελλοντικά προς αυτή τη κατεύθυνση.

Λέξεις Κλειδιά: REST, API, Hydra Specification, DeepGraphs Specification, Semantic Web, Schema.org, Negotiation, Autonomous Agent

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The purpose of this thesis is the embodiment of the ability to negotiate to automated agents that act in the context of DeepGraphs specification.

Negotiation is a natural activity for humans in which they get involved daily when they find themselves in a situation where they have to claim something. It is encountered in the most meaningless of the scenarios, such as haggling over a price, to them most important, such as diplomatic meeting between states. Such a common phenomenon wouldn't be reasonable to be completely absent from the digital world of the internet which, especially the past recent years, has met a huge growth and people replace many of the thing they used to do in person and manually with counterpart digital services. Many have tried in the past to implement an automated negotiation process, but none of them had a broad acceptance, because the communication between the various systems is complex. Every website is built using different architecture, combining plenty of technologies. To succeed on creating a global solution for negotiation, we chose the DeepGraphs specification as the protocol which will be the common base of communication between software services in the web. This specification is an expansion of Hydra specification and uses many technologies and protocols, mainly form the semantic web, such as Shema.org, SWRL, REST, and more, similarly mature and broadly used.

In the context of this thesis we tried to also create an implementation, an example that will accompany the study as a form of validation. This project is available for the open source community and it's a contribution to anyone that is working or is planning to work in this direction.

Keywords:REST, API, Hydra Specification, DeepGraphs Specification, Semantic Web, Schema.org, Negotiation, Autonomous Agent

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΥΧΑΡΙΣΤΙΕΣ

Υπεύθυνος κατά την εκπόνηση της διπλωματικής ήταν ο καθηγητής κ. Ασκούνης Δημήτριος στον οποίο οφείλω ιδιαίτερες ευχαριστίες τόσο για την ανάθεση της διπλωματικής όσο και για την δυνατότητα που μου παρείχε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα μέσα από το οποίο εμπλούτισα τις τεχνικές μου γνώσεις. Ακόμη θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα κ.Πετυχάκη Μιχάλη για την βοήθεια και την υποστήριξη που μου δόθηκε καθόλη τη διάρκεια αλλά και κατά την συγγραφή της εργασίας.

Τέλος θα ήταν παράλειψη να μην ευχαριστήσω όλους τους δικούς μου ανθρώπους για την αμέριστη υποστήριξη και την κατανόηση που επέδειξαν σε όλη την πορεία μου.

Πίνακας περιεχομένων

1

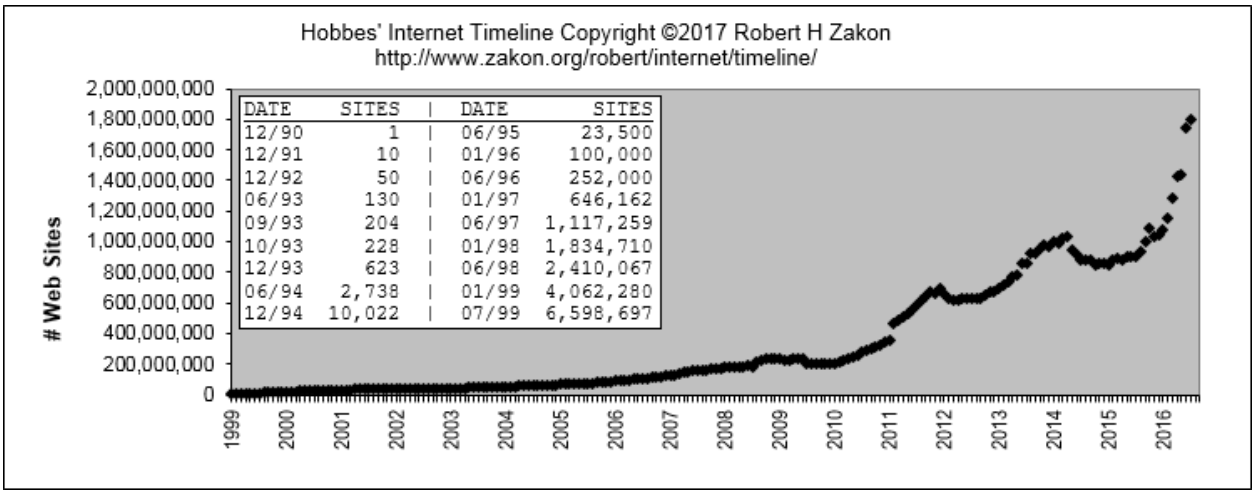
Εισαγωγή

1.1 Περιγραφή του προβλήματος και συνεισφορά

Κοιτάζοντας την πορεία της τεχνολογίας, και πιο συγκεκριμένα του παγκόσμιου ιστού, των τελευταίων χρόνων είναι προφανής μία συνεχόμενη ανοδική τάση, τόσο σε ό,τι αφορά την εξάπλωση (αριθμός χρηστών) όσο και στο μέγεθος (αριθμός ιστοσελίδων). Από την εμφάνιση του διαδικτύου κοντά δύο δεκαετίες πριν από τον εμπνευστή και εφευρέτη του Tim Berners-Lee μέχρι σήμερα ο κόσμος έχει αλλάξει τελείως μορφή. Παρά την μικρή πορεία του στον χρόνο, έχει καταφέρει να παγιωθεί στις συνειδήσεις των ανθρώπων και να μπει για τα καλά στην καθημερινότητά τους. Πολλοί λόγοι υπάρχουν που αιτιολογούν την τεράστια - εκρηκτική σχεδόν- επέκταση αυτής της τεχνολογίας. Σαν παράδειγμα μπορούμε να αναφέρουμε την εύκολη πρόσβαση, καθώς δεν απαιτείται εξαιρετικά περίπλοκος ή ακριβός εξοπλισμός για να μπει ένας χρήστης στο Internet. Ένα άλλο παράδειγμα είναι η ευκολία στην χρήση του. Δεν απαιτούνται ιδιαίτερες τεχνικές γνώσεις για να μπορέσει κανένας να χρησιμοποιήσει το Internet και συνολικά η σχεδίαση τόσο των εργαλείων για την πρόσβαση σε αυτό όσο και οι ίδιες οι σελίδες είναι συνήθως προαισθηματική.

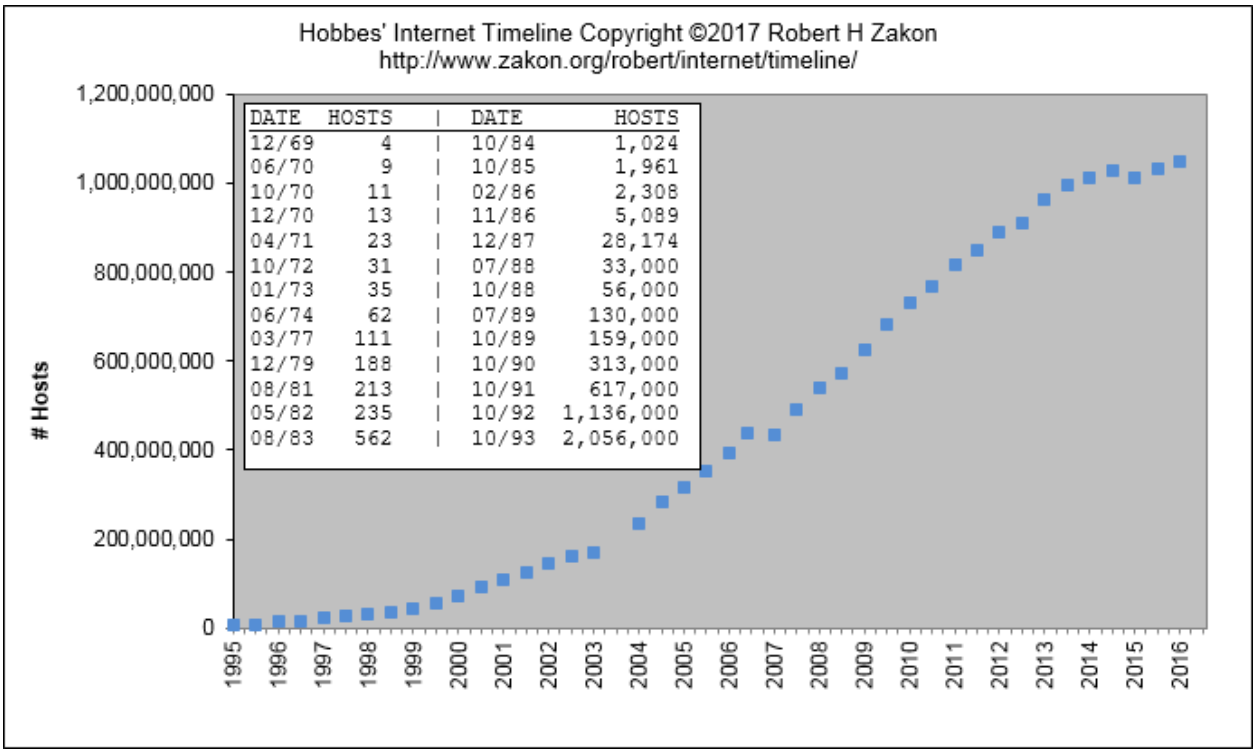
Για να έχουμε μία αίσθηση των αριθμών, όταν κάνουμε λόγο για ανάπτυξη και εξάπλωση του Internet, παρακάτω παρατίθενται 2 χρονοδιαγράμματα. Στο πρώτο απεικονίζεται ο αριθμός των hosts, των υπολογιστών δηλαδή που έχουν μία valid ip address, ενώ στο δεύτερο είναι ο αριθμός των websites από την δεκαετία του ’90 έως σήμερα (το σημαντικό χάσμα ανάμεσα

στα 2 διαγράμματα οφείλεται στο ότι ένα hosting computer μπορεί να έχει παραπάνω από μία σελίδες). Αξίζει να σημειωθεί ότι ο αριθμός των websites κοντεύει τα 2 δισεκατομμύρια ήδη το 2016. Ενδιαφέρον ακόμα παρουσιάζει η ακόλουθη σελίδα, όπου φαίνεται η εξέλιξη του internet σε πραγματικό χρόνο http://www.internetlivestats.com/



DATE	SITES	DATE	SITES
12/90	1	06/95	23,500
12/91	10	01/96	100,000
12/92	50	06/96	252,000
06/93	130	01/97	646,162
09/93	204	06/97	1,117,259
10/93	228	01/98	1,834,710
12/93	623	06/98	2,410,067
06/94	2,738	01/99	4,062,280
12/94	10,022	07/99	6,598,697

Figure 1 Source - https://www.zakon.org/robert/internet/timeline/



DATE	HOSTS	DATE	HOSTS
12/69	4	10/84	1,024
06/70	9	10/85	1,961
10/70	11	02/86	2,308
12/70	13	11/86	5,089
04/71	23	12/87	28,174
10/72	31	07/88	33,000
01/73	35	10/88	56,000
06/74	62	07/89	130,000
03/77	111	10/89	159,000
12/79	188	10/90	313,000
08/81	213	10/91	617,000
05/82	235	10/92	1,136,000
08/83	562	10/93	2,056,000

Figure 2 Source - https://www.zakon.org/robert/internet/timeline/

Ως φυσικό επακόλουθο του μεγέθους του Internet σήμερα, ο ανταγωνισμός έχει αυξηθεί πολύ. Πλέον για να προσεγγίζεις τους πολυάριθμους χρήστες πρέπει να ξεχωρίσεις, να δώσεις δηλαδή κάτι που προηγουμένως δεν υπήρχε. Με την λογική αυτή οι σελίδες άρχισαν σιγά σιγά να γίνονται ολοένα και πιο εντυπωσιακές για να τραβήξουν την προσοχή. Στη συνέχεια άρχισαν, εκτός από το στοιχείο του εντυπωσιασμού, να επικεντρώνονται και σε άλλους

παράγοντες, όπως η ευχρηστία των σελίδων ή των εφαρμογών. Στα πλαίσια αυτά άρχισαν να χτίζονται λογισμικά που είναι όσο γίνεται πιο απλά και ταυτόχρονα περιεκτικά στον χρήστη. Αρχίσαμε δηλαδή να παρατηρούμε την εμφάνιση σελίδων που μέσα από μία και μόνο οθόνη να μπορεί ο χρήστης με το πάτημα ενός κουμπιού να κάνει πιθανώς και παραπάνω από μία δουλειές. Ένα πολύ οικείο παράδειγμα είναι εκείνο ενός ηλεκτρονικού καταστήματος, όπου πλέον από μία οθόνη μπορούσες να επιλέξεις προϊόντα -που πιθανώς ανήκαν σε διαφορετικά καταστήματα ή αποθήκες-, να τα αγοράσεις κάνοντας χρήση της κάρτας σου και να επιλέξεις πως να αποσταλούν. Επιπλέον επιλογές μπορεί να ήταν διαθέσιμες ανάλογα με το είδος του αντικειμένου που αγοράζεται, όπως για παράδειγμα ασφάλιση ή εγγύηση, κλπ. Όλα αυτά εμπλέκουν πολλές υπηρεσίες που στον τελικό χρήστη δεν είναι άμεσα ορατές, ωστόσο έχουν βελτιώσει κατά πολύ την εμπειρία του στο internet και έχουν αυξήσει τις απαιτήσεις του.

Όλα τα παραπάνω που περιγράψαμε δεν θα μπορούσαν να υλοποιηθούν χωρίς την βοήθεια των REST APIs, μία τεχνολογία που αναπτύχθηκε σχετικά πρόσφατα και δίνει ένα ικανοποιητικό επίπεδο αφαίρεσης έτσι ώστε να μπορεί να εφαρμοστεί ευρύτατα χωρίς προβλήματα. Με την χρήση αυτών έγινε δυνατή η εμφώλευση νέων χαρακτηριστικών και δυνατοτήτων που προηγουμένως δεν ήταν δυνατόν να συμβούν αυτόματα και από μία μόνο εφαρμογή ή σελίδα. Επεκτείνοντας αυτό και στα πλαίσια του να εξυπηρετήσουμε τον χρήστη ακόμα περισσότερο, αναπτύσσονται νέες τεχνολογίες που του δίνουν πλέον την δυνατότητα να μπορεί να επικοινωνεί με τις συσκευές του μέσω φωνής και με φυσική γλώσσα.

Στην παρούσα εργασία λοιπόν γίνεται λόγος για το DeepGraphs specification, ένα πρωτόκολλο που θα μπορέσει να εξυπηρετήσει στην ανάπτυξη τέτοιων τεχνολογιών και να βοηθήσει πιθανώς στην απλοποίηση της εμπειρίας του χρήστη στα πλαίσια του internet ακόμα περισσότερο. Πιο συγκεκριμένα, όπως θα δούμε και παρακάτω, το DeepGraphs πρωτόκολλο μπορεί να εξυπηρετήσει στην αδιάλειπτη επικοινωνία διαφόρων υπηρεσιών με αυτοματοποιημένο τρόπο έτσι ώστε η μεσολάβηση του χρήστη για την ολοκλήρωση μίας διαδικασίας να μειωθεί ακόμα περισσότερο. Μέσα σε ένα χάος από υπηρεσίες που καθένα μιλάει την δική του γλώσσα, το DeepGraphs έρχεται να όχι να μεταγλωττίσει από τη μία γλώσσα στην άλλη, αλλά να θέσει κανόνες για το πως η επικοινωνία και η πληροφορία θα πρέπει να είναι δομημένη συνολικά έτσι ώστε να μπορεί να γίνει κατανοητή και επεξεργάσιμη από όλους. Με τον τρόπο αυτό, επομένως, όχι μόνο διευκολύνεται ο χρήστης, αλλά ταυτόχρονα διευκολύνεται και η επικοινωνία μεταξύ των αυτοματοποιημένων πρακτόρων (ή αλλιώς υπηρεσιών του διαδικτύου) καθώς πλέον έχουν μία κοινή βάση να πατήσουν και να «καταλάβουν» ο ένας τον άλλο.

Στα πλαίσια αυτού του προτύπου, οι αυτοματοποιημένοι και ανεξάρτητοι πράκτορες μπορούν πλέον να εκτελούν ενέργειες που μέχρι τώρα παραδοσιακά πραγματοποιούνταν αποκλειστικά

από χρήστες, Μία από αυτές τις δραστηριότητες που θα περιγράψουμε αναλυτικά είναι η διαπραγμάτευση. Στην πορεία θα δούμε εφαρμογές που η διαπραγμάτευση αυτή έχει νόημα αλλά και ένα παράδειγμα υλοποίησης αυτοματοποιημένης διαπραγμάτευσης ενός πράκτορα που ενεργεί στο DeepGraphs.

1.2 Οργάνωση κειμένου

Στο κεφάλαιο 2 γίνεται μία σύντομη εισαγωγή στην διαδικασία της διαπραγμάτευσης. Αναλύονται κάποιες δυσκολίες και προβληματικές που προκύπτουν στην ανθρώπινη διαπραγμάτευση και πως αυτές πιθανώς θα μπορέσουν να λυθούν στα πλαίσια μίας αυτοματοποιημένης διαπραγμάτευσης ανάμεσα σε τεχνητούς πράκτορες. Αναφέρονται ακόμα κάποιες σχετικές μελέτες που είχαν γίνει στο παρελθόν άλλες εκτενώς και άλλες συνοπτικότερες.

Στο κεφάλαιο 3 αναλύονται ικανοποιητικά οι βασικές έννοιες και τεχνολογίες που χρησιμοποιήθηκαν έτσι ώστε να μην υπάρχουν ασάφειες και κενά στον αναγνώστη. Θα γίνει εκτενής αναφορά σε web τεχνολογίες όπως τα APIs και κυρίως τα REST APIs, ενώ ταυτόχρονα θα συζητηθεί η έννοια του semantic web και τι μπορεί αυτό να προσφέρει, άμεσα ή μελλοντικά.

Στο κεφάλαιο 4 θα συνεχίσουμε να παραθέτουμε χρήσιμες ορολογίες και έννοιες που θα μας βοηθήσουν στην συνέχεια (κεφάλαιο 5) να κατανοήσουμε πλήρως την μεθοδολογία της παρούσας εργασίας. Στο κεφάλαιο αυτό λοιπόν γίνεται λόγος για το DeepGraph specification που είναι και ο βασικός κορμός πάνω στον οποίο έχει χτιστεί η εργασία αυτή. Αρχικά αναλύονται οι λόγοι που οδήγησαν ή/και ευνόησαν την ανάπτυξη ενός τέτοιου πρωτοκόλλου και στην συνέχεια περιγράφεται ο τρόπος λειτουργίας του. Για την καλύτερη και πιο ολοκληρωμένη κατανόηση του DeepGraphs specification, δίνονται και άλλοι ορισμοί, όπως για το Hydra specification και το Schema.org, τεχνολογίες που χρησιμοποιεί κατά κόρον.

Στο κεφάλαιο 5 και εφόσον έχουμε ήδη χτίσει το θεωρητικό υπόβαθρο παρουσιάζεται η μεθοδολογία που ακολουθήθηκε στην παρούσα εργασία αναλυτικά, μαζί με τους λόγους που επιλέχθηκε αλλά και τους περιορισμούς που τέθηκαν. Εδώ παρουσιάζεται και εξηγείται ακόμα με λεπτομέρεια ο αλγόριθμος που χρησιμοποιήθηκε, χωρίς όμως να θίξουμε θέματα υλοποίησης, και πως θα μπορούσε να επεκταθεί.

Στο κεφάλαιο 6 παρουσιάζεται η υλοποίηση του αλγορίθμου που περιγράφηκε αναλυτικά στο κεφάλαιο 5. Δίνεται αρχικά το πλαίσιο στο οποίο ο agent που δημιουργήσαμε λειτουργεί, στη συνέχεια κάποιες σύντομες εξηγήσεις για το πως ο agent λειτουργεί και πως πραγματώνει το DeepGraphs πρότυπο και τέλος παρατίθενται τα αποτελέσματα που προέκυψαν. Προτείνονται ακόμα πιθανές αλλαγές και προεκτάσεις πάνω στην συγκεκριμένη υλοποίηση,

κάποιες από τις οποίες είναι εύκολα υλοποιήσιμες και κάποιες άλλες που θα απαιτούσαν εκτενή μελέτη και κατανόηση του χώρου για να πραγματοποιηθούν.

Στο κεφάλαιο 7 που αποτελεί και το τελευταίο της διπλωματικής εργασίας γίνεται μία σύνοψη της δουλειάς που έγινε και μία επισκόπηση και αξιολόγηση των αποτελεσμάτων. Ακόμα παρουσιάζονται συγκεντρωμένες όλες οι προτάσεις επέκτασης της παρούσας εργασίας που αναφέρθηκαν διάσπαρτα σε διάφορα σημεία του κειμένου. Τέλος γίνεται μία προσπάθεια για απολογισμό τόσο της υλοποίησης όσο και του DeepGrpahs specification.

2

Ανάλυση διαπραγμευτικής διαδικασίας

Στο κεφάλαιο αυτό, γίνεται μία προσπάθεια αποδόμησης της διαπραγματευτικής διαδικασίας σε διακριτά βήματα με σκοπό την αποτύπωση όλων των εκφάνσεων μίας τέτοιας δραστηριότητας. Απώτερος στόχος είναι μετά από αυτή τη μελέτη να γίνει φανερό ποια στάδια είναι χρήσιμα για το δικό μας σενάριο και να τα ενσωματώσουμε στο μοντέλο αυτοματοποιημένου πράκτορα.

2.1 Εισαγωγή

Η διαπραγμάτευση αποτελεί αναπόσπαστο κομμάτι της καθημερινότητάς μας. Διαπραγματευόμαστε όταν αγοράζουμε κάτι για να επιτύχουμε καλύτερη τιμή, σε μία συνέντευξη για δουλειά όταν τίθενται το θέμα του μισθού, στο σπίτι ή στην δουλειά όταν πρέπει να γίνει κάποια εργασία που κανένας δεν θέλει να περατώσει, και ούτω καθεξής. Ψάχνοντας ακόμα περισσότερο μπορούμε να βρούμε πολλά ακόμα παραδείγματα απλών καθημερινών διαπραγματεύσεων.

Το ερώτημα είναι γιατί μπαίνουμε σε αυτή την διαδικασία τόσο συχνά? Η απάντηση είναι πολύ απλή. Η διαπραγμάτευση είναι η διαδικασία που θα βοηθήσει ένα άτομο να διεκδικήσει έναν επιθυμητό στόχο. Φυσικά αν αυτός ο στόχος δεν διεκδικείται από κανέναν άλλο ή/και δεν θέτει σε κίνδυνο τα συμφέροντα κάποιου άλλου, τότε φυσικά δεν υπάρχει λόγος για διαπραγμάτευση. Όμως πολύ συχνά τα συμφέροντα του ενός έρχονται σε αντίθεση με τις

ανάγκες και επιθυμίες του άλλου. Σε αυτή την περίπτωση τα δύο μέρη έρχονται σε μία μορφή αντιπαράθεσης ή σύγκρουσης γιατί καθένας θέλει να επιτύχει το βέλτιστο για εκείνον αποτέλεσμα. Η διαπραγμάτευση λοιπόν είναι ένας μηχανισμός για να καταλήξουν αυτά τα αντίπαλα μέρη σε μία συμφωνία αποδεκτή για όλους. Αν έπρεπε να δώσουμε έναν τυπικό ορισμό θα ήταν κάπως έτσι:

Διαπραγμάτευση: η διαδικασία (ή ενέργεια) της συνδιάσκεψης ή συνομιλίας με άλλους με σκοπό να φτάσουν σε κάποια συμφωνία πάνω σε κάποιο θέμα

Επομένως, η διαπραγμάτευση φαίνεται μία πολύ λογική (έως και ενστικτώδης) επιλογή όταν ένα άτομο βρίσκεται σε μια κατάσταση όπου πρέπει να διεκδικήσει κάτι και πρέπει να προσπαθήσει να εξασφαλίσει την βέλτιστη για εκείνον λύση. Η διαδικασία βέβαια δεν εγγυάται ότι θα είναι πάντα επιτυχής ή εύκολη. Όπως και κάθε διαδικασία που στηρίζεται στην ανθρώπινη επικοινωνία, αντιμετωπίζει πολύ συχνά προβληματικές που στέκονται εμπόδιο στην απρόσκοπτη διεξαγωγή της. Ένα προφανές πρόβλημα που μπορεί να προκύψει είναι το πρόβλημα επικοινωνίας, είτε για πρακτικούς λόγους όπως π.χ. διαφορετικές ομιλούμενες γλώσσες, είτε λόγω διαφορετικής αντίληψης. Η πρώτη κατηγορία προβλημάτων είναι αρκετά προφανής. Είναι πολύ δύσκολο, έως ακατόρθωτο, να καταφέρουν να συνεννοηθούν 2 άνθρωποι που δεν μιλούν την ίδια γλώσσα ή δύο άνθρωποι που δεν έρχονται με κάποιον τρόπο σε επικοινωνία (λόγω απόστασης π.χ.). Πιο περίπλοκη είναι η δεύτερη κατηγορία, που αφορά την ανθρώπινη φύση. Οι άνθρωποι τείνουν να μεταφράσουν τον κόσμο γύρω τους με βάση τις δικές τους αντιλήψεις, πεποιθήσεις, ιδεοληψίες, συναισθήματα, κτλπ πράγμα που κάνει την επικοινωνία με άλλους ανθρώπους (που φέρουν με τη σειρά τους το δικό τους παρόμοιο σύνολο) δύσκολη. Σαν παράδειγμα θα μπορούσαμε να δανειστούμε μία σκηνή από μία παιδική ταινία της Disney:

Στην ταινία Pacahontas ο Τζον Σμιθ εξηγεί στην Πακαχόντας ότι έχουν έρθει να πάρουν κάτι πολύτιμο και χρυσό που υπάρχει άφθονο στην γη τους. Η Ποκαχόντας κάνοντας αναγωγή αυτού που μόλις άκουσε στα δικά της βιώματα του δείχνει το καλαμπόκι. Εκείνος της δείχνει ένα χρυσό νόμισμα.

Από το παραπάνω παράδειγμα - αν και όχι πραγματικό - γίνεται νομίζω άμεσα κατανοητό το μήνυμα που θέλουμε να περάσουμε.

Φυσικά αυτά δεν είναι τα μόνα προβλήματα που μπορεί να αντιμετωπίσει κανείς σε μία κατ'ιδίαν διαπραγμάτευση. Ένα άλλο συχνό φαινόμενο είναι ότι οι άνθρωποι αποκρύπτουν πληροφορία, όπως π.χ. τις προθέσεις τους, πράγμα που κάνει την διαπραγμάτευση πολύ πιο

απαιτητική. Ωστόσο η απόκρυψη πληροφοριών δεν γίνεται πάντα εθελημένα. Αντιθέτως τις περισσότερες από τις φορές γίνεται λόγω άγνοιας των ίδιων των διεκδικόντων. Για παράδειγμα μπορεί κάποιος να μην είναι απόλυτα βέβαιος για τον στόχο που έχει θέσει ή για το τι είναι το καλύτερο αποτέλεσμα για αυτόν. Κάτι τέτοιο μπορεί να αποπροσανατολίσει πολύ την διαδικασία και να οδηγήσει σε σημαντικά σφάλματα, με αποτέλεσμα το προϊόν της διαπραγμάτευσης να μην ευνοεί κανένα μέρος. Γενικότερα όσο πιο "σκιερό" είναι το τοπίο της διαπραγμάτευσης (το γνωστικό σύνολο που έχουν οι συμμετέχοντες) τόσο πιο δύσκολα και πιο αργά εξελίσσεται η διαδικασία, καθώς και οι 2 πλευρές γίνονται επιφυλακτικές, αλλά και τόσο περισσότερες οι πιθανότητες να παρεκκλίνει από την βέλτιστη λύση ή ακόμα και να μην καταλήξει πουθενά.

Σε όλα τα παραπάνω μπορούμε να προσθέσουμε εκτός από την υποκειμενικότητα και την ανειλικρίνεια, τον χαρακτήρα, το κοινωνικό ή πολιτισμικό υπόβαθρο (σε κάποιες κοινωνίες το "παζάρι" θεωρείται αποδεκτό, ενώ σε άλλες χαρακτηρίζεται ευτελές) και ακόμα και την ψυχική διάθεση του ατόμου, και πραγματικά η λίστα με τα εμπόδια της ανθρώπινης διαπραγμάτευσης φαίνεται να μην έχει όρια.

Με βάση όσα συζητήσαμε παραπάνω, είναι ήδη σαφές ότι η διαπραγμάτευση είναι ένα πολύσυχνο, καθημερινό φαινόμενο με τεράστια πολυπλοκότητα, πολυάριθμα εμπόδια και αβέβαιο αποτέλεσμα. Αν όμως μπορούσαμε να αφαιρέσουμε τον ανθρώπινο παράγοντα από αυτή την διαδικασία? Αν δηλαδή μπορούσαμε να ελαχιστοποιήσουμε τα προβλήματα που προκύπτουν στην συνεννόηση μεταξύ των συμμετεχόντων? Αν με κάποιο τρόπο μπορούσαμε να έχουμε αντικειμενικά μέρη από όλες τις πλευρές, που δεν αποκρύπτουν πληροφορία και έχουν σαφείς στόχους η διαπραγμάτευση θα έπαιρνε κατευθείαν άλλη μορφή. Στην ουσία της παραμένει η ίδια (δηλαδή ένας μηχανισμός διεκδίκησης ενός στόχου από κάποιο άτομο) αλλά ως προς όλα τα υπόλοιπα έχει απλοποιηθεί σημαντικά.

Κάτι τέτοιο είναι εφικτό και μπορεί να γίνει αν και εφόσον η διαπραγμάτευση διεξάγεται ανάμεσα σε αυτόματους πράκτορες λογισμικού. Οι πράκτορες αυτοί είναι προγράμματα ρυθμισμένα να εκτελούν διαπραγματεύσεις - εκτός των άλλων λειτουργιών τους - προασπιζόμενοι τα συμφέροντα του χρήστη. Επειδή ακριβώς μιλάμε για αυτοματοποιημένες, μηχανικές διαδικασίες που στηρίζονται σε απλά βήματα τα ακόλουθα είναι εξασφαλισμένα:

πλήρη διαφάνεια και πρόσβαση σε όλη την χρήσιμη πληροφορία

εξάλειψη αστάθμητων παραγόντων λόγω υποκειμενικότητα - οι πράκτορες παίρνουν μόνο λογικές αποφάσεις, στηριγμένες σε αντικειμενικά δεδομένα και μαθηματικά μοντέλα που θα αναλύσουμε παρακάτω ο αλγόριθμος θα τερματίζει ακόμα και αν η λύση είναι υποβέλτιστη[1]

Μετά από αυτή την σύντομη εισαγωγική ανάλυση ελπίζω να έγιναν κατανοητά η σημασία και η αναγκαιότητα της διαπραγμάτευσης γενικά, καθώς και κάποια από τα προβλήματα που θα πρέπει να λυθούν για να επιτύχουμε μία καλύτερη διαδικασία. Στα επόμενα κεφάλαια θα εξετάσουμε ειδικότερα σενάριο στα οποία η διαπραγμάτευση ανάμεσα σε πράκτορες στο διαδίκτυο είναι απαραίτητη (ή επιθυμητή) καθώς και τρόπους να προτυποποιήσουμε μία τέτοια διαδικασία.

2.2 Αποδόμηση μίας διαπραγματευτικής διαδικασίας

Για τους λόγους που αναφέρονται στα προηγούμενα κεφάλαιο τις τελευταίες δεκαετίες ερευνητές στον τομέα της τεχνητής νοημοσύνης έχουν επενδύσει αρκετό χρόνο στην δημιουργία μοντέλων διαπραγμάτευσης, κάποια από τα οποία είναι πολλά υποσχόμενα. Τα μοντέλα αυτά διαφέρουν συνήθως αρκετά μεταξύ τους και επομένως είναι μεγάλη η ανάγκη για την ύπαρξη ενός ενιαίου και επεκτάσιμου πλαισίου για τον ορισμό και τον χαρακτηρισμό των βασικών συστατικών που είναι απαραίτητα για την διεξαγωγή μίας αυτοματοποιημένης διαπραγμάτευσης . Η ύπαρξη μίας τέτοιας υποδομής αποτελεί το πρώτο βήμα στην αναγνώριση των πυρηνικών στοιχείων ενός αυτοματοποιημένου πράκτορα και επομένως ένας καλός τρόπος για να μπουν τα θεμέλια για την ανάπτυξή του.

Οι περισσότερες εργασίες και μελέτες μέχρι τώρα χωρίζουν την διαπραγμάτευση σε 2 τομείς, ανάλογα με την οπτική γωνία που κρατάνε: την θεωρητική ή μαθηματική οπτική και την πρακτική ή system-building οπτική. Η πρώτη, όπως δηλώνει και το όνομά της, είναι στηρίζεται κυρίως σε μαθηματικές θεωρίες, όπως είναι η θεωρία παιγνίων ή οι οικονομικές μέθοδοι, και μαθηματικά μοντέλα. Οι περισσότερες από αυτές έχουν αρκετές επιθυμητές ιδιότητες, όπως είναι η εξασφάλιση σύγκλισης και η σταθερότητα (stability), αλλά δυσκολεύονται να ανταποκριθούν σε πιο αφηρημένα προβλήματα που συνδυάζουν υποθέσεις

[1] *η διαδικασία δεν τερματίζει απαραίτητα για κάθε μαθηματική μοντελοποίηση (βλ. Κεφάλαιο 5). Στα πλαίσια της δική μας μελέτης όμως και επειδή θεωρούμε ότι η παραγωγή κάποιας λύσης, έστω και υποβέλτιστης, είναι σημαντικότερη από την αναζήτηση μίας τέλειας λύσης που μπορεί και να μην έρθει ποτέ, έχουμε προτυποποιήσει διαπραγματεύσεις με μαθηματικά μοντέλα που αναγκαστικά τερματίζουν*

ή τυχαιότητα με αποτέλεσμα να περιορίζονται ως ένα βαθμό. Συνήθεις παραδοχές που γίνονται για να επιτύχουν τέτοιου είδους μοντέλα είναι οι ακόλουθες:

οι πράκτορες είναι απόλυτα λογικοί

το σύνολο των πιθανών αποτελεσμάτων είναι σταθερό και γνωστό σε όλους τους συμμετέχοντες από την αρχή ως το τέλος

όλοι γνωρίζουν το αντίκτυπο που θα έχει για κάθε έναν ξεχωριστά το κάθε πιθανό αποτέλεσμα

όλοι γνωρίζουν την ενδεχόμενη αντίδραση των υπολοίπων σε κατάσταση ρίσκου (τις ενδεχόμενες κινήσεις των αντιπάλων)

Είναι εύκολο να καταλάβουμε επομένως πως τέτοιου είδους μοντέλα δυσκολεύονται να ανταποκριθούν σε πραγματικές καταστάσεις, να περιγράψουν την πολυπλοκότητα των υπαρχόντων συστημάτων και να αναπαραστήσουν την ποικιλία και τον πλούτο των πραγματικών προβλημάτων.

Αντίθετα, η πρακτική οπτική ακολουθεί περισσότερο μοντέλα και τεχνικές των κοινωνικών επιστημών. Με τα μοντέλα αυτά είναι πιο εύκολο να επικεντρωθούμε στην ίδια την διαδικασία της διαπραγμάτευσης (σε αντίθεση με τα θεωρητικά μοντέλα που επικεντρώνονται ως επί τω πλείστω στο αποτέλεσμα και αντιμετωπίζουν τα ενδιάμεσα βήματα σαν έναν απλό τρόπο για την επιλογή της βέλτιστης λύσης από το σύνολο των πιθανών αποτελεσμάτων) και να εντάξουμε σε αυτήν ρεαλιστικές υποθέσεις. Επίσης συνήθως τα μοντέλα αυτά έχουν λιγότερες απαιτήσεις σε ότι αφορά την υπολογιστική δύναμη που απαιτείται για την εκπόνηση κάποιου αποτελέσματος. Παρά το ότι έχουν ήδη χρησιμοποιηθεί με επιτυχία σε αρκετές πραγματικές καταστάσεις, υπάρχουν ακόμα προβληματικές που πρέπει να λυθούν. Κάποιες από αυτές είναι ότι τα συστήματα αυτά είναι πάντα ad hoc (αντιδρούν με βάση τα ερεθίσματα που παίρνουν ανά περίπτωση και δεν μπορούν να βγάλουν πιο γενικευμένα αποτελέσματα ή να προβλέψουν εκ των προτέρων καταστάσεις), τις περισσότερες φορές τα αποτελέσματα που προκύπτουν είναι υποβέλτιστα και τέλος δεν είναι πάντα κατανοητές οι συνθήκες ή το λογικό μονοπάτι που ακολουθήθηκε για να προκύψει κάποιο αποτέλεσμα.

Φυσικά υπάρχουν αρκετές άλλες προτάσεις για την κατηγοριοποίηση των μοντέλων διαπραγμάτευσης. Ένα παράδειγμα που επίσης έχει χρησιμοποιηθεί πολύ από αρκετούς ερευνητές είναι εκείνο που προτείνεται στην εργασία με τίτλο "Automated Negotiation: Prospects, Methods and Challenges" (Jennings, 2001), όπου προτείνεται μία τριμερής και περιλαμβάνει τις ακόλουθες κατηγορίες: θεωρητικά

μοντέλα, ευριστικά μοντέλα και μοντέλα βασισμένα στην επιχειρηματολογία. Δεν θα μπούμε σε περισσότερες λεπτομέρειες σχετικά με αυτά τα μοντέλα (άλλωστε είναι φανερό ότι ο προηγούμενος διαχωρισμός δημιουργεί 2 κατηγορίες υπερσύνολα αυτού και δεν έρχονται άμεσα σε αντίφαση), απλά το αναφέραμε σαν παράδειγμα για να δείξουμε ότι πολλές και διαφορετικές προσεγγίσεις είναι πιθανές και εξίσου δημοφιλείς. Στην παρούσα εργασία ακολουθούμε την πρώτη κατηγοριοποίηση και ακόμα και αν στην πράξη έχει ελάχιστο αντίκτυπο στην πορεία της εργασίας, ωστόσο θεωρούμε ότι εξυπηρετεί καλύτερα τον αναγνώστη για μία πληρέστερη κατανόηση του κειμένου.

Για τις ανάγκες του σχεδιασμού και ανάπτυξης αυτοματοποιημένων πρακτόρων που επικοινωνούν μεταξύ τους χρησιμοποιώντας το DeepGraphs πρότυπο και εμπλέκονται σε διαπραγμάτευση για τον τελικό τους στόχο, ήταν απαραίτητη η αποδόμηση της διαδικασίας της διαπραγμάτευσης σε διακριτά βήματα και την ανάλυση αυτών όσο γίνεται πιο κοντά στις ανάγκες μας. Για τον λόγο αυτό, ακολουθήσαμε την ανάλυση του **Lopes (2008).** Στην εργασία αυτή γίνεται μία εκτενής ανάλυση του χώρου και προτείνεται η ακόλουθη μορφή για το πλαίσιο διαπραγμάτευσης:

Προκαταρκτικά	**1**	Κοινωνική Διαμάχη (εντοπισμός ή αναζήτηση)
	2	Συμμετέχοντα μέρη
Προ της διαπραγμάτευσης	**3**	Συλλογή και σχηματισμών προσωπικών πληροφοριών (ορισμός και εκτέλεση των εργασιών-κλειδί για την προ της διαπραγμάτευσης φάση)
	4	Ανάλυση αντιπάλου (συλλογή και χρήση πληροφοριών κλειδί)
	5	Ορισμός πρωτοκόλλου και επιλογή αρχικής στρατηγικής
Πραγματική διαπραγμάτευση	**6**	Ανταλλαγή προσφορών και feedback
	7	Επιχειρηματολογία (ανταλλαγή απειλών, υποσχέσεων, κλπ)
	8	Εκμάθηση (στην διαπραγμάτευση)
	9	Δυναμική επιλογή στρατηγικής (επιλογή νέας στρατηγικής)
	10	Αδιέξοδο αποτέλεσμα
Επαναδιαπραγμάτευση	**11**	Ανάλυση και βελτίωση του τελικού αποτελέσματος

Αν παρατηρήσουμε την ανωτέρω μορφή μπορούμε αυτόματα να κάνουμε κάποιες λίγες παρατηρήσεις. Αρχικά βλέπουμε ότι τα στάδια είναι απολύτως διακριτά μεταξύ τους, πράγμα το οποίο ίσως προκαλεί εντύπωση αρχικά στον αναγνώστη, γιατί σαν διαδικασία ούτε είναι

ούτε φαίνεται απλή. Αντιθέτως, συνήθως οι διαπραγματεύσεις είναι πολύπλοκες διαδικασίες με πολυάριθμους συμμετέχοντες. Ωστόσο, τόσο η διαπραγμάτευση - στην ουσία της - όσο και οι περισσότερες κοινωνικές συναναστροφές μπορούν να αναλυθούν σε ευδιάκριτα στάδια, η επανάληψη και διάνθιση των οποίων οδηγεί στο πολυδαίδαλο αποτέλεσμα που φτάνει στον τελικό παρατηρητή. Μία δεύτερη παρατήρηση είναι ότι τα 11 αυτά βήματα ανήκουν σε ευρύτερες κατηγορίες: προκαταρκτικά, προ της διαπραγμάτευσης, πραγματική διαπραγμάτευση, επαναδιαπραγμάτευση. Οι περισσότερες σχετικές μελέτες που επιχειρούν κάτι αντίστοιχο παραδέχονται ότι η διαδικασία σε γενικές γραμμές χωρίζεται σε 3 φάσεις: αρχή ή αρχικοποίηση (το στάδιο προετοιμασίας για την διαπραγμάτευση), μέση ή φάση επίλυσης, τέλος ή φάση αποτελέσματος.

Στις ακόλουθες σελίδες τα παραπάνω στάδια αναλύονται συνοπτικά. Σε περίπτωση που ο αναγνώστης επιθυμεί να εμβαθύνει περισσότερο, παραπέμπεται στην αντίστοιχη δημοσίευση.

Κοινωνική Διαμάχη

Σε αυτό το στάδιο γίνεται ο εντοπισμός της διαμάχης, η αιτία που πυροδοτεί την διαδικασία, η κινητήρια δύναμη. Είναι το στάδιο που διαπιστώνεται η διαμάχη και μία ανάγκη για επίλυση προκύπτει. Συγκεκριμένα σε αυτό το σημείο γίνεται φανερό ότι υπάρχουν διακριτά "στρατόπεδα" καθένα με το δικό του συμφέρον (που όμως έρχεται σε αντίθεση με το συμφέρον κάποιου άλλου) και μία προσπάθεια για προσδιορισμό της έκτασης και του σκοπού της αντιπαράθεσης. Αρκετή δουλειά έχει γίνει και στον τομέα αυτό, τον εντοπισμό δηλαδή μίας διαμάχης, πράγμα διόλου εύκολο σε ένα πολύπλοκο σύστημα που συνδυάζει πολλούς συμμετέχοντες, με πολλές επιθυμίες, περιορισμούς, κλπ.

Συμμετέχοντα μέρη

Εδώ είναι το στάδιο που αποφασίζεται ποιος θα λάβει μέρος στην διαπραγμάτευση. Μετά το πρώτο βήμα που έχει γίνει σαφές το αντικείμενο της διαπραγμάτευσης δηλώνουν συμμετοχή όσοι έχουν πραγματικό ενδιαφέρον, δηλαδή διεκδικούν το έπαθλο και έχουν κάτι σημαντικό να κερδίσουν από την διαδικασία.

Η διαπραγμάτευση μπορεί να περιλαμβάνει δύο μέρη (διμερής) ή περισσότερα μέρη (πολυμερής) και να αφορά ένα θέμα (single-issue) ή πολλά θέματα (multi-issue). Στην περίπτωση της πολυμερούς διαπραγμάτευσης, αυτή μπορεί να κατηγοριοποιηθεί περαιτέρω ανάλογα με τα μέρη που θα συμμετέχουν:

- ένας-προς-πολλούς (one-to-many): όπως δηλώνει και το όνομα έχουμε ένα μόνο μέρος να διαπραγματεύεται με πολλά «αντίπαλα» μέρη. Συνήθως τέτοια συστήματα αντιμετωπίζονται ως πολλά ένα-προς-ένα συστήματα.
- Πολλοί-προς-πολλούς (many-to-many): στην περίπτωση αυτή έχουμε πολλαπλά μέρη να εμπλέκονται με πολλά μέρη.

Είναι σαφές ότι η πολυπλοκότητα τέτοιων συστημάτων είναι μεγάλη και όσο περισσότερα είναι τα μέρη που εμπλέκονται τόσο αυξάνεται η δυσκολία με πολλούς και διαφορετικούς τρόπους. Στην εργασία αυτή -όπως και η περισσότερη υπάρχουσα έρευνα και βιβλιογραφία- θα επικεντρωθούμε κυρίως στην ένα-προς-ένα διαπραγμάτευση ακριβώς για να αποφύγουμε την επιπλέον πολυπλοκότητα που προσθέτουν τα πολλαπλά μέρη. Ωστόσο η λύση που προτείνουμε είναι απόλυτα επεκτάσιμη. Περισσότερες λεπτομέρειες σχετικά παρουσιάζονται σε επόμενα κεφάλαια.

Συλλογή και σχηματισμών προσωπικών πληροφοριών

Οι επιτυχημένες ανθρώπινες διαπραγματεύσεις συμφωνούν σε ένα πράγμα: το κλειδί για την επιτυχία είναι η προετοιμασία. Αυτό λοιπόν είναι από τα βασικά στάδια της προ της διαπραγμάτευσης διαδικασίας και αποτελεί το βήμα στο οποίο οι συμμετέχοντες χτίζουν το γνωστικό τους πλαίσιο γύρω από την διαπραγμάτευση. Οπλίζουν την φαρέτρα των επιχειρημάτων τους όσο γίνεται καλύτερα γίνεται για να ανταποκριθούν σωστά σε αυτό που πρόκειται να ακολουθήσει.

Για παράδειγμα, αν κάποιος θέλει να αγοράσει ένα αμάξι και σκοπεύει να συζητήσει σχετικά με την τιμή, θα πρέπει να έχει εκ των προτέρων μία εικόνα για τα αμάξια αρχικά και δευτερευόντως για το τι ο ίδιος χρειάζεται. Και φυσικά η λίστα δεν τελειώνει εδώ! Καλό θα ήταν ακόμα να ξέρει και άλλες πληροφορίες, όπως που κυμαίνονται οι τιμές των αυτοκινήτων στην αγορά, τι χαρακτηριστικά πρέπει να έχει ο κινητήρας για να είναι αποδοτικός και οικονομικός, τι εγγύηση θα έχει, κ.α. Ένας καλός αγοραστής επομένως πρέπει να έχει γνώσεις από μηχανολογία έως λογιστική για να πετύχει μία καλή προσφορά.

Οι διαπραγματευτές που προετοιμάζονται προσεκτικά και σχεδιάζουν προσπαθούν να κάνουν όσο γίνεται περισσότερες ενέργειες πριν την διαπραγμάτευση, περιλαμβάνοντας και τα ακόλουθα:

- Προσδιορισμός των θεμάτων
- Προσδιορισμός της ατζέντας της διαπραγμάτευσης
- Προτεραιότητα των θεμάτων
- Ορισμός των ορίων και στόχων

Λέγοντας θέματα εννοούμε τους πόρους που θα πρέπει να διανεμηθούν ή τα προβλήματα που πρέπει να λυθούν στην διαπραγμάτευση. Οι δύο πρώτες ενέργειες είναι αρκετά προφανές γιατί είναι απαραίτητες και σημαντικές για την πρόοδο της διαπραγμάτευσης. Οι συμμετέχοντες πρέπει να είναι κατασταλαγμένοι για ποιο λόγο λαμβάνουν μέρος, τι προσπαθούν να κερδίσουν και τι θα διεκδικήσουν. Συνήθως όταν πολλαπλά μέρη συμμετέχουν, από την ένωση όλων των θεμάτων προκύπτει μία λίστα που αποτελεί την ατζέντα της διαπραγμάτευσης.

- Ο ορισμός προτεραιοτήτων συνήθως περιλαμβάνει δύο βήματα:
- Προσδιορισμός ποιών θεμάτων είναι πιο σημαντικά
- Εντοπισμός συνδέσεων μεταξύ των θεμάτων
- Τις περισσότερες φορές οι προτεραιότητες ενός μέρους δεν είναι διαθέσιμες στα υπόλοιπα μέρη.

Τελευταία από τις ενέργειες που αναφέραμε παραπάνω είναι ο ορισμός ορίου και στόχου. Όριο λέμε το σημείο εκείνο που ο συμμετέχων αποφασίζει ότι θα πρέπει να σταματήσει την διαπραγμάτευση αντί να συνεχίσει, καθώς κάθε διακανονισμός από εκεί και πέρα δεν είναι αποδεκτός. Στόχος είναι το σημείο στο οποίο ο συμμετέχων ρεαλιστικά αποδέχεται τον διακανονισμό.

Ιδανικά τα βήματα αυτά διαδραματίζονται με την σειρά που τα παρουσιάσαμε και διακριτά το ένα από το άλλο. Ωστόσο στην πραγματικότητα σπάνια η πληροφορία λαμβάνεται τόσο εύκολα και άμεσα από την αρχή. Το συνηθέστερο είναι ότι όλα τα μέρη της διαδικασίας θα μάθουν κάποια στιγμή στην πορεία κάποιο κομμάτι πληροφορίας που θα τους ωθήσει να αναθεωρήσουν ή να αλλάξουν κάποια από τις προηγούμενες ενέργειές τους – στην πιο απλή να αναθεωρήσουν ένα κομμάτι πληροφορίας, στην χειρότερη να αλλάξουν τον στόχο τους ή μέρος αυτού.

Οι περισσότερες έρευνες που έχουν γίνει μέχρι τώρα αφορούν το κυρίως κομμάτι της διαπραγμάτευσης και δίνουν ελάχιστη έως και καθόλου σημασία στο προηγούμενο κομμάτι, την προετοιμασία δηλαδή όπως την αναλύσαμε παραπάνω. Πράγματα όπως η ατζέντα και τα θέματα της διαπραγμάτευσης θεωρούνται προκαθορισμένα (άγνωστο και αδιάφορο το πώς) για διευκόλυνση και εσπευσμένη προώθηση στην διεξαγωγή της κατεξοχήν διαπραγματευτικής διαδικασίας. Αυτό ωστόσο δεν συνεπάγεται ότι τα βήματα αυτά είναι ήσσονος σημασίας, αντιθέτως όπως είπαμε αποτελούν από τα σημαντικότερα στάδια εφόσον μπορεί έμμεσα να εξασφαλίσει την νίκη σε κάποιον καλά προετοιμασμένο συμμετέχοντα.

Ανάλυση αντιπάλου

Στα πλαίσια της προετοιμασίας είναι πολύ σημαντικό να έχει ο συμμετέχων μία καλή γνώση του αντιπάλου. Να ξέρει δηλαδή τι θέλει ο αντίπαλος (στόχους), πόσο το θέλει (προτεραιότητες) για να δομήσει μία ακόμα πιο στέρεη και αποτελεσματική επιχειρηματολογία στην συνέχεια. Συγκεκριμένα τα παρακάτω κομμάτια πληροφορίας είναι πολύ σημαντικά:

- Τα όρια και τους στοχους του αντιπάλου (ή των αντιπάλων)
- Το ιστορικό διαπραγμάτευσης του αντιπάλου
- Τις στρατηγικές που πρόκειται να χρησιμοποιήσει

Τα περισσότερα από αυτά, στην πραγματικότητα σπάνια είναι διαθέσιμα για όλους τους συμμετέχοντες. Συνήθως είναι πληροφορίες που ο κάθε συμμετέχων κρατάει για τον εαυτό του και επομένως πρέπει να τα υποθέσει για τους υπολοιπους. Κλασσική τακτική είναι να κάνει εικασίες για τα όρια του αντιπάλου ή να χρησιμοποιούν τις δικές τους αρχικές προθέσεις σαν οδηγό και να υποθέτουν ότι και οι υπόλοιποι θέλουν παρόμοια πράγματα.

Η συλλογή της ιστορικότητας των αντιπάλων επίσης έχει μεγάλη σημασία. Το πως έχουν δράσει στο παρελθόν είναι μία καλή ένδειξη για το πως θα δράσουν πιθανώς και στο μέλλον. Επομένως μια προσεκτική συλλογή και επεξεργασία του ιστορικού μπορεί να δώσει χρήσιμα στοιχεία. Ωστόσο αυτό είναι απλά και μόνο μία υπόθεση και κάθε συμμετέχων μπορεί να αντιδράσει εν τέλει διαφορετικά σε κάθε περίσταση, οπότε απαιτείται μεγάλη προσοχή.

Τέλος, οι στρατηγικές είναι επισης ένα πολύ χρήσιμο κομμάτι πληροφορίας. Είναι σχεδόν απίθανο οι συμετέχοντες να αποκαλύψουν κάποια σχετική πληροφορία, αλλά μπορεί να συμπεράνει κάποια πράγματα από τα δεδομένα που συλλέγει για αυτούς κατά τη διάρκεια της διαδικασίας.

Γενικά είναι εξαιρετικά χρήσιμο να συλλέξεις όσο γίνεται περισσότερη πληροφορία για τα αντίπαλα μέρη πριν από την εκκίνηση της διαπραγμάτευσης. Σε πραγματικές συνθήκες η όλη προσπάθεια για την συλλογή αυτής της πληροφορίας απαιτεί πολύ χρόνο και κόπο, ωστόσο στην δική μας εργασία θα θεωρήσουμε ότι πράκτορες είναι απόλυτα ειλικρινείς, αντικειμενικοί και καλώς προσκείμενοι στους αντιπάλους τους και άρα δεν έχουν ιδιαίτερο λόγο να αποκρύψουν πληροφορίες που είναι σημαντικές για την διαδικασία και δεν πρόκειται να βλάψουν τον χρήστη άμεσα ή έμμεσα (π.χ. να μοιραστούν προσωπικά στοιχεία του χρήστη μπορεί να αποβεί επικίνδυνο και άρα θα πρέπει να αποφευχθεί με κάθε κόστος, ακόμα και αν είναι χρήσιμο για τον αντίπαλο. Αντιθέτως η αποκάλυψη του στόχου, του οριου ή και της στρατηγικής μπορεί μόνο να βοηθήσει στην εξέλιξη της διαπραγμάτευσης χωρίς αντίκτυπο στον χρήστη ή στο αποτέλεσμα).

Ορισμός πρωτοκόλλου και επιλογή αρχικής στρατηγικής

Το πρωτόκολλο της διαπραγμάτευσης είναι εκείνο που καθορίζει τα στάδια της διαπραγμάτευσης (π.χ. τις αποδεκτές καταστάσεις), τις έγκυρες ενέργειες των συμμετεχόντων στα διάφορα στάδια (π.χ. σε ποιο στάδιο μπορεί να στείλει κάποιος μήνυμα και αντίπαλο μέρος) και τα γεγονότα που θα προκαλέσουν αλλαγή από μία κατάσταση σε μία άλλη (π.χ. αποδοχή μίας πρότασης). Τις περισσότερες φορές το πρωτόκολλο περιορίζει το ποιες ενέργειες μπορούν ή δεν μπορούν να γίνουν σε κάθε βήμα, αλλά δεν προσδιορίζει ακριβώς τις ενέργειες αυτές. Στην πραγματικότητα περιγράφει κομβικά σημεία στα οποία οι συμμετέχοντες καλούνται να αποφασίσουν με βάση την στρατηγική τους και ανάλογα με το πως θα ενεργήσουν ξετυλίγεται και ένα διαφορετικό μονοπάτι κάθε φορά.

Η περιγραφή ενός πρωτοκόλλου είναι εξέχουσας σημασίας για την διεξαγωγή της διαπραγμάτευσης, καθώς επιτρέπει στα διάφορα μέλη να οργανωθούν με τον βέλτιστο τρόπο. Εφόσον έχουν συλλέξει ήδη ότι δυνατή πληροφορία μπορούσαν για τους εαυτούς τους και τους αντιπάλους (βλέπε προηγούμενα στάδια) εάν ξέρουν και τους κανόνες που διέπουν την αλληλεπίδραση των μερών της διαπραγμάτευσης, μπορούν να επιλέξουν πλέον την καλύτερη δυνατή στρατηγική. Το πρωτόκολλο ωστόσο δεν είναι απαραίτητο να είναι πάντα περίπλοκο ή «εκλεπτυσμένο». Ανάλογα τις περιστάσεις μπορεί να επιλεγεί ένα πολύ πρωτόλειο πρωτόκολλο, για παράδειγμα ένα πρωτόκολλο που απλώς επιτρέπει την ανταλλαγή προσφορών και προσφορών ανάμεσα στους συμμετέχοντες, ή να εμπλουτιστεί με κάποια μορφή ανάδρασης, να επιτρέπει δηλαδή σε όσους κάνουν προσφορά να παίρνουν πίσω κάποια αποτελέσματα ώστε να μπορούν να προβλέψουν καλύτερα τις κινήσεις των αντιπάλων και να σχεδιάσουν καλύτερα τις δικές τους μελλοντικές προτάσεις. Κάποια φυσικά είναι εξαιρετικά περίπλοκα, όπως εκείνα που επιτρέπουν στους συμμετέχοντες να αιτιολογούν και να επιχειρηματολογούν πάνω στις προτάσεις τους.

Το άλλο κομμάτι αυτού του σταδίου είναι η επιλογή στρατηγικής. Δεν θα εμβαθύνουμε ιδιαίτερα σχετικά με τους λόγους που η επιλογή της σωστής αρχικής στρατηγικής, θα αναφέρουμε μόνο ότι είναι ένα πολύ κρίσιμο βήμα στην προετοιμασία της διαπραγμάτευσης, αλλά και για την πορεία της (ειδικά σε διαπραγματεύσεις που δεν επιτρέπουν την αλλαγή στρατηγικής σε επόμενα στάδια), και σε πολλές περιπτώσεις μπορεί να καθορίσει σε μεγάλο βαθμό την έκβαση της.

Οι διαπραγματευτικές στρατηγικές αντικατοπτρίζουν μία ποικιλία συμπεριφορών και μπορούν να οδηγήσουν σε εντυπωσιακά διαφορετικά αποτελέσματα. Οι 3 βασικότερες ομάδες που χρησιμοποιούνται συνήθως (στις ανθρώπινες διαπραγματεύσεις) είναι οι ακόλουθες:

- *Contending (competing or dominating)* – στα ελληνικά θα το μεταφράζαμε κυριαρχική ή ανταγωνιστική. Στην στρατηγική αυτή οι συμμετέχοντες προσπαθούν

να πείσουν τους αντιπάλους τους να υποκύψουν, εμμένοντας στις δικές τους επιθυμίες

- *Concession making (yielding, accommodating, obliging)* – είναι η στρατηγική όπου ο διαπραγματευτής εμφανίζεται υποχωρητικός ή ενδοτικός και προτίθεται να ελαττώσει τις απαιτήσεις του προκειμένου να φτάσουν με τα υπόλοιπα μέλη σε μία μέση λύση κοινώς αποδεκτή
- *Problem solving (collaborating, integrating)* – η περίπτωση αυτή θυμίζει λίγο μία ενδιάμεση κατάσταση στις 2 προηγούμενες, καθώς ο διαπραγματευτής προσπαθεί χωρίς να μειώσει τις επιθυμίες και τους στόχους του να βρει τρόπους να τις ενσωματώσει στους στόχους και των αντίπαλων μερών και έτσι να φτάσουν σε μία συμφωνία (ενσωματωτική συμφωνία)

Δεν θα επεκταθούμε περισσότερο στην ανάλυση των στρατηγικών, απλά παραθέσαμε τις παραπάνω σαν παραδείγματα πιθανών επιλογών. Αξίζει ωστόσο να σημειωθεί ότι καθεμία από τις παραπάνω στρατηγικές έχει δυνατά και αδύνατα σημεία. Για παράδειγμα ένας διαπραγματευτής που ακολουθεί την πρώτη στρατηγική βασίζεται στην πίεση του και στην αδυναμία των αντιπάλων του, ελπίζοντας ότι θα «σπάσουν» και εν τέλει θα δεχτούν την οπτική του. Στην πράξη βέβαια τόσο άκαμπτες στάσεις δύσκολα ευδοκιμούν καθώς δεν αφήνουν κανένα περιθώριο για ενοποιητικές συμφωνίες και άρα συχνά αποτυγχάνουν (γενικά μιλώντας οι διαπραγματευτές που απαιτούν πολλά συνήθως αποτυγχάνουν). Επομένως είναι πολύ σημαντική η προσεκτική επιλογή στρατηγικής, ανάλογα την περίσταση.

Τέλος, απλά να αναφέρουμε ότι σε κάποιες διαπραγματευτικές διαδικασίες είναι δυνατόν η στρατηγική των μερών να είναι δυναμική, να αλλάζει δηλαδή ανάλογα με την νέα πληροφορία και την πορεία που παίρνει η διαπραγμάτευση κάθε φορά (η επιλογή γίνεται πάλι σε προκαθορισμένα βήματα). Κάτι εξαιρετικά συνηθισμένο στις ανθρώπινες διαπραγματεύσεις, που όμως είναι ιδιαίτερα περίπλοκο να μιμηθεί ένα αυτόματο σύστημα.

Ανταλλαγή προσφορών και feedback

Είναι το πρώτο από τα 5 συνολικά στάδια της «Πραγματικής Διαπραγμάτευσης» και αποτελεί και την καρδιά της διαδικασίας. Είναι το στάδιο όπου ανταλλάσσονται προσφορές ανάμεσα στα μέλη και δίνονται και οι αντίστοιχες απαντήσεις. Όλα γίνονται πάντα με προϋπόθεση ότι τα όλα τα μέρη της διαπραγμάτευσης έχουν καλές προθέσεις και δεσμεύονται στην διαδικασία, δηλαδή θέλουν να προχωρήσει και να βρεθεί μία λύση. Καλή πρακτική για να αποδείξει κάποιος τα παραπάνω είναι να κάνει υποχωρήσεις. Ένας διαπραγματευτής που δεν

υποχωρεί σε τίποτα αντιμετωπίζεται σαν κακόβουλος (προσπαθεί να σαμποτάρει την διαδικασία) και η στάση του μεταφράζεται σαν απροθυμία για διαπραγμάτευση.

Ιδανικά μετά από αυτό το στάδιο η διαπραγμάτευση θα φτάσει σε ένα σημείο που είναι κοινώς αποδεκτό και αποτελεί την λύση ή το αποτέλεσμα της διαπραγμάτευσης. Ωστόσο, όπως μπορεί κανείς να φανταστεί το στάδιο αυτό δεν είναι καθόλου απλό. Εκτός από την προφανή ανταλλαγή προσφορών και προτάσεων, σε αυτό το στάδιο γίνεται και η βασική μεταφορά πληροφορίας από το ένα μέρος στο άλλο κυρίως μέσω των απαντήσεων. Συζητήσαμε ήδη ότι στο στάδιο 4 (Ανάλυση αντιπάλου) έχει γίνει μία πρώτη γνωριμία με τον αντίπαλο. Ανάλογα την κατάσταση το στάδιο αυτό μπορεί να είναι αρκετό από μόνο του για να ξέρουμε ακριβώς τους στόχους και τις μεθόδους του αντιπάλου. Στις περισσότερες περιπτώσεις πάντως αυτό από μόνο του δεν αρκεί, επομένως η σωστή επεξεργασία των προτάσεων και των απαντήσεων του αντιπάλου μπορεί να αποδειχτεί πολύτιμη. Αξίζει να σημειωθεί πως οι απαντήσεις από κάποιους ερευνητές θεωρούνται τόσο σημαντικές ώστε τις αντιμετωπίζουν σαν ένα είδος επιχειρήματος και αυτές.

Στις ανθρώπινες διαπραγματεύσεις προκύπτουν 2 πολύ συχνά διλήμματα στο βήμα αυτό. Το πρώτο αφορά την ειλικρίνεια του διαπραγματευτή και το δεύτερο την εμπιστοσύνη. Όταν ένας συμμετέχων θέλει να κάνει μία πρόταση, σκέφτεται πόσα πράγματα πρέπει να αποκαλύψει και πόσα να κρατήσει κρυφά. Αν πει περισσότερα από όσα χρειάζεται, τότε δίνει στους αντιπάλους του το πλεονέκτημα, ενώ αν πει λιγότερα, κινδυνεύει να μην προχωρήσει καθόλου η διαδικασία. Από την άλλη το θέμα της εμπιστοσύνης τίθεται σε εκείνους που καλούνται να αποτιμήσουν μία πρόταση. Πόσο πρέπει να εμπιστευθούν την πρόταση αυτή? Και σε αυτό το σημείο φυσικά, η πορεία της διαπραγμάτευσης θα εξαρτηθεί από την επιλογή στάσης και συμπεριφοράς που θα κάνουν οι συμμετέχοντες. Αν αποφασίσουν να είναι ανταγωνιστικοί, να μην υποχωρούν σε τίποτα και να αποκρύπτουν κάθε πιθανή πληροφορία, ή να είναι πιο συνεργάσιμοι και να ανταλλάσσουν πληροφορία αναζητώντας μία κοινή λύση.

Είναι σαφές ότι αν και μπορεί να προτυποποιηθεί μαθηματικά (π.χ. συντελεστές ειλικρίνειας) στην πραγματικότητα τα αυτοματοποιημένα συστήματα προσπαθούν να αποτρέψουν ακριβώς τέτοιες καταστάσεις ανειλικρίνειας και ασάφειας που περιγράψαμε. Μιας και αποτελεί τον πυρήνα της διαπραγματευτικής διαδικασίας, έχουν γίνει πολλές δουλειές πάνω σε αυτό το στάδιο, είτε με εστίαση στο κομμάτι των προτάσεων-αντιπροτάσεων είτε στο κομμάτι της συλλογής πληροφορίας και ανάδρασης. Ωστόσο υπάρχουν ακόμα κάποια σημεία που μένουν να εξετασθούν, όπως πως να μεταφερθεί και να αξιοποιηθεί βέλτιστα η πληροφορία που συλλέχθηκε (π.χ. σχηματισμός καλύτερων προτάσεων) και η μέτρηση της απόδοσης (π.χ. πόσο βελτιώθηκαν οι προτάσεις μου μετά την χρήση της πληροφορίας).

Επιχειρηματολογία (ανταλλαγή απειλών, υποσχέσεων, κλπ)

Παρότι οι συμμετέχοντες μπορεί να περνούν ένα μεγάλο χρονικό διάστημα στην συλλογή και πληροφορίας για να στηρίξουν την οπτική τους, ωστόσο η κατασκευή επιχειρημάτων (και αντεπιχειρημάτων) είναι ένα από τα πιο δυνατά σημεία της διαπραγμάτευσης. Τα πειστικά επιχειρήματα είναι το βασικό όπλο του διαπραγματευτή για να πείσει τον αντίπαλό του να δεχτεί μία συγκεκριμένη πρόταση. Ενδεικτικά μπορούμε να αναφέρουμε 3 γενικές κατηγορίες επιχειρημάτων που όμως σε καμία περίπτωση δεν είναι αντιπροσωπευτικές όλου του φάσματος επιχειρημάτων, καθώς τα επιχειρήματα έχουν νόημα και υπόσταση μόνο στα πλαίσια της διαπραγμάτευσης που τα γεννά και άρα έτσι πρέπει να αντιμετωπίζονται και να μελετώνται:

- Επίκληση σε προηγούμενα αντιπαραδείγματα –προσπαθεί ο διαπραγματευτής να τονίσει μία αντίφαση ανάμεσα στο τι ζητάει και κάποια γεγονότα του παρελθόντος
- Επίκληση σε επικρατούσες πρακτικές – χρησιμοποιώντας παραδείγματα προηγούμενων επιτυχημένων πρακτικών
- Επίκληση σε προσωπικά συμφέροντα – ο διαπραγματευτής προσπαθεί να εξηγήσει ότι η αποδοχή της πρότασής του θα βοηθήσει επίτευξη ενός σημαντικού στόχου

Τα πιο συνηθισμένα επιχειρήματα είναι οι υποσχέσεις και οι απειλές. Οι απειλές δείχνουν την πρόθεση να τιμωρήσουν κάποιον στόχο σε περίπτωση που εκείνος αρνηθεί την πρόταση ή την δεδομένη ενέργεια. Όσο μεγαλύτερη είναι η απειλή τόσο πιο αποδοτική είναι. Επίσης είναι πολύ σημαντική η πειστικότητα της απειλής, δηλαδή η πιθανότητα όντως να συμβεί σε περίπτωση που χρειαστεί. Οι απειλές στοχεύουν στο να κάνουν λιγότερο ελκυστική την απόρριψη της πρότασης που διακυβεύεται σε σχέση με την αποδοχή της. Το μειονέκτημα στην χρήση απειλών είναι ότι τείνουν να δημιουργούν δυσαρέσκεια και αντίσταση, οπότε είναι πολύ πιθανό να γεννήσουν νέες απειλές και συγκρούσεις.

Στον αντίποδα των απειλών βρίσκονται οι υποσχέσεις που είναι το «αρνητικό» τους. Οι υποσχέσεις έχουν σκοπό να επιβραβεύσουν τον στόχο σε περίπτωση που δεχτεί την πρόταση, αντί να τον τιμωρήσουν σε περίπτωση που αρνηθεί. Η πειστικότητα και η αξιοπιστία της υπόσχεσης παίζει και εδώ μεγάλο ρόλο, όπως και στις απειλές. Το κακό με τις υποσχέσεις είναι ότι σε περίπτωση που επιτύχουν τον σκοπό τους (τα αντίπαλα μέρη δεχτούν την πρόταση) ο διαπραγματευτής πρέπει να εκπληρώσει εκείνο που υποσχέθηκε, πράγμα που συνήθως κοστίζει.

Συνήθως χρησιμοποιείται ένας συνδυασμός από απειλές και υποσχέσεις. Οι απειλές λειτουργούν ως ενδείξεις των συνόρων: σηματοδοτούν τα όρια κάτω από τα οποία οι

διαπραγματευτές αρνούνται να υποχωρήσουν και να θέσουν ένα εύρος που είναι επιθυμητό και ανεκτό να κυμαίνεται η συμπεριφορά των αντιπάλων. Οι υποσχέσεις από την άλλη λειτουργούν ενθαρρυντικά στην συνεργασία, στα πλαίσια των ορίων που έχουν τεθεί.

Μέχρι πρόσφατα η επιχειρηματολογία ήταν ένα κομμάτι της ανθρώπινης συμπεριφοράς που αφορούσε καθαρά φιλοσόφους και επιστήμονες της λογικής, στα πλαίσια του να κατανοήσουν πως οι άνθρωποι αλληλεπιδρούν μεταξύ τους. Πρόσφατα ωστόσο άρχισε να απασχολεί και ένα μεγάλο μέρος των ερευνητών τεχνητής νοημοσύνης και συγκεκριμένα εκείνων που ασχολούνται με αυτόματους πράκτορες και αόριστη λογική. Συγκεκριμένα, όπως ήδη αναφέραμε, το κομμάτι αυτό θεωρείται από πολλούς ερευνητές ένα από τα πιο νευραλγικά της όλης διαδικασίας για αυτό και έχει μελετηθεί εκτενώς από πάρα πολλούς στον χώρο. Ωστόσο, οι περισσότερες προσεγγίσεις αντιμετωπίζουν ακόμα κάποιες προβληματικές στην μοντελοποίηση της διαδικασίας, είτε αυτές αφορούν την πολυπλοκότητα (αλγόριθμοι που μπορεί να έχουν καλά αποτελέσματα αλλά εισάγουν μεγάλες καθυστερήσεις) είτε αφορούν την ευελιξία του συστήματος (δύσκαμπτα και δύσκολα επεκτάσιμα συστήματα).

Η σχέση πάντως ανάμεσα στην διαπραγμάτευση και το πρωτόκολλο επιχειρηματολογίας φαίνεται να είναι πολύ σημαντική και για αυτό το λόγο αξίζει ίσως να μελετηθεί περεταίρω.

Εκμάθηση (στην διαπραγμάτευση)

Η διαπραγμάτευση είναι από μόνη της μία διαδικασία που μαθαίνεται. Αυτό αποδεικνύεται άλλωστε από το γεγονός ότι οι άνθρωποι μπορούν να εκπαιδευτούν στην διαδικασία αυτή ώστε να βελτιώσουν τις ικανότητές τους μέσω εξάσκησης, προπόνησης, καθοδήγησης, κλπ. Υπάρχουν 4 βασικές μέθοδοι που ακολουθούν οι διαπραγματευτές για εκμάθηση και αυτές περιγράφονται επιγραμματικά στην συνέχεια.

- Διδακτική εκμάθηση ή βασισμένη σε αξιώματα: στην περίπτωση αυτή ο διαπραγματευτής μαθαίνει (εκπαιδεύεται) σε συγκεκριμένες γενικές αρχές ή αξιώματα και στην συνέχεια καλείται να λύσει προβλήματα με βάση αυτές (ιδανικά προβλήματα που προφανώς σχετίζονται με αυτές)
- Εκμάθηση μέσω ανατροφοδότησης (feedback) ή αποκάλυψης πληροφορίας: βασίζεται στην αρχή ότι ο διαπραγματευτής μπορεί να μάθει από την ίδια την συμπεριφορά του στο παρόν ή στο παρελθόν, δηλαδή αναγνωρίζει ότι το μελλοντικό αποτέλεσμα επηρεάζεται από το παρόν

- Εκμάθηση μέσω αναλογίας ή αναλογική εκμάθηση: εάν και εφόσον παρατηρηθεί κάποια συσχέτιση ανάμεσα σε λύσεις του παρόντος προβλήματος με κάποια άλλα από το παρελθόν, ο διαπραγματευτής προσπαθεί να αντιστοιχήσει αντικείμενα και καταστάσεις για να παράξει μία λύση
- Παρατηρητική εκμάθηση ή μίμηση: μέθοδος βασισμένη στην πεποίθηση ότι ο διαπραγματευτής μπορεί να βελτιώσει τις ικανότητές του παρατηρώντας άλλους

Και αυτό το βήμα μπορεί να αποτελέσει κλειδί στην εξέλιξη μίας διαπραγμάτευσης, επομένως και εδώ έχουν γίνει σημαντικές προσπάθειες. Μπορούμε να πούμε ότι από μετρήσεις που έχουν γίνει οι πιο αποδοτικές μέθοδοι εκμάθησης για αυτοματοποιημένους πράκτορες από τις παραπάνω είναι η παρατηρητική και η αναλογική ή και συνδυασμός αυτών, σε σχέση με τις άλλες δύο που είχαν πολύ χειρότερα αποτελέσματα.

Δυναμική επιλογή στρατηγικής (επιλογή νέας στρατηγικής)

Ήδη συζητήσαμε στο στάδιο «Ορισμός πρωτοκόλλου και επιλογή αρχικής στρατηγικής» σχετικά με την επιλογή στρατηγικής και την σημασία της. Οι αποτελεσματικοί διαπραγματευτές κάνουν μία συνειδητή ανάλυση τόσο της κατάστασης της διαπραγμάτευσης όσο και των αντιπάλων και επιλέγουν ενεργά μία αρχική στρατηγική που αντιστοιχεί στην κρίση που κάνανε. Ωστόσο, είναι πολύ συνήθης πρακτική -επίσης των επιτυχημένων διαπραγματευτών- να ανανεώνουν την κρίση τους καθώς η διαπραγμάτευσης εκτυλίσσεται. Η πληροφορία που παίρνουν κατά την διάρκεια πολύ συχνά προκαλεί αλλαγές στην αντίληψη που είχαν για την δομή της κατάστασης της διαπραγμάτευσης και για την γνώμη τους για τα άλλα μέρη. Επομένως, οι διαπραγματευτές μπορεί να κινούνται μπρος και πίσω ανάμεσα σε διαφορετικές στρατηγικές ανάλογα την κατάσταση.

Όπως ήδη αναφέραμε (βλ. Ορισμός πρωτοκόλλου και επιλογή αρχικής στρατηγικής) υπάρχουν 3 διαφορετικές ευρείες κατηγορίες στρατηγικών, κάτω από τις οποίες εντάσσεται ένα τεράστιο ποσοστό στρατηγικών. Μία πολύ συνηθισμένη τακτική που ακολουθείται στις διαπραγματεύσεις είναι μία που αποτελείται από δύο διαδοχικά στάδια – το πρώτο περιλαμβάνει μία μίξη ανταγωνιστικής και ενδοτικής στρατηγικής, ενώ το δεύτερο περιλαμβάνει μία αμιγώς problem solving στρατηγική. Τυπικά στην αρχή οι διαπραγματευτές ξεκινούν με πολλές απαιτήσεις (ανταγωνιστικές τακτικές) για να δημιουργήσουν μία αυστηρότητα γύρω από τα σημεία-κλειδιά που τους ενδιαφέρουν και να πείσουν τους αντιπάλους να κινηθούν προς αυτά. Ταυτόχρονα κάνουν κάποιες λίγες, εύκολες υποχωρήσεις ώστε να δείξουν καλή θέληση και να κρατήσουν την διαπραγμάτευση ζωντανή. Σε αυτό το

στάδιο μπορεί να φαίνεται ότι η διαπραγμάτευση θα πέσει σε αδιέξοδο, καθώς τα διάφορα μέρη διεκδικούν με ζήλο το στόχο τους και παράλληλα οι υποχωρήσεις δεν είναι αρκετές για να φτάσουν σε μία κοινή απόφαση, εφόσον αυτές θέτουν σε κίνδυνο τους πόρους του διαπραγματευτή (π.χ. χρόνος, χρήμα, κλπ). Για να βγουν από το φαινομενικό αδιέξοδο οι διαπραγματευτές μεταβαίνουν στο στάδιο όπου πλέον προσπαθούν σε πνεύμα συνεργασίας να λύσουν το πρόβλημα (problem solving τακτικές), κάνοντας brainstorming, ανταλλάσσοντας προτάσεις, πληροφορίες και συμφωνώντας στους όρους μιάς λύσης.

Η τακτική αυτή είναι πολύ συνηθισμένη στις ανθρώπινες διαπραγματεύσεις, π.χ. συχνά απαντάται σε διεθνείς διαπραγματεύσεις. Στις αυτοματοποιημένες διαπραγματεύσεις ωστόσο η έννοια της δυναμικής στρατηγικής δεν είναι εξίσου δημοφιλής. Συνήθως η στρατηγική αντιμετωπίζεται σαν κάτι στατικό καθόλη την διάρκεια της διαπραγμάτευσης. Λίγες μόνο προσπάθειες έχουν γίνει σε αυτή την κατεύθυνση, ενώ τα περισσότερα μοντέλα δεν υποστηρίζουν την επιλογή νέων στρατηγικών κατά την διάρκεια της διαδικασίας.

Αδιέξοδο αποτέλεσμα

Αδιέξοδο είναι το σημείο ή η κατάσταση της διαπραγμάτευσης όπου δεν υπάρχει προφανής και εύκολη λύση – το μέρη είναι ανίκανα να δημιουργήσουν αμοιβαία επωφελείς προτάσεις που ταυτόχρονα ικανοποιούν τις επιθυμίες τους. Ο παραγωγικός διάλογος σταματά και, ακόμα και αν η επικοινωνία συνεχιστεί, συνήθως εξαντλείται στο πόσο παράλογες είναι η προτάσεις των διαφόρων μελών, πόσο μη-συνεργάσιμα είναι και καταλήγει σε μία προσπάθεια επιβολής απόψεων. Στο στάδιο αυτό τα διάφορα μέρη φαίνεται να πιστεύουν ότι δεν υπάρχει συμβατότητα και δεν είναι δυνατόν να βρεθεί μία κοινή λύση.

Σε αυτό το σημείο τα διάφορα μέρη πρέπει να κάνουν μαζί κάποιες ενέργειες για να μπορέσει η διαπραγμάτευση να πάει ένα βήμα πίσω, όταν ακόμα μπορούσε να κυλήσει παραγωγικά. Ενδεδειγμένες τεχνικές για την επίλυση έντονα πολωμένων αδιεξόδων είναι οι ακόλουθες:

Να αποκλιμακώσουν την επιθετικότητα και να βελτιώσουν την ακρίβεια της επικοινωνίας

Να κρατήσουν τον αριθμό των ανοιχτών ζητημάτων υπό έλεγχο έτσι ώστε να μπορέσουν να είναι διαχειρίσιμα, τα μεγάλα ζητήματα καλό είναι να σπάνε σε μικρότερα και καινούργια θα πρέπει να μπαίνουν με πολύ μεγάλο σύνεση

Να καταλήξουν σε κάποια κοινά της λύσης (π.χ. με ποιο τρόπο θα μπορούσαν να προσεγγίσουν την λύση ή να θέσουν χρονικούς περιορισμούς)

Να κάνουν τις προτάσεις τους πιο ελκυστικές για κοινή επίλυση (π.χ. να κατανοήσουν τις ανάγκες των αντιπάλων και να προσπαθήσουν να κάνουν προτάσεις με σεβασμό ως προς τις απαιτήσεις τους)

Αυτές οι τεχνικές φυσικά θα μπορούσαν να χρησιμοποιηθούν και από τα ίδια τα μέρη κατά την διάρκεια της διαπραγμάτευσης για αυτοβελτίωση, έτσι ώστε να έχουν καλύτερες πιθανότητες να επιλυθεί το πρόβλημα και να αποφευχθεί η αδράνεια. Ωστόσο αυτά δεν είναι εύκολο πάντα να εφαρμοστούν από τους συμμετέχοντες της διαπραγμάτευσης, επομένως μπορεί να υπάρξει ανάγκη για παρέβμαση κάποιου ενδιάμεσου (third-party) φορέα. Η παρέμβαση θα έπρεπε να αποφεύγεται σε καταστάσεις όπου η διαπραγμάτευση εξελίσσεται ανενόχλητη ή/και φαίνεται να μπορεί να καταλήξει κάπου μέσα σε φυσιολογικά πλαίσια χρόνου και πόρων. Σε καταστάσεις όμως αδιεξόδου μπορεί να αποδειχτεί μία πολύ αποδοτική μέθοδος για να επαναφέρει διαπραγματεύσεις σε έναν δρόμο με υποσχόμενο ορίζοντα.

Η μεσολάβηση είναι η πιο γνωστή και συχνά χρησιμοποιούμενη μέθοδος εξωτερικής παρέμβασης. Αυτό που συνήθως κάνουν είναι να συναντούν τα διάφορα μέρη ξεχωριστά, εξασφαλίζοντας ότι έχουν μία καλή εικόνα των ζητημάτων που διακυβεύονται, να αναγνωρίσουν τις πιθανές περιοχές συμβιβασμού και να προσπαθήσουν να επιτύχουν υποχωρήσεις προς μία συμφωνία. Οι ενδιάμεσοι μεσολαβητές δεν έχουν επίσημη επιρροή στο αποτέλεσμα και δεν μπορούν να λύσουν την διαφωνία μόνοι τους ή να επιβάλουν κάποια λύση. Δεν θα μπούμε σε λεπτομέρειες σχετικά με την γενική δομή του μοντέλου που ακολουθούν οι μεσολαβητές, αν και υπάρχουν και εδώ διακριτά στάδια που συνήθως ακολουθούνται. Αυτό που αξίζει να αναφέρουμε είναι ότι συνολικά οι μεσολάβηση φαίνεται να είναι αποτελεσματική σαν τεχνική και το αποτέλεσμα που προκύπτει είναι κατά κανόνα αποδεκτό. Η μεσολάβηση είναι πολύ πιο πιθανό να επιτύχει όταν πληρούνται κάποιες προϋποθέσεις, όπως το να είναι όλα τα μέρη σύμφωνα με την μεσολάβηση, όλα τα μέρη έχουν την ίδια δύναμη και επιρροή στο αποτέλεσμα, τα ζητήματα δεν αφορούν κάποια γενική αρχή και τα μέρη έχουν σημαντικό κίνητρο να φτάσουν σε κάποια συμφωνία.

Στις περισσότερες προσεγγίσεις που γίνονται, η εστίαση είναι στην κατανόηση και προτυποποίηση μίας επιτυχημένης διαπραγμάτευσης – οι ερευνητές παίρνουν σαν δεδομένο ότι οι διαπραγματεύσεις θα καταλήγουν σε συμφωνία, δεν γίνονται δύσκολες ούτε τραβάνε την αυστηρότητα μίας στάσης έως σημείο αδιαξόδου. Για το λόγο αυτό οι μελέτη «δύσκολων» διαπραγματεύσεων και η επίλυση ή αποφυγή αδιεξόδων είναι σε νηπιακό στάδιο. Από τις λίγες προσπάθειες που έχουν γίνει μπορούμε να πούμε ότι η πολυπλοκότητα είναι μεγάλη, γιατί ο αυτόματος πράκτορας θα πρέπει να επισκέπτεται τακτικά τα ζητήματα και να τα ανανεώνει κατάλληλα κάθε φορά που η διαπραγμάτευση έχει φτάσει ή τείνει να καταλήξει σε αδιέξοδο.

Επίλυση (ανάλυση και βελτίωση του τελικού αποτελέσματος)

Στο στάδιο αυτό συνήθως οι συμμετέχοντες είναι χαρούμενοι που βρέθηκε ένα αποτέλεσμα το οποίο και τους ικανοποιεί (εφόσον συμφώνησαν ήδη σε αυτό). Ένας εκ των υστέρων απολογισμός ωστόσο μπορεί να δείξει ότι η διαπραγμάτευσης εν τέλει κόστισε περισσότερο από ότι έδωσε πίσω, μπορεί να αποκαλυφθεί ότι ένα μέρος ευνοήθηκε περισσότερο από κάποιο άλλο, ότι σημεία-κλειδιά αγνοήθηκαν (για κάποια ή και για όλα τα μέρη), μπορεί να προκύψουν νέες καταστάσεις που αλλάζουν τα δεδομένα και γενικά να αποδειχτεί ότι η συμφωνία είναι προβληματική με κάποιον τρόπο για ένα ή περισσότερα από τα συμμετέχοντας μέρη της διαπραγμάτευσης. Στην περίπτωση αυτή η διαπραγμάτευση συνήθως ανοίγει ξανά με κάποια ή και όλα τα ζητήματα που ήταν υπό συζήτηση.

Οι ερευνητές τεχνητής νοημοσύνης παραδοσιακά επικεντρώνονται στην μελέτη της συμφωνίας που περιλαμβάνει πολλαπλά επίπεδα δέσμευσης και κυρώσεις σε περίπτωση αποδέσμευσης. Οι περισσότεροι δημιουργούν δεσμεύσεις που δεν σπάνε, δηλαδή μοντέλα που δεν μπορούν με κανέναν τρόπο να αποδεσμευτούν από μία δέσμευση που έκαναν. Αυτό φυσικά θέτει διάφορα προβλήματα, καθώς μελλοντικές καταστάσεις μπορεί να προκύψουν που απαιτούν τέτοια αντίδραση και όντως υπάρχουν κάποιες λίγες εργασίες που επιτρέπουν την αποδέσμευση του διαπραγματευτή με σκοπό να εξυπηρετήσει μελλοντικά γεγονότα. Συνολικά πάντως λίγες είναι οι προσπάθειες ανάπτυξης μοντέλων που μπορούν να κάνουν αποτελεσματική επαναδιαπραγμάτευση.

Τέλος, είναι σημαντικό να σημειωθεί ότι οι διάφορες μελέτες που έχουν γίνει δίνουν κάθε φορά βάρος σε διαφορετικό κομμάτι της διαπραγμάτευσης (πράγμα που λίγο έως πολύ επηρεάζει και την αποδόμηση της διαδικασίας σε στάδια που θα ακολουθήσουν). Υπάρχει περίπτωση για παράδειγμα κάποια έρευνα να εστιάζει μόνο στο κομμάτι της επιχειρηματολογίας και κατά πόσο αυτή είναι αποτελεσματική, ενώ μία άλλη μελέτη να αγνοεί τελείως το κομμάτι των επιχειρημάτων. Κάποιος μπορεί να επικεντρώνεται στο στάδιο της εκμάθησης και πως αυτό επηρεάζεται από νέα πληροφορία και κάποιος άλλος να ασχολείται αποκλειστικά με την στρατηγική. Κι εμείς με την σειρά μας λοιπόν, με γνώμονα το δικό μας πρόβλημα και το περιορισμένο οικοσύστημα στο οποίο θα δουλέψουμε έχουμε ξεχωρίσει κάποια στάδια στα οποία θα σταθούμε λίγο περισσότερο, κάποια θα τα προσπεράσουμε γρήγορα θεωρώντας τα δεδομένα (υποθέσεις, απλουστεύσεις, κλπ.) και κάποια θα τα αγνοήσουμε τελείως θεωρώντας ότι δεν έχουν εφαρμογή στο πρόβλημα που επιχειρούμε να λύσουμε. Σε γενικές γραμμές πάντως, θα πατήσουμε πάνω σε αυτό το πλαίσιο διαπραγμάτευσης για να περιγράψουμε την διαπραγματευτική διαδικασία στα πλαίσια του DeepGraphs specification και να στην συνέχεια να υλοποιήσουμε έναν τέτοιο πράκτορα. Περισσότερες λεπτομέρειες σχετικά με την ακριβή μεθοδολογία, τις υποθέσεις και

τους περιορισμούς που τέθηκαν στα πλαίσια αυτής της εργασίας βρίσκονται στο αντίστοιχο κεφάλαιο (Κεφάλαιο 6).

3

Βασικές Τεχνολογίες

Εδώ περιγράφουμε τεχνολογίες και έννοιες που είναι απαραίτητες για την κατανόηση της παρούσας εργασίας, όπως η αρχιτεκτονική REST, ο σημασιολογικός ιστός και τα συμβόλαιο στο διαδίκτυο.

3.1 Εισαγωγή

Το World Wide Web είναι σαν ένα ενιαίο client-server σύστημα. Οι clients (χρήστες) έχουν πρόσβαση σε αρχεία κειμένου μέσω υπερκειμένου (hypertext) τα οποία εντοπίζονται από Universal Resource Identifiers(URI) σε servers που βρίσκονται διάσπαρτοι στο Internet και επικοινωνούν μέσω του Hypertext Transfer Protocol (HTTP). Οι συνδέσεις (links) μαζί με διάφορες ανακαλύψεις που έγιναν περίπου στον χώρο των δικτύων παράλληλα, έπαιξαν κομβικό ρόλο για την επιτυχία και υιοθέτηση της τεχνολογίας του Web. Τα Universal Resource Identifiers (URIs) αποτέλεσαν έναν βασικό μηχανισμό, ώστε να εμπλουτίσουν κείμενα με αναφορές σε άλλα σχετικά κείμενα.

Η κύρια διαφορά ανάμεσα στη συγκεκριμένη τεχνολογία υπερκειμένου και τις υπάρχουσες που υπήρχαν τη συγκεκριμένη χρονική περίοδο, μπορεί να τοποθετηθεί στην απόφαση να είναι οι σύνδεσμοι προς μια μόνο κατεύθυνση, χωρίς να υπάρχει απαίτηση διπλής, όπως ήταν ως τότε. Πρακτικά, αυτό σημαίνει ότι είναι αδύνατο να αποφύγει κανείς τα σπασμένα links (υπερσυνδέσμους), όταν πλέον αυτά δεν είναι διαθέσιμα για προσπέλαση. Αν και αυτό

δημιουργεί σίγουρα αρκετά «σκουπίδια» από άχρηστα links, το οποίο προφανώς ανήκει στα αρνητικά αυτής της απόφασης σχεδιασμού, τα υπέρ ξεπερνούν κατά πολύ τα οποιαδήποτε αρνητικά. Αυτή, ίσως, είναι άλλωστε μια από τις βασικές σχεδιαστικές καινοτομίες που βοήθησαν το Web να κάνει scale, δηλαδή να μεγαλώσει στο μέγεθος που βλέπουμε σήμερα. Εκτός από το ότι εξαφανίζουν τους ελέγχους των διαφόρων integrity constraints, αφού πλέον δεν χρειάζεται κανείς να ελέγχει τη διπλή κατεύθυνση των συνδέσμων, η συγκεκριμένη ευκολία κάνει την υλοποίηση servers και clients για το Web μια αρκετά απλή διαδικασία. Επίσης, η μη-αμφίδρομη σύνδεση πλέον έδωσε το έναυσμα για την αποκεντροποιημένη, μη κεντρικά συντονισμένη δημιουργία και ανταλλαγή δεδομένων. Αυτή η αποκεντροποίηση ήταν ο βασικός και κύριος λόγος για τον οποίο το Web κατάφερε σε τόσο σύντομο χρονικό διάστημα να επισκιάσει όλα τα προηγούμενα συστήματα υπερκειμένου και φυσικά του επέτρεψε να μεγαλώσει όπως ποτέ τίποτα προηγουμένως στην ιστορία.

Για ένα μεγάλο χρονικό διάστημα, η αρχιτεκτονική του Web καθώς και οι τεχνολογίες του δεν ήταν καταλλήλως τυποποιημένες. Τα έγγραφα οδηγιών αποτελούνταν κυρίως από ανεπίσημες σελίδες ιστού[11] , προσχέδια της αναφοράς[12] (specification) και τέλος από τον πηγαίο κώδικα του client και του server που είχαν υλοποιηθεί και κυκλοφορήσει από το CERN. Μιας και αυτά τα έγγραφα δεν ήταν πάντα σε πλήρη συμμόρφωση με τις διάφορες υλοποιήσεις που έτρεχαν, έγινε ολοένα και πιο δύσκολο να χτιστούν διαλειτουργικά συστήματα. Αναπόδραστα, αυτό όλο κατέληξε στην απαίτηση της βιομηχανίας να ζητάει για standardization των βασικών τεχνολογιών, και την άμεση δημιουργία working groups που κατέληξαν σε σταθερές περιγραφές των specifications.

Χωρίς, να μπούμε σε συγκεκριμένες λεπτομέρειες ουσιαστικά από εκεί και πέρα ξεκίνησε η μεγάλη ανάπτυξη του world wide web και η εξέλιξή του σε αυτό που ξέρουμε σήμερα. Από τη στιγμή που άρχισαν οι συζητήσεις για δημιουργία standards, κανόνων δηλαδή που θα πρέπει να ακολουθούνται από όλους για τον σχεδιασμό και την δημιουργία σελίδων, ξεκίνησε το internet να μεγαλώνει ομοιόμορφα και δομημένα. Οι κανόνες ή υποδείξεις αυτές μπορεί να αφορούσαν την ονοματοδοσία των σελίδων, το συντακτικό των URIs (Uniform Resource Identifier) ή την δομή των HTML (Hypertext Markup Language) σελίδων, κάποια από τα οποία ισχύουν έως και σήμερα. Επίσης σημαντική ήταν η δημιουργία μίας σαφούς ορολογίας, στην οποία προσδιορίζονται όροι όπως «πόρος» (resource) που φαίνεται να είναι εξαιρετικά δημοφιλής ακόμα και σήμερα.

Όλα αυτά κατέληξαν σε πολλές από τις τεχνολογίες που γνωρίζουμε σήμερα και χρησιμοποιούμε, όπως η HTML 2.0 και τα πρωτόκολλα HTTP (Hypertext Transfer Protocol) 1.0 και 1.1. Στα μέσα της δεκαετίας του ενενήντα ,ο Roy T. Fielding, ένας από τους συγγραφείς των URI και HTTP specifications, άρχισε να δουλεύει σε ένα αρχιτεκτονικό μοντέλο για το πως θα μπορούσε να λειτουργεί το Web, έτσι ώστε να εξυπηρετεί σαν οδηγός

για τα Web protocol standards[20]. Το αποτέλεσμα αυτής της εργασίας ήταν ένα αρχιτεκτονικό στυλ, το Representational State Transfer Protocol (REST[21]). Σύμφωνα με τον ίδιο, το REST καλύπτει όλες τις πτυχές ενός διασπαρμένου διασυνδεδεμένου δικτύου, οι οποίες θεωρούνται βασικές για την συμπεριφορά και την απόδοση ενός συστήματος, όπως το διαδίκτυο. Μιας και το REST χρησιμοποιείται ως επί το πλείστων σήμερα για τον σχεδιασμό και την ανάπτυξη της αρχιτεκτονικής του μοντέρνου διαδικτύου, θα το συζητήσουμε με περισσότερες λεπτομέρειες στο επόμενο κεφάλαιο.

3.2 Representational State Transfer Architectural Style (REST)

Το REST είναι ένα αρχιτεκτονικό στυλ που προτάθηκε ώστε να συγκεκριμενοποιήσει τους διάφορους περιορισμούς, την επεκτασιμότητα, την αξιοπιστία και τέλος να ορίσει ένα αφαιρετικό επίπεδο των πόρων μέσα σε ένα διάσπαρτο σύστημα υπερμέσων. Αγνοεί τις λεπτομέρειες υλοποίησης, καθώς και το συντακτικό των πρωτοκόλλων, με σκοπό να δώσει έμφαση στους διάφορους ρόλους των επιμέρους εξαρτημάτων ενός συστήματος, την αλληλεπίδραση αυτών με άλλα εξαρτήματα και την απεικόνιση των δεδομένων που ανταλλάσσουν. Οι σχολαστικές επιλογές και οι συμβιβασμοί στη σχεδίαση, απελευθερώνουν τη δημιουργία επεκτάσιμων, διατηρήσιμων, εξελίξιμων και αποσυνδεδεμένων (χωρίς ιδιαίτερες εξαρτήσεις δηλαδή) και διάσπαρτων συστημάτων στην κλίμακα του Internet. Πρέπει να αναγνωριστεί ότι το Web, το μεγαλύτερο και πιο επιτυχημένο διασπαρμένο σύστημα που έχει χτιστεί ποτέ, βασίζεται στις αρχές και τις αξίες του REST, το κάνει να ξεχωρίζει και προσδίδει αρκετά από τα μοναδικά του ανώτερα χαρακτηριστικά.

Το REST βασίζεται σε ένα κλασικό μοντέλο αρχιτεκτονικής client-server, στα οποία ο server προσφέρει έναν αριθμό από υπηρεσίες, με τις οποίες ο client μπορεί να αλληλοεπιδράσει με το να στέλνει αιτήματα(requests) στον server. Η ώθηση για μια τέτοια αρχιτεκτονική, είναι ξεκάθαρα ένας διαχωρισμός αρμοδιοτήτων που απλοποιεί την υλοποίηση δυο συστημάτων, το οποίο συχνά καταλήγει στη θετική συνέπεια του να αυξάνει την επεκτασιμότητα του server και να επιτρέπει την ανεξάρτητη εξέλιξη του κάθε συστήματος, όσο φυσικά η διεπαφή ανάμεσά τους παραμένει σταθερή. Το REST προσθέτει έναν αυστηρό κανόνα ανάμεσα στους περιορισμούς του, ότι ο server πρέπει να είναι χωρίς κατάσταση(stateless), δηλαδή δεν είναι υπεύθυνος για τις αλλαγές καταστάσεων. Αυτό έχει σαν άμεση συνέπεια κάθε αίτηση από τον client προς τον server να πρέπει να περιέχει όλη την πληροφορία, ώστε να μπορέσει ο δεύτερος να καταλάβει τι του ζητείται. Με άλλα λόγια, ένας client δεν μπορεί να εκμεταλλευτεί οποιαδήποτε εσωτερική κατάσταση του server, ώστε να εξυπηρετηθεί ή να αποκτήσει παραπάνω πληροφορίες. Ο server φυσικά γνωρίζει για την κατάσταση των resources του, αλλά δεν κρατάει συγκεκριμένο ιστορικό των αλληλεπιδράσεών του με τους

clients. Όλες οι καταστάσεις και οι αλλαγές αυτών πρέπει να βρίσκονται εξ' ολοκλήρου στον client. Αυτό είναι ένα πολύ σημαντικό συστατικό, καθώς διευκολύνει την παρακολούθηση και την καταγραφή στα συστήματα εξαιτίας των ξεκάθαρων και διακριτών αλληλεπιδράσεων. Επίσης, αυξάνει την αξιοπιστία καθώς όλη η πληροφορία της αλληλεπίδρασης βρίσκεται μέσα στην αίτηση. Έτσι είναι ευκολότερο να βρεθεί και να επιλυθεί οποιοδήποτε πρόβλημα. Επιπροσθέτως, η επεκτασιμότητα βελτιώνεται αισθητά, καθώς ο server πλέον δεν έχει την υποχρέωση να αποθηκεύει ενδιάμεσες καταστάσεις. Κατά συνέπεια, απελευθερώνει γρηγορότερα resources που πλέον δεν χρειάζεται και απλοποιεί την υλοποίηση τέτοιων συστημάτων, καθώς ο server δεν χρειάζεται πλέον να διαχειρίζεται τη χρήση των διαφόρων πόρων ανάμεσα σε διαδοχικές αιτήσεις του client. Είναι ακριβώς αυτό το στοιχείο που κάνει εφικτό το load balancing, μια ευρέως χρησιμοποιούμενη τεχνική, για να εξακριβωθεί η ποιότητα μιας εφαρμογής. Στα αρνητικά του συγκεκριμένου stateless χαρακτηριστικού που ορίζει το REST, είναι η μειωμένη απόδοση του δικτύου, λόγω της επανάληψης δεδομένων ανάμεσα στην αλληλεπίδραση client-server, καθώς κάθε φορά πρέπει να στέλνουν όλη την πληροφορία, ενώ θα ήταν πιο βολικό και αποδοτικό ο server να κράταγε ενδιάμεσες καταστάσεις με αυτή την πληροφορία. Επιπλέον, μειώνει αισθητά τον έλεγχο που έχει ο server στην συμπεριφορά της εφαρμογής καθώς ο ίδιος μοιράζει την προσοχή του ανάμεσα σε πολλές εφαρμογές με πιθανώς και διαφορετικούς σκοπούς.

Για να μετριαστεί ο παραπάνω φόρτος που φέρνει μαζί του ο ανωτέρω περιορισμός στα RESTful συστήματα έχει, επίσης, προστεθεί υποστήριξη για caching. Ο περιορισμός για cache ουσιαστικά δηλώνει ότι κάθε πληροφορία που ανταλλάσσεται θα φέρει μαζί της και μια ετικέτα για το αν είναι cacheable or non-cacheable. Αυτό βελτιώνει φανερά τον αριθμό των αιτήσεων προς τον server σε μικρότερες αιτήσεις, που δείχνουν ότι κάτι δεν έχει αλλάξει στα δεδομένα από την τελευταία φορά που έγινε η ίδια αίτηση. Αυτό έχει θετικά αποτελέσματα στην επίδοση και επεκτασιμότητα των RESTful συστημάτων και βελτιώνει την εμπειρία του χρήστη με το να ελαττώνει δραστικά την αναμονή. Το αρνητικό, είναι ότι η αξιοπιστία του συστήματος μπορεί να μειωθεί εξαιτίας του γεγονότος ότι κάποιες φορές οι πληροφορίες δεν έχουν ανανεωθεί και άρα είναι ξεπερασμένες.

Η έμφαση που δίνει το REST σε μια ενιαία διεπαφή ανάμεσα στα διάφορα υποσυστήματα που συνθέτουν το μεγαλύτερο, είναι το βασικότερο χαρακτηριστικό που το διαχωρίζει από τα υπόλοιπα στυλ. Απλοποιεί αρκετά ολόκληρη την αρχιτεκτονική του συστήματος και βελτιώνει την εποπτεία στην αλληλεπίδραση των διαφόρων υποσυστημάτων. Με το να ξεχωρίζει τις υλοποιήσεις από τις υπηρεσίες που παρέχουν, η εξέλιξη αυτών βελτιώνεται με το κόστος της μειωμένης απόδοσης που αναφέραμε ήδη σε σχέση με τα υπέρ-εξειδικευμένα συστήματα για έναν και μόνο σκοπό. Το REST δηλώνει τέσσερις περιορισμούς διεπαφής ώστε να εξασφαλίσει ένα ενιαίο μέτωπο διεπαφής:

- αναγνώριση των πόρων (identification of resources).
- διαχείριση των πόρων μέσω αναπαραστάσεων (manipulation of resources through representation).
- μηνύματα που περιέχουν όλη την πληροφορία (self-descriptive messages).
- υπερμέσα σαν το βασικό μέσο για τη μετάβαση καταστάσεων
- (hypermedia as the engine of application state).

Το REST είναι μια αντικειμενοστραφής αρχιτεκτονική, με την έννοια ότι η κύρια αφαίρεση στην πληροφορία μέσω του REST είναι το αντικείμενο, πόρος. Κάθε έννοια μπορεί κανείς να την φανταστεί σαν ένα αντικείμενο. Ο Fielding ορίζει ένα αντικείμενο R σαν μια συνάρτηση προσωρινής "συνδρομής" Mn(t), η οποία για έναν χρόνο t, δείχνει σε ένα σύνολο από οντότητες ή τιμές οι οποίες είναι μεταξύ τους ισοδύναμες. Ο τρόπος που το REST διαφοροποιεί τα αντικείμενα, προϋποθέτει ότι τα αντικείμενα είναι αναγνωρίσιμα, ώστε να μπορούν να προσπελαστούν και να μεταβληθούν μέσα από γενικές διεπαφές. Στο Web, τα αντικείμενα είναι προσπελάσιμα από IRIs[22]. Μιας και τα διάφορα αντικείμενα μπορεί να απεικονίζουν ιδέες οι οποίες να μην είναι δυνατό άμεσα να αποτυπωθούν σε μια αλληλουχία από bytes (όπως για παράδειγμα ένας άνθρωπος ή ένα συναίσθημα), οι συγκριμένες απεικονίσεις δεν διαχειρίζονται άμεσα. Αντιθέτως, το REST έχει δομηθεί με τη φιλοσοφία ότι η διαχείριση των αντικειμένων θα γίνεται μέσω απεικονίσεων αυτών, οπότε υιοθετείται ένα παραπάνω επίπεδο αλληλεπίδρασης. Μια απεικόνιση είναι μια ακολουθία από bytes συνοδευόμενη από κάποια μεταπληροφορία (metadata). Τα Media types προτυποποιούν τον τρόπο απεικόνισης δεδομένων στο Web. Δεδομένου, όπως αναφέρθηκε, ότι η επικοινωνία μεταξύ των διαφόρων συστημάτων στο web είναι stateless και ότι τα μηνύματα που ανταλλάσσουν μεταξύ τους είναι συγκεκριμένης μορφής που έχει προτυποποιηθεί, το REST "επιβάλλει" μηνύματα τα οποία είναι πλήρως αυτο-περιγραφόμενα με αποτέλεσμα να προσφέρει τη δυνατότητα να διαχειριστούν και να περιγραφούν και από ενδιάμεσα συστήματα χωρίς να χρειάζεται περαιτέρω γνώση του κόσμου. Το μόνο κομμάτι που λείπει για να τελειώσουμε την περιγραφή του REST σαν ενιαία διεπαφή είναι το τελευταίο που αναφέραμε πριν, δηλαδή τα υπερμέσα (hypermedia), βασικό μέσο για τη μετάβαση καταστάσεων. Αυτό αναφέρεται στην χρήση των hypermedia στις απεικονίσεις των αντικειμένων σαν ένας τρόπος πλοήγησης ανάμεσα στις καταστάσεις κατάστασης μιας εφαρμογής. Αν και ο συγκεκριμένος περιορισμός είναι αυτός που δίνει την δυνατότητα τα συστήματα να χτίζονται δυναμικά και να είναι μεταξύ τους ανεξάρτητα, είναι και αυτός που έχει κατανοηθεί λιγότερο και κατ' επέκταση είναι αυτός που σπάνια υλοποιείται σωστά.

Πολλά συστήματα, ανεξάρτητα αν δηλώνουν RESTful ή όχι, βασίζονται έντονα σε έναν αφανή, σιωπηρό τρόπο να δηλώνουν τη ροή πληροφορίας και καταστάσεων, που είναι χαρακτηριστικό στα απομακρυσμένα συστήματα κλήσης (Remek Procedure Call - RPC). Τα μηνύματα που επιτρέπεται από την υλοποίηση να ανταλλαχθούν καθώς και το πως αυτά

ερμηνεύονται, επηρεάζονται από τα προηγούμενα μηνύματα, υποδηλώνοντας έμμεσα(σιωπηρά) την κατάσταση στην οποία βρίσκεται ένα σύστημα.

Τρίτοι που προσπαθούν να κατανοήσουν την κουβέντα δύο συστημάτων, χρειάζονται όλο τον πίνακα καταστάσεων και μεταβάσεων και φυσικά την αρχική κατάσταση επικοινωνίας, κάτι το οποίο είναι αρκετά δύσκολο και ουσιαστικά μη πρακτικό. Αυτό, επίσης, καθιστά εξαιρετικά δύσκολο το να ανακτηθεί πληροφορία σε περίπτωση που κάτι πάει στραβά σε μεγάλα απομακρυσμένα συστήματα.

Για να λυθεί αυτό το πρόβλημα, η χρήση των hypermedia είναι μια από τις απαραίτητες συνθήκες της REST αρχιτεκτονικής. Σύμφωνα με τον Fielding[23], "ένα REST API πρέπει να μπορεί να διατρέχεται χωρίς τίποτα παραπάνω από την γνώση της αρχικής διεύθυνσης αυτού (URI) και ένα σύνολο από προτυποποιημένα media types. […] Από αυτό το σημείο και πέρα, όλες οι μεταβάσεις του συστήματος πρέπει να οδηγούνται από επιλογές του client σε αυτά που επιτρέπει ο server, οι οποίες είτε είναι εμφανής από την απάντηση του είτε έχουν ειπωθεί σε προηγούμενη αλληλεπίδραση και δηλώνονται έμμεσα." Το web αξιοποιεί αυτού του τύπου την αλληλεπίδραση και τη μετάβαση καταστάσεων, όταν πολύ λίγα είναι γνωστά από την αρχή, επιτρέποντας έτσι την διεσπαρμένη φύση αυτού. Οι άνθρωποι είναι εξαιρετικά ικανοί να προσαρμόζονται σε αλλαγές στη ροή πληροφορίας που δίνεται σε πραγματικό χρόνο, για παράδειγμα σε μια νέα σελίδα για πρόσβαση σε μια υπηρεσία ή αλλαγές στην παραγγελία σε ένα σύστημα.

Ο Παραστρατίδης[24], ορίζει το σύνολο των επιτρεπτών ενεργειών απαραίτητων για να επιτευχθεί ένας συγκεκριμένος στόχος για μια συγκεκριμένη εφαρμογή, σαν πρωτόκολλο συγκεκριμένης εφαρμογής μιας υπηρεσίας (domain application protocol of a service). Το πρωτόκολλο ορίζει τους κανόνες αλληλεπίδρασης μεταξύ δυο διαφορετικών συμμετεχόντων. Σαν συνέπεια αυτού, η κατάσταση ενός συστήματος είναι απλά μια απεικόνιση ενός συστήματος σε μια δεδομένη στιγμή. Αυτό συμπίπτει με τον ορισμό του Fielding για την κατάσταση ενός συστήματος την οποία ορίζει ως αιτήματα σε αναμονή, τοπολογία από συνδεδεμένα συστήματα, τα ενεργά αιτήματα ανάμεσα σε αυτά, η αλληλουχία καταστάσεων ανάμεσα σε αυτά και τέλος η επεξεργασία αυτών από τους πράκτορες του χρήστη (user agents). Συμπερασματικά, η συνολική κατάσταση του συστήματος συνοψίζεται στην κατάσταση του server και αυτή της εφαρμογής. Χρησιμοποιώντας, λοιπόν, την ιδέα του πρωτοκόλλου συγκεκριμένης εφαρμογής μιας υπηρεσίας, η φράση "υπερμέσα σαν το βασικό μέσο για την μετάβαση καταστάσεων " μπορεί τώρα να επεξηγηθεί ως η χρήση υπερμέσων σαν έλεγχος για να αποφασίζεται σε πραγματικό χρόνο εκτέλεσης η διαδοχή καταστάσεων, αντί για να υπάρχει μια στατική συμφωνία κατά τη σχεδίαση του συστήματος. Οι αλλαγές πλέον στο σύστημα μπορούν να εμφανίζονται στους χρήστες αυτού σε πραγματικό χρόνο. Αυτό φέρνει κάποιες από τις ανθρώπινες ικανότητες προσαρμογής στο web της

αλληλεπίδρασης των μηχανών και επιτρέπει την αρχιτεκτονική διασκορπισμένων συστημάτων που μπορούν να εξελίσσονται. Οπότε, αντί να χρειάζεται να κατανοήσουν ένα συγκεκριμένο σύστημα πως έχει χτιστεί, χρειάζεται να κατανοήσουν τη σημασιολογία ορισμένων επιχειρησιακών όρων στο οποίο το σύστημα βρίσκεται[25]. Δυστυχώς, τα σύγχρονα Web APIs σπάνια παρουσιάζουν τέτοια χαρακτηριστικά και είναι σχεδόν αδύνατο να πετύχουμε ένα αντίστοιχο επίπεδο προσαρμοστικότητας στο Web των μηχανών. Αυτό είναι και ένα από τα βασικά ζητήματα της παρούσας διατριβής.

Ενοχλημένος από την πραγματικότητα ότι πολλές υπηρεσίες δηλώνουν RESTful, αν και παραβιάζουν εμφανώς τη συνθήκη για hypermedia , ο Fielding ξεκαθάρισε ότι τα hypermedia είναι από τα βασικά και αναγκαία χαρακτηριστικά μιας RESTful αρχιτεκτονικής. Δεδομένου ότι ο όρος REST συχνά χρησιμοποιείται λάθος, η κοινότητα προσπάθησε μέσα από συζητήσεις να ορίσει έναν νέο όρο για να περιγράφει APIs που σέβονται αυτόν τον περιορισμό, όπως πχ Hypermedia APIs.

Αν και όπως αναφέραμε η hypermedia συνθήκη συχνά παραβιάζεται ,η συνθήκη για επίπεδα στην απεικόνιση σχεδόν πάντα εφαρμόζεται σωστά, κάτι που βοηθάει στην ανεξαρτησία των επιμέρους συστημάτων μιας αρχιτεκτονικής. Οι συνθήκες επιβάλουν ότι τα RESTful συστήματα συνθέτονται από ιεραρχικά επίπεδα, στα οποία τα συστήματα ενός συγκεκριμένου επιπέδου παρέχουν υπηρεσίες μόνο για τα συστήματα του παραπάνω επιπέδου και καταναλώνουν υπηρεσίες του κάτω. Αυτό περιορίζει τη γνώση που υπάρχει σε κάθε επίπεδο, και θα έλεγε κανείς ότι περιορίζει αντίστοιχα, θέτοντας ένα πάνω όριο, και στο επίπεδο πολυπλοκότητας του συστήματος. Πηγαίνοντας ένα βήμα παραπέρα, επιτρέπει την εισαγωγή των load balancers, caches, firewalls, gateways και proxies σε διάφορα συστήματα για το σύστημα πιο ευέλικτο σε αλλαγές χωρίς να χρειάζεται να αλλάζουν τις διεπαφές. Προφανώς, προσθέτοντας παραπάνω συστήματα αυξάνεται και η υπολογιστική πολυπλοκότητα, καθώς και η καθυστέρηση των απαντήσεων, που καταλήγει σε χαμηλότερη εμπειρία του χρήστη. Αυτό προφανώς μπορεί να επιλυθεί προσθέτοντας ενδιάμεσες caches.

Τέλος, το REST έχει μια code-on-demand (κώδικας ανά απαίτηση) συνθήκη που επιτρέπει σε έναν client να επεκτείνει τις δυνατότητές του από κώδικα που φορτώνει δυναμικά. Το βασικό προσόν αυτής της συνθήκης είναι η επεκτασιμότητα των συστημάτων. Φαίνεται καλύτερα αυτή η συνθήκη από τις σύγχρονες web εφαρμογές που φορτώνουν δυναμικά javascript code για να υλοποιούν δυνατότητες, που δεν υπάρχουν πάντα μέσα στους web browsers. Δεν πρέπει βέβαια να ξεχνάμε ότι, φορτώνοντας δυναμικά κώδικα μειώνει δραματικά την απόδοση και ανοίγει πόρτες σε επιθέσεις ασφαλείας. Οπότε για αυτόν ακριβώς τον λόγο το συγκεκριμένο είναι προαιρετικό.

3.3 Εισαγωγή στον σημασιολογικό ιστό

Η έννοια του σημασιολογικού ιστού είναι πολύ απλή στη βάση της. Το διαδίκτυο είναι γεμάτο πληροφορία η οποία προορίζεται για τους χρήστες (κατεξοχήν) και επομένως μπορεί πολύ εύκολα να αφομοιωθεί από αυτούς. Για παράδειγμα σε μία ιστοσελίδα το περιεχόμενο της σελίδας θα είναι κάποιο κείμενο ή φωτογραφία που προορίζεται να θεαθεί ή/και να διαβαστεί από χρήστες και δεν έχει απολύτως κανένα νόημα στον εξυηρετητή που σερβίρει την σελίδα ή στον browser που την παρουσιάζει. Ακόμα και άλλου είδους αλληλεπιδράσεις με την ιστοσελίδα, όπως για παράδειγμα το πάτημα ενός κουμπιού, δεν έχουν κανένα απολύτως νόημα για τις μηχανές. Ενώ πιθανότατα πυροδοτούν μία απόκριση κάποιας μηχανής (κάποιου server) ή υπηρεσίας αυτό γίνεται μηχανιστικά και χωρίς «επίγνωση» της σημασίας της ενέργειας ή του περιεχομένου. Μία σχετικά καλή παρομοίωση θα ήταν με έναν ταχυδρόμο που μεταφέρει ένα γράμμα, ο οποίος ξέρει ότι έχει υποχρέωση ότι πρέπει να παραδώσει ένα γράμμα σε μία συγκεκριμένη διεύθυνση, αλλά δεν έχει καμία ιδέα γιατί αυτό το γράμμα έχει αποσταλεί ή ποιο είναι το περιεχόμενό του.

Όλα αυτά άλλαξαν με την εισαγωγή της έννοιας του σημασιολογικού ιστού στον χώρο του διαδικτύου. Μία πρώτη νύξη είχε γίνει από τον ίδιο τον εφευρέτη του internet (www protocol) Tim Burners Lee, ο οποίος είχε αντιληφθεί από την αρχή αυτή την «αποξένωση» των υπολογιστών από το περιεχόμενο του Internet και πρότεινε να προστεθεί σημασιολογία, η οποία θα είναι κατανοητή από μηχανές με το να επιτρέψει στα δεδομένα να έχουν πληροφορία σε δομές και μορφές αντιληπτές από μηχανές και να αφήσει τους συνδέσμους να έχουν συγκεκριμένες τιμές. Ο ίδιος μάλιστα έδωσε την ονομασία Σημασιολογικός Ιστός (Semantic Web) και έκανε την πρώτη προσπάθεια να προτυποποιήσει τις τεχνολογίες αυτού, οι οποίες κατέληξαν στο Resource Description Framework (RDF).

3.3.1 Τεχνολογίες του Σημασιολογικού ιστού

Χωρίς να επεκταθούμε πολύ, αξίζει να σημειωθούν κάποια πράγματα σχετικά με τις τεχνολογίες του Σημασιολογικού ιστού και συγκεκριμένα το RDF.

Όπως ήδη αναφέρθηκε ο σημασιολογικός ιστός είναι μία επέκταση του παρασοδιακού διαδικτύου, η οποία στοχεύει στην παρουσίαση πληροφορίας υπό την μορφή κειμένου φυσικής γλώσσας, αλλά και σε μορφή κατανοητή από τις μηχανές. Το RDF είναι το θεμέλιο πάνω στο οποίο βασίζεται ο σημασιολογικός ιστός.

Το RDF ορίζει ένα απλό μοντέλο που βασίζεται σε τριπλέτα, στο οποίο κάθε δήλωση αποτελείται από το subject, το predicate και τέλος το object (Βλ. Figure 3). Πολλές τέτοιες τριπλέτες χτίζουν έναν γράφο και πολλοί τέτοιοι γράφοι ένα σύνολο δεδομένων (dataset).

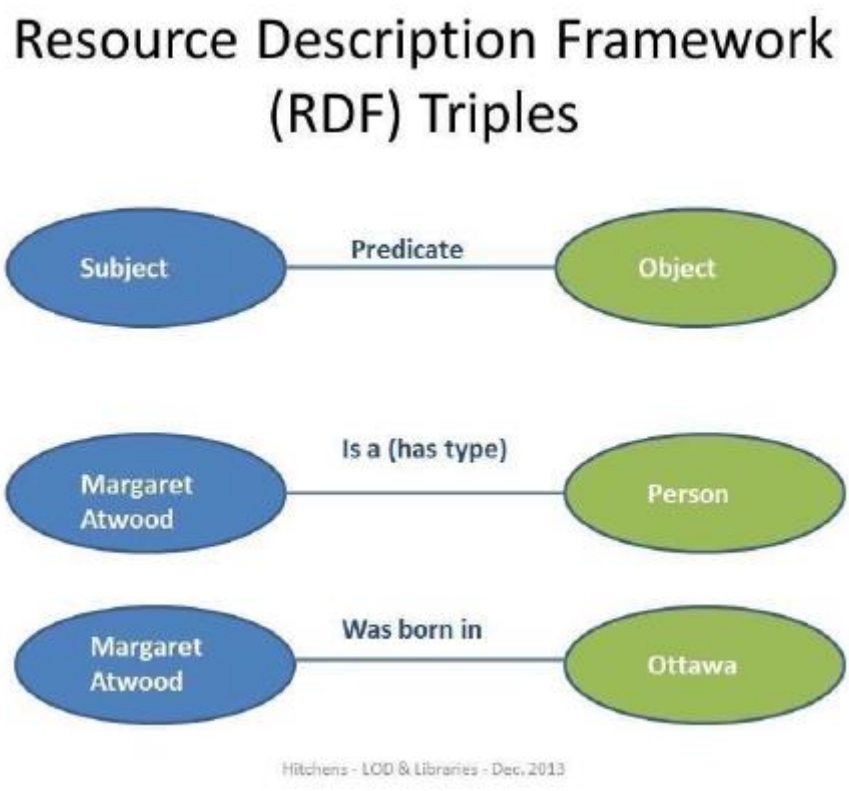


Figure 3 Σχηματική απεικόνιση του RDF

Όπως φαίνεται και από την εικόνα μπορούμε να καταλάβουμε πως το RDF λειτουργεί περίπου σαν το συντακτικό μίας φυσικής γλώσσας ή -όπως θα το έβλεπε ένας προγραμματιστής- πολύ κοντά στην λογική του αντικειμενοστραφούς προγραμματισμού. Τα στοιχεία κάθε τριπλέτας είναι (αναγνωρίζονται από) IRIs (International Resource Identifier), όπως είναι οι συμβολοσειρές, οι αριθμοί, κ.α. Τα IRIs είναι γενικά αναγνωριστικά, οπότε δύο εμφανίσεις ενός IRI δηλώνουν ουσιαστικά την ίδια έννοια. Ένα σύνολο από έννοιες που περιγράφονται από IRIs, το οποίο στοχεύει σε ένα συγκεκριμένο σενάριο ή ένα πλαίσιο εφαρμογών, ονομάζεται λεξιλόγιο ή πιο επίσημα οντολογία. Η W3C προτυποποίησε δύο λεξολόγια, το RDF Schema (RDFS) και το Web Ontology Language (OWL), για να περιγράφονται τα λεξιλόγια με έναν διαλειτουργικό τρόπο.

Το RDF Schema ορίζει τις έννοιες για να περιγράφονται οι κατηγορίες (classes), οι τύποι δεδομένων (data types) και οι ιδιότητες (properties) παρόμοια με το μοντέλο του αντικειμενοστραφούς προγραμματισμού. Ακόμα, ορίζει έννοιες για να περιγράφει σύνολα και λίστες. Αν και αυτό μπορεί να ακούγεται οικείο σε προγραμματιστές με εμπειρία στον αντικειμενοστραφή προγραμματισμό, οι λεπτομέρειες κάνουν τη διαφορά. Σε αντίθεση με τις γλώσσες προγραμματισμού, τα resources μπορούν να είναι συγχρόνως και στιγμιότυπα, αλλά και classes συγχρόνως. Οι classes, επίσης, δεν χρειάζεται να είναι απαραίτητα ξέχωρες

(disjoint) και βασικά στην RDFS δεν υπάρχει τρόπος αυτό να δηλωθεί ξεκάθαρα. Τέλος, τα properties μπορούν να εφαρμοστούν στα στιγμιότυπα οποιασδήποτε κλάσης.

Σε αντίθεση με το RDF Schema, η Web Ontology Language επιτρέπει να ορίσει κανείς την ένωση, την τομή ή τη μη ένωση και τομή δύο κλάσεων. Σε μια απλή ανάλυση, θα μπορούσε να περιγραφεί σαν μια επέκταση του RDFS, η οποία προσθέτει αρκετές νέες ιδέες και έννοιες κάνοντάς τη μια αρκετά εκφραστική περιγραφική γλώσσα (αν και από καθαρά τεχνική σκοπιά μόνο κάποια προφίλ της OWL μπορούν να θεωρηθούν άμεσες επεκτάσεις της RDFS). Μιας και οι δυο, η RDFS και η OWL, παίζουν έναν περιθωριακό ρόλο στην εργασία αυτή, δεν θα μπούμε σε περισσότερες τεχνικές λεπτομέρειες στις διαφορές αλλά ο ενδιαφερόμενος αναγνώστης μπορεί να επισκεφθεί τα αντίστοιχα specifications τους.

Παρότι το RDF είναι αρκετά εύληπτο (όπως αναφέραμε λειτουργεί αντίστοιχα με τον αντικειμενοστραφή προγραμματισμό και άρα γίνεται εύκολα κατανοητό από τους προγραμματιστές), ωστόσο στις αρχές του η μόνη σειριοποίησή του ήταν το RDF/XML, το οποίο κρίθηκε ακατάλληλο για χρήση λόγω της πολυπλοκότητας που προσέδιδε. Λόγω αυτού, το RDF απέκτησε την φήμης της περίπλοκης και δύσχρηστης τεχνολογίας και η υιοθέτησή του περιορίστηκε σημαντικά. Αργότερα όμως κάνοντας χρήση των URIs (σαν ονόματα αντικειμένων) ο σημασιολογικός ιστός κατάφερε να ενσωματωθεί με το web και να γίνει προσπελάσιμος (οι χρήστες πλέον πλοηγούνται κανονικά μέσω HTTP URIs), δίνοντας έτσι μία νέα οπτική στην χρήση και το μέλλον του. Πλέον ο σημασιολογικός ιστός και τα διασυνδεδεμένα δεδομένα, όπως ονομάζονται, έχει αυξηθεί κατά πολύ τα τελευταία χρόνια, όπως φαίνεται και στην εικόνα (Figure 4), και πολλές εταιρίες έχουν ασχοληθεί εμπορικά με αυτό τον τομέα, όπως π.χ. το Facebook. Αυτό δεν σημαίνει ωστόσο ότι ο σημασιολογικός ιστός έχει εξαπλωθεί σε όλο το web και ούτε αναμένεται να γίνει στο άμεσο μέλλον. Είναι μία τεχνολογία που ακόμα αναπτύσσεται και έχει δρόμο μέχρι να καλύψει όλα τα κενά και τα προβλήματα της.

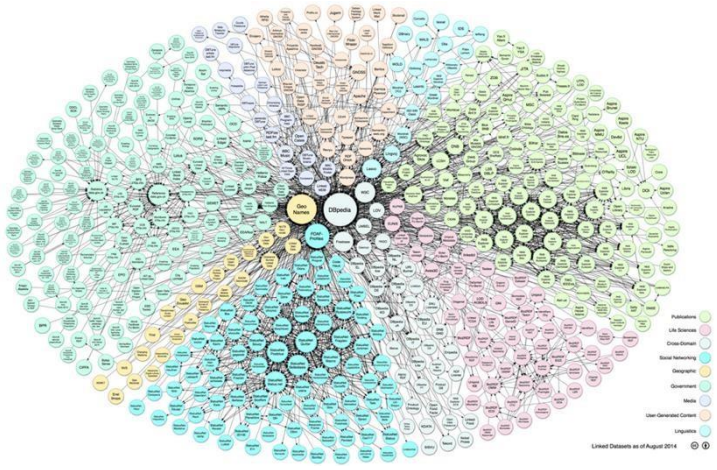


Figure 4 Γράφος διασυνδεδεμένων δεδομένων (2015)

3.4 Υπηρεσίες στο διαδίκτυο

Πολλές σελίδες στο Web έχουν σχεδιαστεί και υλοποιηθεί με το να παρουσιάζουν δεδομένα τα οποία βρίσκονται οργανωμένα μέσα σε βάσεις δεδομένων. Για να επεξεργαστεί κανείς αυτές τις σελίδες και τους «τόνους» από δεδομένα, αρκετές πρακτικές έχουν αναπτυχθεί, όπως το screen scraping, μια τεχνική με την οποία αυτόματα ένα πρόγραμμα επισκέπτεται όμοιες σελίδες και προσπαθεί να εξάγει δεδομένα με ίδιο τρόπο, ώστε να μπορέσει να ανακατασκευάσει τα αρχικά δεδομένα όπως ήταν στη βάση τους με σκοπό να γίνουν πάλι σε μορφή κατανοητή από μηχανές. Σκοπός του σημασιολογικού ιστού είναι να παρακάμψει αυτό τον περιορισμό και να δημιουργήσει ένα διαδίκτυο δεδομένων, το οποίο θα είναι άμεσα προσπελάσιμο από μηχανές. Στην πράξη, ακόμα και σήμερα ο συγκεκριμένος στόχος δεν επιτυγχάνεται επαρκώς με τις τεχνολογίες του σημασιολογικού ιστού. Αντιθέτως, η πλειονότητα των δομημένων δεδομένων δημοσιοποιείται με τη μορφή XML ή και JSON δεδομένων.

Γενικά μιλώντας, αυτού του τύπου η δημοσιοποίηση πληροφορίας ονομάζεται κοινώς Web services, αλλά δεδομένου ότι και αυτά που απευθύνονται σε ανθρώπους, όπως για παράδειγμα HTML σελίδες, που είναι τουλάχιστον ως έναν βαθμό αναγνώσιμες από μηχανή, ο προηγούμενος όρος εσφαλμένα και παραπλανητικά θα μπορούσε να ταξινομήσει ιστοσελίδες σαν υπηρεσίες. Παρόμοια λοιπόν, ένα Web service θα μπορούσε να θεωρηθεί μια ιστοσελίδα για μηχανές. Οπότε, στα πλαίσια της συγκεκριμένης διατριβής ορίζουμε τον όρο Web service σαν ένα σύνολο από HTTP βασιζόμενες διεπαφές με στόχο να υποστηρίζουν διαλειτουργικές μηχανή προς μηχανή διεργασίες με την ανταλλαγή σωστά δομημένων δεδομένων. Ο σκοπός των διεργασιών αυτών είναι να οδηγήσουν τις εμπορικές συναλλαγές, με στόχο να εκτελούν συγκεκριμένες εργασίες, στα πλαίσια πάντα ενός συγκεκριμένου πλαισίου εφαρμογής. Από τη στιγμή που τα Web services δίνουν έμφαση στις machine-to-machine (μηχανή προς μηχανή) συναλλαγές, αυτές είναι βελτιστοποιημένες για μηχανές αντί για ανθρώπους, όπως αντίστοιχα είναι οι ιστοσελίδες. Στην πράξη, υπάρχουν δυο μεγάλες κατηγορίες από Web services που μπορούν να διαχωριστούν, τα SOAP-based και τα RESTful. Θα τα συζητήσουμε και τα δυο στις αμέσως επόμενες ενότητες.

3.4.1 SOAP-based Υπηρεσίες

Σε μια προσπάθεια να βελτιωθεί η ευελιξία και η δυναμικότητα των προϊόντων τους, η βιομηχανία της πληροφορικής ξεκίνησε να δουλεύει στον επίσημο φορμαλισμό και την προτυποποίηση των Web υπηρεσιών στα τέλη της δεκαετίας του ενενήντα. Το αποτέλεσμα ήταν ένα αρκετά περίπλοκο πρότυπο βασισμένο στο XML, το οποίο κυρίως λόγω της

δημοτικότητας του την συγκεκριμένη περίοδο επιλέχθηκε τότε ας το βασικό φόρμα σειριοποίησης. Τα τρία βασικά αποτελέσματα αυτής της προσπάθειας είναι το SOAP, την WSDL και το UDDI.

Το SOAP[64], αρχικά ορίστηκε από την Microsoft ως Simple Object Access Protocol, ορίζει έναν τρόπο επικοινωνίας ο οποίος αποτελείται από ένα επεκτάσιμο μοντέλο, ένα πρωτόκολλο που να δένει αυτόν τον τρόπο (framework binding) και φυσικά την απεικόνιση αυτού σε XML. Η Web Service Description Language (WSDL[65]) περιγράφει μια διεπαφή για μια Web υπηρεσία και τα Universal Description Discovery and Integration (UDDI[66]) αποθετήρια επιτρέπουν την εύρεση υπηρεσιών, services και τις περιγραφές του.

Αν και έχουν γίνει μεγάλες επενδύσεις πάνω σε αυτές τις τεχνολογίες, η υπόσχεση για έναν ενιαίο τρόπο προσέγγισης στις περιγραφές των υπηρεσιών και η ενιαία προσέγγιση στην αποθήκευση και ανεύρεση μέσω των SOAP, WSDL και του UDDI τελικά αποδείχθηκε δύσκολη να επιτευχθεί. Το Universal Business Registry, το κύριο δημόσιο UDDI αποθετήριο, έκλεισε επίσημα το 2006 και τα περισσότερα δημόσια SOAP-based Web services τερματίστηκαν λίγο αργότερα. Υπήρχαν αρκετά θέματα που οδήγησαν σε αυτή την παύση, αλλά ο βασικότερος λόγος ήταν πως το θεμελιώδες συστατικό αυτής της αρχιτεκτονικής ήταν το Remore Procedure Call (RPC), το οποίο ήταν γνωστό πως είχε αδυναμίες και ελαττώματα από χρόνια[67]. Επιπροσθέτως, αντί να χρησιμοποιείται το HTTP σαν πρωτόκολλο εφαρμογών, όπως άλλωστε είναι σχεδιασμένο, εσφαλμένα χρησιμοποιείτο σαν απλό πρωτόκολλο μεταφοράς μηνυμάτων. Στο SOAP για παράδειγμα, τα δεδομένα συχνά επιστρέφονται με μια POST κλήση στην υπηρεσία, η οποία έπειτα επιστρέφει τα επιθυμητά αποτελέσματα. Αυτό φυσικά καταστρέφει τους ενδιάμεσους που εξυπηρετούν είτε σαν caches είτε σαν proxies, οι οποίοι βασίζονται στη σημασιολογία των ρημάτων του HTTP, και στα headers του μηνύματος που διέπουν. Στην πράξη, αυτό μειώνει δραματικά το scalability και σημαίνει ότι, το να τρέχει και να συντηρεί κάποιος μια δημόσια υπηρεσία ενέχει απαγορευτικά έξοδα. Κατά συνέπεια, ο αριθμός των δημοσίων SOAP υπηρεσιών είναι πλέον αισθητά μικρός. Κοιτάζοντας, βέβαια, σε υπηρεσίες εντός μια εταιρείας, εκεί το τοπίο αλλάζει αρκετά, αφού τα θετικά των εργαλείων που συνοδεύουν αυτές τις τεχνολογίες ξεπερνούν τα κόστη, οπότε, σε τέτοια σενάρια, αξίζει να χρησιμοποιηθούν.

Ένα ακόμα πρόβλημα, όταν δεν υπάρχει αυστηρή διαχείριση υπηρεσιών ανάμεσα σε αυτόν που παράγει την πληροφορία και σε αυτόν που την καταναλώνει, είναι πως αν και υπάρχει η αφαιρετικότητα στις δομές δεδομένων που βρίσκονται στις υλοποιήσεις, οι διεπαφές υπηρεσιών με βάση το WSDL συχνά δεν περιέχουν πληροφορίες σχετικά με την υλοποίηση, με αποτέλεσμα να δημιουργούνται αρκετά στενά συνδεδεμένα συστήματα, που δεν επιτρέπουν δηλαδή αλλαγές. Επίσης, η αντιστοίχηση αυτών στις αφαιρετικές δομές δεδομένων δεν είναι πάντα εύκολα εφικτή και έτσι συχνά υπάρχουν σοβαρά προβλήματα δια

λειτουργικότητας. Ειδικότερα, η κληρονομική δυσκολία αντιστοίχησης ανάμεσα στο XML Schemas (XSD[68]) και στον αντικειμενοστραφή προγραμματισμό, αυτό που συχνά αποκαλείται ως την O/X μη αντιστοίχιση, φέρνει ακόμα περισσότερες δυσκολίες διασύνδεσης συστημάτων. Η γλώσσα XML Schemas έχει ορισμένες ένα σύνολο από δομές δεδομένων που δεν βρίσκονται στην Java[69]. Κατά συνέπεια, προκύπτουν δυσκολίες στη διαλειτουργικότητα, αφού κάθε υποδομή λογισμικού του SOAP έχει τον δικό της τρόπο να υλοποιεί την αντιστοίχηση ανάμεσα στους διάφορους XSD τύπους και στις δομές του συστήματος ανά πλατφόρμα και γλώσσα προγραμματισμού και αντίστροφα.

Συνοψίζοντας, το πρόβλημα που περιεγράφηκε παραπάνω και η πολυπλοκότητα της τεχνολογίας, που αποτελείται από περισσότερα πρότυπα από το SOAP, την WSDL και το UDDI, οδήγησε στην αναζήτηση απλούστερων και πιο ελαφριών λύσεων, που θα μπορούσαν να συνδέσουν καλύτερα και να υποστηρίζουν την περίπλοκη κατανεμημένη φύση του Web.

3.4.2 Restful Υπηρεσίες

Σύμφωνα με στατιστικά από το ProgrammableWeb[70], το μεγαλύτερο ιστολόγιο στον χώρο των services που παρέχει έναν σχεδόν πλήρη κατάλογο, τρία από τα τέσσερα Web services βασίζονται στην αρχιτεκτονική του REST. Αυτό, προφανώς, δεν σημαίνει ότι ακολουθούν με πλήρη αυστηρότητα όλους τους περιορισμούς του REST, αλλά ότι χρησιμοποιούν το HTTP σαν το βασικό πρωτόκολλο επικοινωνίας και ότι κάθε resource έχει το δικό του μοναδικό IRI. Στην πραγματικότητα, τα περισσότερα services που ισχυρίζονται ότι είναι RESTful (REST APIs), δεν είναι. Για να καλύψει τα διαφορετικά επίπεδα του πόσο REST είναι ένα service, ο Richardson όρισε ένα μοντέλο ωριμότητας[71], αλλά εκ των πραγμάτων μια υπηρεσία είναι είτε RESTful, όταν ακολουθεί όλους τους κανονισμούς του REST, είτε δεν είναι. Φανερά ενοχλημένος από το γεγονός ότι οι κατασκευαστές τέτοιων υπηρεσιών παρέλειπαν συγκεκριμένα την hypermedia συνθήκη, έγραψε ο Fielding ένα άρθρο[72] όπου ανέλυε πως η συνθήκη αυτή δεν είναι προαιρετική. Μιας και ο όρος REST χρησιμοποιείται εσφαλμένα τόσο συχνά, πρόσφατα οι HTTP Web υπηρεσίες απλά αναφέρονται σαν Web APIs. Από αυτές, όσες σέβονται την συνθήκη για τα hypermedia, ονομάζονται hypermedia APIs.

Τα Web APIs έχουν σαν κοινό ότι χρησιμοποιούν ένα πολύ μικρό σύνολο από πρότυπα. Πιο συχνά, αποτελούνται από το ίδιο το HTTP και είτε από κάποιο από τα XML ή JSON, σαν την βασικήμέθοδο σειριοποίησης. Αν και το XML με την ονοματοδοσία που έχει εγγενώς που επιτρέπει στα μηνύματα να έχουν έλεγχο υπερσυνδέσεων(hypermedia controls) ή να είναι αυτό-περιγραφόμενα θα ήταν ιδανικό για τις REST υπηρεσίες, το JSON έχει γίνει το αγαπημένο μέσο ανταλλαγής δεδομένων στα Web APIs τα τελευταία χρόνια. Το αρνητικό σε αυτό είναι πως εξαιτίας της απλότητας του οι περισσότερες υπηρεσίες είναι μοναδικές και

περιγράφονται μόνο σε φυσική γλώσσα κατανοητή από τον άνθρωπο. Αυτό βέβαια καθιστά την δημιουργία εργαλείων που θα βγάζουν αυτόματα κώδικα ή γενικών εργαλείων για πολλές

4

Εισαγωγή στο DeepGraphs specification

Στις ενότητες που ακολουθούν γίνεται λόγος για το DeepGraphs specification που είναι θεμέλιο για την κατανόηση της εργασίας. Το DeepGraphs είναι ένα πρότυπο (σε πειραματικό ακόμα στάδιο) που στοχεύει στην εξυπηρέτηση της επικοινωνίας υπηρεσιών του διαδικτύου στα πλαίσια του σημασιολογικού ιστού. Βασιζόμενοι σε όλα τα προηγούμενα κεφάλαια αλλά και σε κάποιες επιπλέον τεχνολογίες που αναλύονται στο παρόν κεφάλαιο, όπως το Hydra specification και το Schema.org, θα κάνουμε μία συνοπτική αλλά εις βάθος μελέτη του προτύπου.

4.1 Hydra specification

Η Hydra είναι μία προσπάθεια που γίνεται για την απλοποίηση της ανάπτυξης των hypermedia-driven Web APIs. Τα δύο δομικά στοιχεία της Hydra είναι το JSON-LD και το Hydra Core Vocabulary.

Το JSON-LD είναι μία μορφή σειριοποίησης (serialization) που χρησιμοποιείται στην επικοινωνία μεταξύ του server και των πελατών του. Το Hydra Core Vocabulary αντιπροσωπεύει το κοινό λεξιλόγιο που μοιράζονται. Ο προσδιορισμός ορισμένων concepts που χρησιμοποιούνται συχνά στα web APIs, μπορεί να λειτουργήσει ως βάση για την δημιουργία υπηρεσιών web που διαθέτουν τα πλεονεκτήματα του REST. Δηλαδή χαλαροί δεσμοί, εύκολη συντήρηση, δυνατότητα εξέλιξης και ανάπτυξης. Επιπλέον επιτρέπει την

δημιουργία γενικών πελατών API σε αντίθεση με την ανάγκη για εξειδικευμένους πελάτες για κάθε απλό API.

Η Hydra είναι ένα project που βρίσκεται σε εξέλιξη και δεν έχει ολοκληρωθεί ακόμη. Για τον λόγο αυτό τα στοιχεία που παρουσιάζονται στη συνέχεια περιγράφουν την πιο πρόσφατη έκδοσή της και είναι πιθανό να αλλάξουν στο μέλλον.

4.1.1 Hydra Core Vocabulary

4.1.1.1 Εισαγωγή

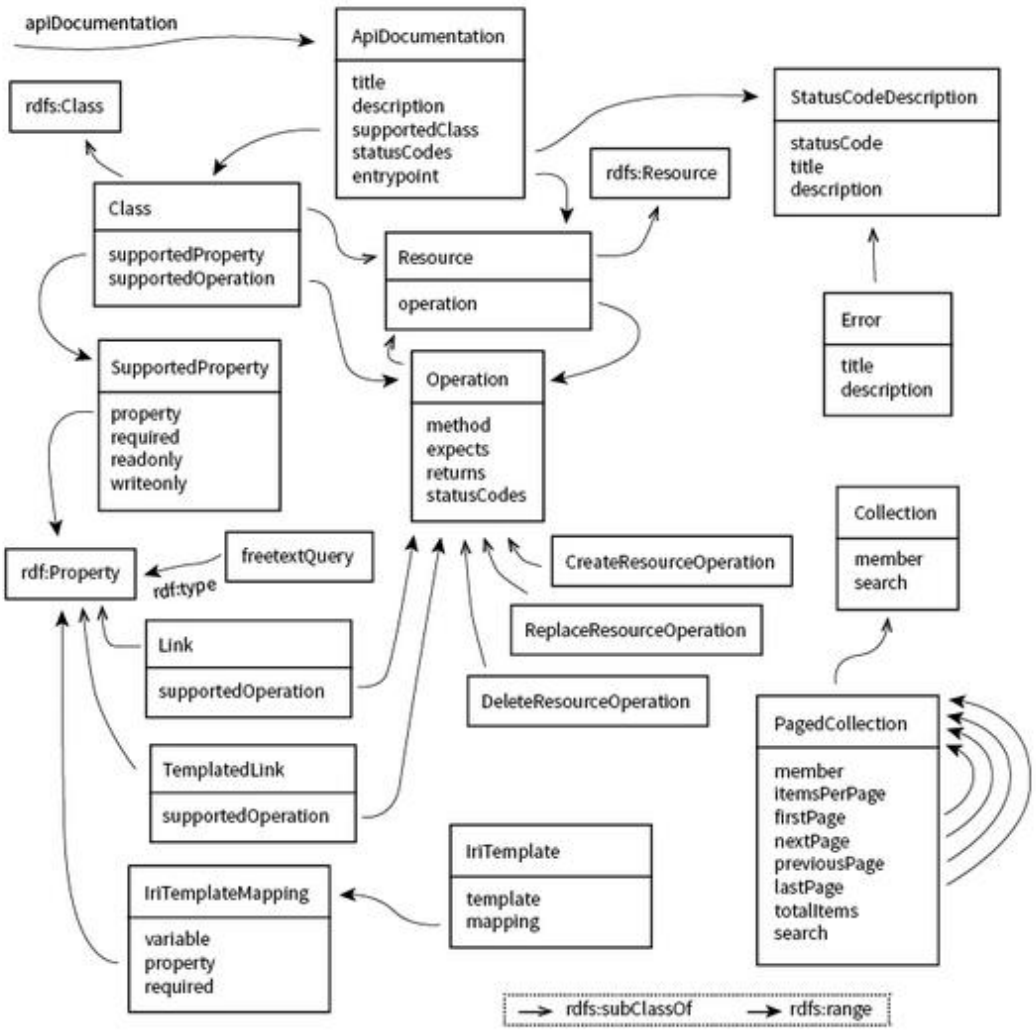
Το να τα βγάζει κανείς πέρα με την όλο και αυξανόμενη ποσότητα δεδομένων αποτελεί μία σημαντική πρόκληση. Για να μειωθεί ο φόρτος πληροφοριών που τίθεται στους ανθρώπους τα συστήματα αρχίζουν να συνδέονται άμεσα μεταξύ τους. Ανταλλάσσουν, αναλύουν και χρησιμοποιούν τεράστιες ποσότητες δεδομένων χωρίς καμία ανθρώπινη παρέμβαση. Οι λύσεις που υπάρχουν σήμερα όμως, δεν αξιοποιούν όλες τις δυνατότητες που προσφέρει το διαδίκτυο και αγνοούν την δυναμική των τεχνολογιών Linked Data.

Ο συνδυασμός του αρχιτεκτονικού στυλ REST και οι αρχές των Linked Data προσφέρουν ευκαιρίες για την εξέλιξη του διαδικτύου των μηχανών με τρόπο όμοιο με αυτόν που προσέφερε το hypertext στο διαδίκτυο των ανθρώπων. Τα περισσότερα στοιχεία για αυτό υπάρχουν ήδη και βρίσκονται σε εφαρμογή αλλά σπάνια χρησιμοποιούνται μαζί. Η Hydra προσπαθεί να γεφυρώσει αυτό το κενό. Επιτρέπει στα δεδομένα να εμπλουτιστούν με στοιχεία που μπορούν να διαβαστούν από μηχανές και επιτρέπουν την αλληλεπίδραση. Αυτό όχι μόνο αντιμετωπίζει το πρόβλημα των Linked Data που είναι σχεδόν αποκλειστικά για ανάγνωση αλλά ετοιμάζει το έδαφος για νέα web APIs. Το γεγονός ότι επιτρέπει την δημιουργία σύνθετων τμημάτων κώδικα σημαίνει ότι τα μοντέλα αλληλεπίδρασης των web APIs θα μπορούν να επαναχρησιμοποιηθούν σε μεγάλο βαθμό.

4.1.1.2 Μία γρήγορη ματιά στην Hydra

Η βασική ιδέα πίσω από την Hydra είναι η δημιουργία ενός λεξιλογίου που θα επιτρέπει σε έναν server να δείχνει μία έγκυρη αλλαγή κατάστασης σε έναν client. Ο client τότε μπορεί να χρησιμοποιήσει αυτή την πληροφορία ώστε να κάνει HTTP requests τα οποία τροποποιούν την κατάσταση του server ώστε να επιτευχθεί ένας συγκεκριμένος στόχος. Αφού όλες οι πληροφορίες σχετικά με τις έγκυρες αλλαγές καταστάσεων ανταλλάσσονται μεταξύ μηχανών κατά την διάρκεια της εκτέλεσής τους αντί να είναι καταγεγραμμένες στον κώδικα του client από τον σχεδιασμό του, οι clients μπορούν να αποσυζευχθούν από τον server και να προσαρμόζονται στις αλλαγές ευκολότερα.

Στο παρακάτω διάγραμμα παρουσιάζεται το βασικό λεξιλόγιο της Hydra. Σκοπός του είναι να δείξει πως χρησιμοποιείται η Hydra και όχι τον ακριβή ορισμό της.



The Hydra core vocabulary

4.1.1.3 Χρησιμοποιώντας την Hydra

Σε αυτή την ενότητα θα παρουσιαστεί η χρήση της Hydra μέσα από ένα παράδειγμα. Στο API του παραδείγματος οι χρήστες μπορούν να εισάγουν νέο θέμα, να επεξεργάζονται ή να διαγράφουν τα υπάρχοντα και να τα σχολιάζουν.

4.1.1.4 Προσθέτοντας δυνατότητες στις αναπαραστάσεις

Το API του παραδείγματος πρέπει να δείχνει αναπαραστάσεις από θέματα και σχόλια. Για να επιτρέψει την αλληλεπίδραση με αυτούς τους πόρους, ένας client πρέπει να ξέρει ποιες λειτουργίες υποστηρίζει ο server. Σε websites που προορίζονται για χρήση του ανθρώπου αυτές οι λειτουργίες συνήθως παρουσιάζονται ως links ή φόρμες και περιγράφονται σε

φυσική γλώσσα. Όμως οι πληροφορίες αυτές δεν μπορούν να αναγνωριστούν εύκολα από μηχανές. Η λύση για αυτό είναι να μειωθεί η γλώσσα σε ένα μικρό σύνολο από ξεκάθαρες ενότητες που μπορούν εύκολα να αναγνωριστούν από την μηχανή του client. Η Hydra προτυποποιεί τις ενότητες αυτές.

Ο πιο απλός και σημαντικός τρόπος για να προσφέρουμε δυνατότητες στο web είναι μέσω hyperlinks. Χωρίς αυτά θα ήταν αδύνατο να έχουμε πρόσβαση στο διαδίκτυο. Οι χρήστες επιλέγουν συνήθως το link ανάλογα με το κείμενο που το περιγράφει. Για να δώσουμε στις μηχανές μία αντίστοιχη ικανότητα κατανόησης, τα link μπορούν να σηματοδοτούνται με έναν τύπο link relation – ένα registered token ή ένα URI που προσδιορίζει την σημασιολογία του link. Στο ακόλουθο παράδειγμα βλέπουμε πως ένα link χρησιμοποιείται σε HTML για να δείξει σε ένα stylesheet

```
<link rel="stylesheet" href="http://www.example.com/styles.css" />
```

Στα Linked Data, ο τύπος link relation αντιστοιχεί στην ίδια την ιδιότητα. Γι αυτό ένα παράδειγμα σε JSON-LD θα γραφόταν

```
{
  "urn:iana:link-relations:stylesheet":
  { "@id": "http://www.example.com/styles.css" }
}
```

Γενικά, ένας client αποφασίζει εάν θα ακολουθήσει ένα link ή όχι με βάση τη σχέση του link που καθορίζει την σημασιολογία του. Υπάρχουν όμως clients όπως οι web crawlers που απλά ακολουθούν οποιοδήποτε link. Στην HTML αυτό σημαίνει ότι όλα τα links με την ετικέτα <a> ακολουθούνται.

Ενώ τα links είναι αρκετά για την δημιουργία APIs μόνο για ανάγνωση, για την δημιουργία web APIs που διαβάζονται και γράφονται απαιτούνται ισχυρότερα μέσα. Για τον λόγο αυτό η Hydra εισάγει την έννοια των λειτουργιών. Μία λειτουργία - Operation αντιπροσωπεύει τις πληροφορίες που είναι απαραίτητες για έναν client προκειμένου να δημιουργήσει ένα έγκυρο HTTP request ώστε να αλλάξει την κατάσταση του server. Επομένως η μόνη απαραίτητη ιδιότητα για μία λειτουργία είναι η HTTP μέθοδός της. Προαιρετικά είναι δυνατό να περιγράφει τις πληροφορίες που ο server αναμένεται να επιστρέψει συμπεριλαμβανομένων επιπλέον πληροφοριών σχετικά με τους κωδικούς κατάστασης HTTP που μπορεί να επιστραφούν. Αυτό βοηθά έναν προγραμματιστή στο να κατανοήσει τι να περιμένει όταν εκτελεί μία λειτουργία. Βέβαια αυτή η πληροφορία δεν θεωρείται πλήρης αλλά αποτελεί μία βοήθεια. Οι προγραμματιστές για παράδειγμα πρέπει να περιμένουν ότι μπορεί να επιστρέψουν και διαφορετικοί κωδικοί κατάστασης HTTP ώστε να προγραμματίσουν ανάλογα τους clients.

Η Hydra έχει τρείς κλάσεις προκαθορισμένων λειτουργιών με ονόματα *CreateResourceOperation*, *ReplaceResourceOperation*, και *DeleteResourceOperation*. Όπως λένε και τα ονόματά τους, μπορούν να χρησιμοποιηθούν για να εισάγουν απλή CRUD λειτουργικότητα. Περισσότερο εξειδικευμένες λειτουργίες μπορούν να δημιουργηθούν εύκολα με την τροποποίηση της βασικής κλάσης Operation.

Το ακόλουθο παράδειγμα δείχνει πώς αναπαραστάσεις μπορούν να επαυξηθούν με πληροφορίες που διευκολύνουν τους clients να αλληλεπιδράσουν μαζί τους. Ένας client θα μπορούσε να καταλάβει ότι το resource στο ακόλουθο παράδειγμα μπορεί να διαγραφεί στέλνοντας ένα HTTP DELETE request.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "/an-issue",
  "title": "An exemplary issue representation",
  "description": "This issue can be deleted with an HTTP DELETE request",
  "operation": [
    {
      "@type": "DeleteResourceOperation",
      "method": "DELETE"
    }
  ]
}
```

Στο παράδειγμα αυτό βλέπουμε ότι η Hydra συνδυάζει ιδιότητες όπως operation και method και τιμές όπως DeleteResourceOperation με URL που ξεκάθαρα αναφέρονται σε αυτές. Το ευρύτερο πλαίσιο είναι κοινό για πολλές αναπαραστάσεις σε web APIs και για τον λόγο αυτό έχει νόημα να μειώσουμε το μέγεθος της απάντησης σε ένα πλαίσιο που θα μπορεί εύκολα να φορτωθεί από τους clients.

4.1.1.5 Τεκμηρίωση ενός Web API

Στα web APIs οι περισσότερες αναπαραστάσεις μοιάζουν πολύ. Επιπλέον τα resources συχνά υποστηρίζουν τις ίδιες λειτουργίες. Για τον λόγο αυτό γίνεται μία προσπάθεια για την συγκέντρωση όλων αυτών των πληροφοριών σε ένα κεντρικό documentation. Παραδοσιακά, αυτό υπάρχει στην φυσική γλώσσα και οι προγραμματιστές ενσωματώνουν αυτή τη γνώση στους clients. Η Hydra πλησιάζει αυτό το ζήτημα προσπαθώντας να κάνει αυτό το κεντρικό documentation επεξεργάσιμο από μηχανές. Το γεγονός ότι όλοι οι ορισμοί μπορούν να αναγνωριστούν από URL δίνει την δυνατότητα επαναχρησιμοποίησης σε μεγάλο βαθμό.

Η κλάση της Hydra ApiDocumentation θέτει τις βάσεις για την περιγραφή ενός web API. Όπως φαίνεται στο ακόλουθο παράδειγμα, η Hydra περιγράφει ένα API δίνοντάς του έναν τίτλο, μία σύντομη περιγραφή και τεκμηριώνοντας το κύριο σημείο πρόσβασης σε αυτό. Επιπλέον, μπορούν να περιγραφούν οι κλάσεις που υποστηρίζονται από το web API και άλλες πληροφορίες σχετικά με τους κωδικούς κατάστασης που μπορούν να επιστραφούν. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν για την αυτόματη δημιουργία τεκμηριώσεων σε φυσική γλώσσα.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/doc/",
  "@type": "ApiDocumentation",
  "title": "The name of the API",
  "description": "A short description of the API",
  "entrypoint": "URL of the API's main entry point",
  "supportedClass": [
    ... Classes known to be supported by the Web API ...
  ],
  "statusCodes": [
    ... Additional information about HTTP status codes ...
  ]
}
```

Αφού η Hydra χρησιμοποιεί κλάσεις για να περιγράψει τις πληροφορίες που αναμένονται ή στέλνονται από μία λειτουργία, ορίζει και έναν τρόπο για να περιγράφει τις ιδιότητες που υποστηρίζονται από μία κλάση. Αυτό φαίνεται στο ακόλουθο παράδειγμα. Αντί να αναφερθούν οι ιδιότητες άμεσα, το supportedProperty αποτελεί μία ενδιάμεση δομή δεδομένων. Αυτό δίνει την δυνατότητα να ορίσουμε εάν μία συγκεκριμένη ιδιότητα απαιτείται, εάν είναι read-only ή write-only ανάλογα με την κλάση με την οποία έχει συσχετιστεί.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/doc/#Comment",
  "@type": "Class",
  "title": "The name of the class",
  "description": "A short description of the class.",
  "supportedProperty": [
    ... Properties known to be supported by the class ...
    {
      "@type": "SupportedProperty",
```

```
      "property": "#property", // The property
      "required": true, // Is the property required in a request to be valid?
      "readonly": false, // Can the property's value be modified or is it read-only?
      "writeonly": true // Can the property's value be retrieved or is it write-only?
    }
  ]
}
```

4.1.1.6 Εύρεση ενός Web API που βασίζεται στην Hydra

Το πρώτο βήμα όταν προσπαθούμε να αποκτήσουμε πρόσβαση σε ένα web API είναι να βρούμε ένα σημείο εισόδου. Συνήθως αυτό γίνεται κοιτάζοντας την τεκμηρίωση στην κεντρική σελίδα του εκδότη του API. Η Hydra επιτρέπει το κεντρικό σημείο εισόδου του API να εντοπίζεται αυτόματα εάν ο εκδότης του API το σηματοδοτήσει με ένα ειδικό HTTP Link Header όπως ορίζεται στο [RFC5988]. Ένας client της Hydra θα έψαχνε για ένα Link Header με relation type http://www.w3.org/ns/hydra/core#apiDocumentation .

Στο ακόλουθο παράδειγμα, ένας client της Hydra απλά μπαίνει στο homepage ενός εκδότη API (http://www.example.com) για να βρει το σημείο εισόδου του API. Ο client μπορεί να κάνει ένα HTTP GET ή HEAD request. Η διαφορά μεταξύ των δύο είναι ότι το πρώτο μπορεί να επιστρέψει ένα message-body ως απάντηση ενώ το δεύτερο όχι.

```
HEAD / HTTP/1.1
Host: www.example.com

==================================

HTTP/1.1 200 OK
...
Content-Type: application/ld+json
Link:                                    <http://api.example.com/doc/>; rel="http://www.w3.org/ns/hydra/core#apiDocumentation"
```

Η απάντηση στο προηγούμενο παράδειγμα περιέχει ένα HTTP Link Header που δείχνει στο http://api.example.com/doc/ . Ανακτώντας αυτό το resource ο client παίρνει ένα Hydra API documentation που ορίζει το κύριο σημείο εισόδου του API:

```
GET /doc/ HTTP/1.1
Host: api.example.com
Accept: application/ld+json

==================================

HTTP/1.1 200 OK
...
Content-Type: application/ld+json

{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/doc/",
  "title": "The example.com API",
  "entrypoint": "http://api.example.com/",
  ...
}
```

Βέβαια στις περισσότερες περιπτώσεις το σημείο εισόδου είναι ήδη γνωστό στον client. Για τον λόγο αυτό η εύρεση του API documentation χρησιμοποιώντας HTTP Link Headers συνήθως δεν είναι απαραίτητη.

4.1.1.7 Collections

Σε πολλές περιπτώσεις έχει νόημα να παρουσιάζονται μαζί resources που αναφέρονται σε ένα σύνολο από συσχετιζόμενα με κάποιο τρόπο resources. Τα αποτελέσματα μιας αναζήτησης ή ένα βιβλίο διευθύνσεων είναι δύο παραδείγματα. Για να απλοποιηθούν αυτές οι περιπτώσεις η Hydra ορίζει δύο κλάσεις: *hydra:Collection* και *hydra:PagedCollection*.

Η *hydra:Collection* χρησιμοποιείται για να αναφερθούμε σε ένα σύνολο από resources με τον εξής τρόπο:

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/an-issue/comments",
  "@type": "Collection",
  "member": [
    {
      "@id": "/comments/429"
    },
    {
      "@id": "/comments/781",
      "title": "Properties may be embedded directly in the collection"
    },
```

```
    ...
  ]
}
```

Όπως φαίνεται, τα στοιχεία-μέλη μπορούν να αποτελούνται είτε μόνο από ένα link ή να περιέχουν και κάποιες ιδιότητες. Σε μερικές περιπτώσεις το να εισάγονται ορισμένες ιδιότητες των μελών άμεσα στη συλλογή είναι ιδιαίτερα βοηθητικό καθώς έτσι μπορεί να μειωθεί ο αριθμός των HTTP requests που απαιτούνται ώστε να ληφθούν οι απαραίτητες πληροφορίες για την επεξεργασία του αποτελέσματος.

Επειδή οι συλλογές μπορεί να γίνουν πολύ μεγάλες, τα web APIs συχνά επιλέγουν να χωρίσουν μία συλλογή σε σελίδες. Στην Hydra αυτό μπορεί να υλοποιηθεί με το *hydra:PagedCollection.* Εκτός από την ιδιότητα member μία *PagedCollection* μπορεί να περιέχει links για την *firstPage*, *nextPage*, *previousPage*, ή *lastPage* όπως και πληροφορίες σχετικά με το *itemsPerPage* και το *totalItems* όπως παρουσιάζεται στο ακόλουθο παράδειγμα.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "http://api.example.com/an-issue/comments?page=3",
  "@type": "PagedCollection",
  "totalItems": "4980",
  "itemsPerPage": "10",
  "firstPage": "/an-issue/comments?page=1",
  "nextPage": "/an-issue/comments?page=4",
  "previousPage": "/an-issue/comments?page=2",
  "lastPage": "/an-issue/comments?page=498",
  "member": [
    ... the members of this PagedCollection ...
  ]
}
```

4.1.1.8 Templated Links

Μερικές φορές είναι αδύνατον για έναν server να παράξει ένα URL γιατί το URL αυτό εξαρτάται από πληροφορίες που είναι γνωστές μόνο από τον client. Μία τυπική περίπτωση είναι τα URL που επιτρέπουν στον client να κάνει αίτημα για αναζήτηση στον server. Σε μία τέτοια περίπτωση ο server δεν μπορεί να φτιάξει το URL επειδή δεν ξέρει το αίτημα για το οποίο ενδιαφέρεται ο client. Αυτό όμως που μπορεί να κάνει είναι να δώσει στον client ένα

template για να δημιουργήσει ένα τέτοιο URL. Στη Hydra η κλάση *IriTemplate* χρησιμοποιείται για αυτόν τον σκοπό.

Το *IriTemplate* αποτελείται από ένα *template*, του οποίου το συντακτικό αναλύεται στο [RFC6570], και από ένα σύνολο από *mappings*. Κάθε *IriTemplateMapping* ταιριάζει τη μεταβλητή (*variable)* που χρησιμοποιείται στην φόρμα με ένα *property* και μπορεί προαιρετικά να ορίζει εάν αυτή η μεταβλητή είναι απαραίτητη (*required*) ή όχι .

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@type": "IriTemplate",
  "template": "http://api.example.com/issues{?q}",
  "mapping": [
    {
      "@type": "IriTemplateMapping",
      "variable": "q",
      "property": "hydra:freetextQuery",
      "required": true
    }
  ]
}
```

Το παράδειγμα αυτό αντιστοιχίζει την μεταβλητή q στην ιδιότητα *freetextQuery* της Hydra και ορίζει ότι είναι απαραίτητη.

Όπως η κλάση *Link* της Hydra επιτρέπει τον ορισμό ιδιοτήτων που αντιπροσωπεύουν Hyperlinks (όπως περιγράφηκε στην παράγραφο Προσθέτοντας δυνατότητες στις αναπαραστάσεις), έτσι η κλάση *TemplatedLink* επιτρέπει τον ορισμό ιδιοτήτων που η τιμή τους είναι IRI templates. Η Hydra έχει μία τέτοια ιδιότητα που ονομάζεται *search* και χρησιμοποιείται για την περιγραφή διαθέσιμων διεπαφών αναζήτησης.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@id": "hydra:search",
  "@type": "hydra:TemplatedLink"
}
```

4.1.1.9 Περιγραφή των HTTP κωδικών κατάστασης και λαθών

Οι κωδικοί κατάστασης HTTP έχουν καλά ορισμένη σημασιολογία και μπορούν να χρησιμοποιηθούν για να δείξουν το αποτέλεσμα μίας λειτουργίας. Όμως αρκετές φορές οι κωδικοί αυτοί δεν είναι αρκετά εξειδικευμένοι κάνοντας δύσκολο το να καταλάβουμε την πραγματική αιτία ενός λάθους. Για παράδειγμα η απάντηση 429 Too Many Requests μας

ενημερώνει πολύ γενικά. Για την αντιμετώπιση αυτής της κατάστασης η Hydra ορίζει την κλάση StatusCodeDescription που επιτρέπει να σταλούν επιπλέον πληροφορίες με τον κωδικό κατάστασης HTTP.

```
{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@type": "StatusCodeDescription",
  "statusCode": 429,
  "title": "Too Many Requests",
  "description": "A maximum of 500 requests per hour and user is allowed.",
  ...
}
```

Ένας server μπορεί να στείλει κατευθείαν σε μία απάντησή του StatusCodeDescription. Όταν γίνεται αυτό καλό είναι να γράφεται σε κάποια υποκλάση ώστε η σημασία του να γίνεται πιο ξεκάθαρη. Η Hydra ορίζει μόνο μία τέτοια υποκλάση με το όνομα Error. Αυτό προσφέρει εκτεταμένο πλαίσιο για να φτάσουν λεπτομέρειες για το λάθος στον client.

```
HTTP/1.1 400 Bad Request
Content-Type: application/ld+json

{
  "@context": "http://www.w3.org/ns/hydra/context.jsonld",
  "@type": "Error",
  "title": "An error occurred",
  "description": "Typically, a specialization of this class is used in practice.",
  ...
}
```

4.1.2 JSON – LD

Το Linked Data είναι ένας τρόπος για να δημιουργήσουμε ένα δίκτυο από δεδομένα κατανοητά από μηχανές ανάμεσα σε διαφορετικά έγγραφα και web sites. Επιτρέπει σε μία εφαρμογή να ξεκινήσει από ένα σημείο των Linked Data και ακολουθώντας ενσωματωμένα links να φτάσει σε άλλα τμήματά τους που βρίσκονται σε διαφορετικές ιστοσελίδες του διαδικτύου.

Το JSON – LD είναι ένα “ελαφρύ” συντακτικό για την σειριοποίηση των Linked Data σε JSON. Ο σχεδιασμός του δίνει τη δυνατότητα σε υπάρχοντα JSON να ερμηνευθούν ως Linked Data με ελάχιστες αλλαγές. Κύριος σκοπός του JSON – LD είναι να αποτελέσει έναν τρόπο για την χρήση των Linked Data σε διαδικτυακά προγραμματιστικά περιβάλλοντα, για την δημιουργία υπηρεσιών web που θα συνεργάζονται μεταξύ τους και για την αποθήκευση Linked Data σε μηχανές αποθήκευσης που βασίζονται σε JSON. Επειδή το JSON-LD είναι 100% συμβατό με το JSON, ο μεγάλος αριθμός από εργαλεία και βιβλιοθήκες JSON που έχουν αναπτυχθεί ως σήμερα μπορούν να επαναχρησιμοποιηθούν. Εκτός από όλα τα χαρακτηριστικά που προσφέρει το JSON, το JSON-LD εισάγει:

- Έναν μηχανισμό αναγνώρισης για JSON objects μέσω της χρήσης IRIs
- Έναν μηχανισμό στον οποίο η τιμή ενός JSON object μπορεί να αναφέρεται σε ένα JSON object ενός άλλου web site.
- Έναν τρόπο να σχετίζονται τύποι δεδομένων με τιμές όπως ημερομηνίες και χρόνοι
- Την δυνατότητα να εκφραστεί ένας ή περισσότεροι γράφοι, όπως ένα κοινωνικό δίκτυο, σε ένα έγγραφο.

Το JSON-LD σχεδιάστηκε ώστε να χρησιμοποιείται κατευθείαν ως JSON χωρίς γνώση για RDF. Επίσης σχεδιάστηκε ώστε να χρησιμοποιείται αν το επιθυμεί κανείς ως RDF σε συνδυασμό με άλλες τεχνολογίες για Linked Data όπως την SPARQL. Το συντακτικό του δεν προκαλεί προβλήματα σε συστήματα που τρέχουν ήδη JSON αλλά παρέχει μία αναβάθμιση σε αυτά.

Το JSON-LD ικανοποιεί τους ακόλουθους σχεδιαστικούς στόχους:

- Απλότητα: Για να χρησιμοποιηθεί το JSON-LD στην βασική του μορφή δεν απαιτούνται επιπλέον επεξεργαστές ή βιβλιοθήκες λογισμικού. Η γλώσσα διευκολύνει τους προγραμματιστές κατά την εκμάθησή της καθώς το μόνο που χρειάζεται να ξέρουν είναι JSON και δύο λέξεις κλειδιά, @context και @id, για να εφαρμόσουν τα βασικά της χαρακτηριστικά.
- Συμβατότητα: Ένα έγγραφο JSON-LD είναι πάντα ένα έγκυρο έγγραφο JSON. Αυτό διαβεβαιώνει ότι όλες οι βιβλιοθήκες του JSON λειτουργούν εξίσου και για τα έγγραφα JSON-LD.
- Εκφραστικότητα: Το συντακτικό σειριοποιεί τους κατευθυνόμενους γράφους. Αυτό διαβεβαιώνει ότι σχεδόν οποιοδήποτε μοντέλο από τον πραγματικό κόσμο μπορεί να εκφραστεί.
- Λακωνικότητα: Το συντακτικό του JSON-LD είναι πολύ λακωνικό και ευανάγνωστο από τον άνθρωπο, απαιτώντας την ελάχιστη δυνατή προσπάθεια από τον προγραμματιστή.
- Ελάχιστες τροποποιήσεις: Το JSON-LD κάνει μία ομαλή μετάβαση από τα υπάρχοντα συστήματα σε JSON. Στις περισσότερες περιπτώσεις το μόνο που χρειάζεται είναι η προσθήκη μιας γραμμής στο HTTP response. Αυτό επιτρέπει στους οργανισμούς που έχουν μεγάλη υποδομή σε JSON να χρησιμοποιήσουν τα χαρακτηριστικά του JSON-LD με έναν τρόπο που δεν θα επηρεάσει την καθημερινή τους λειτουργία και είναι οικείος στους υπάρχοντες πελάτες. Βέβαια υπάρχουν

φορές που η αντιστοίχιση ενός JSON σε μορφή γράφου είναι μία σύνθετη διαδικασία. Ενώ οι μηδενικές μετατροπές είναι στόχος του JSON-LD, μερικές φορές αυτό δεν είναι δυνατό χωρίς η γλώσσα να γίνει σύνθετη. Το JSON-LD επικεντρώνεται στην απλότητα όπου αυτό είναι δυνατό.

- Χρήση ως RDF: Το JSON-LD μπορεί να χρησιμοποιηθεί από τους προγραμματιστές σαν ιδιωματική JSON, χωρίς την ανάγκη κατανόησης RDF. Επιπλέον το JSON-LD μπορεί να χρησιμοποιηθεί ως RDF, έτσι οι άνθρωποι που πρόκειται να χρησιμοποιήσουν JSON-LD μαζί με άλλα εργαλεία για RDF θα ανακαλύψουν ότι μπορεί να χρησιμοποιηθεί ως οποιοδήποτε άλλο συντακτικό RDF.

4.1.2.1 Μοντέλο δεδομένων

Γενικά, το μοντέλο δεδομένων που χρησιμοποιείται από το JSON-LD είναι ένας labeled, κατευθυνόμενος γράφος. Ο γράφος περιέχει κόμβους που συνδέονται με ακμές. Ένας κόμβος είναι συνήθως δεδομένα όπως string, αριθμός, τυποποιημένες τιμές όπως ημερομηνίες και χρόνοι ή IRI. Επίσης υπάρχει μία ειδική κλάση κόμβου που ονομάζεται blank node (κενός κόμβος) και συνήθως χρησιμοποιείται για να εκφράσει δεδομένα που δεν έχουν ένα καθολικό αναγνωριστικό όπως ένα IRI. Οι blank nodes ταυτοποιούνται χρησιμοποιώντας ένα αναγνωριστικό κενού κόμβου. Αυτό το απλό μοντέλο δεδομένων είναι απίστευτα προσαρμόσιμο και ισχυρό, με δυνατότητα να μοντελοποιήσει σχεδόν όλα τα είδη δεδομένων.

Δύο χαρακτηριστικές λέξεις κλειδιά για ένα έγγραφο JSON-LD είναι οι @context και @id.

@context

Χρησιμοποιείται για να ορίσει τα σύντομα ονόματα που χρησιμοποιούνται σε ένα JSON-LD έγγραφο. Τα ονόματα αυτά ονομάζονται terms και βοηθούν τους προγραμματιστές να εκφράσουν ορισμένα αναγνωριστικά συνοπτικά.

@id

Χρησιμοποιείται για να ορίσει μοναδικά τα *things* που περιγράφονται στο έγγραφο με IRIs ή με αναγνωριστικά κενού κόμβου.

4.1.2.2 Βασικά χαρακτηριστικά

Το JSON είναι ένα “ελαφρύ”, ανεξάρτητο από γλώσσα format για την ανταλλαγή δεδομένων. Είναι εύκολο να αναλυθεί και να δημιουργηθεί. Βέβαια είναι δύσκολο να ενσωματώσεις JSON από διαφορετικές πηγές καθώς τα δεδομένα μπορεί να περιέχουν κλειδιά που συγκρούονται με άλλες πηγές δεδομένων. Επιπλέον το JSON δεν έχει ενσωματωμένη υποστήριξη για hyperlinks που είναι βασικό συστατικό του διαδικτύου. Ακολουθεί ένα παράδειγμα από απλό JSON

```
{
  "name": "Manu Sporny",
```

```
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Για τους ανθρώπους είναι προφανές ότι πρόκειται για έναν άνθρωπο που το όνομά του (name) είναι “Manu Sporny” και ότι η ιδιότητα homepage περιέχει το URL αυτού του ανθρώπου. Μία μηχανή δεν μπορεί να καταλάβει κάτι τέτοιο και μερικές φορές είναι δύσκολο ακόμη και για τον άνθρωπο. Αυτό το πρόβλημα μπορεί να λυθεί χρησιμοποιώντας ξεκάθαρα αναγνωριστικά για να δείξουμε το διαφορετικό νόημα και όχι απλά σύμβολα όπως “name” και ”homepage”.

Τα Linked Data, και το διαδίκτυο γενικότερα, χρησιμοποιούν IRIs (Internationalized Resource Identifiers) για ξεκάθαρη ταυτοποίηση. Η ιδέα είναι να χρησιμοποιούμε IRIs για να ορίζουμε ξεκάθαρα αναγνωριστικά σε δεδομένα που μπορεί να χρησιμοποιηθούν και από άλλους προγραμματιστές. Είναι χρήσιμο για όρους όπως name και homepage να δημιουργηθούν IRIs ώστε οι προγραμματιστές να χρησιμοποιούν κοινή ορολογία και μαζί με τις μηχανές να βλέπουν τη σημασία των όρων που αναζητούν. Η διαδικασία αυτή είναι γνωστή ως IRI dereferencing.

Αξιοποιώντας το δημοφιλές λεξιλόγιο schema.org, το προηγούμενο παράδειγμα θα μπορούσε να γραφεί ως:

```
{
  "http://schema.org/name": "Manu Sporny",
  "http://schema.org/url": { "@id": "http://manu.sporny.org/" },    ← The
'@id' keyword means 'This value is an identifier that is an IRI'
  "http://schema.org/image":                    {                    "@id":
"http://manu.sporny.org/images/manu.png" }
}
```

Εδώ βλέπουμε ότι κάθε ιδιότητα είναι ξεκάθαρα ορισμένη από ένα IRI και ότι όλες οι τιμές που αντιπροσωπεύουν IRIs σημειώνονται από την λέξη κλειδί @id. Ενώ το έγγραφο αυτό είναι ένα έγκυρο έγγραφο JSON-LD που περιγράφει τα δεδομένα του με μεγάλη ακρίβεια είναι ταυτόχρονα υπερβολικά μακροσκελές και δύσκολο για να δουλέψουν πάνω του άνθρωποι. Για να λυθεί αυτό το θέμα το JSON-LD εισάγει την έννοια του context.

4.1.2.3 Context

Όταν δύο άνθρωποι μιλάνε μεταξύ τους, η συζήτηση λαμβάνει χώρα σε ένα κοινό περιβάλλον που ονομάζεται το context της συζήτησης. Αυτό το κοινό context επιτρέπει στους συμμετέχοντες να χρησιμοποιούν συντομεύσεις όπως το μικρό όνομα ενός κοινού φίλου ώστε να επικοινωνούν γρηγορότερα χωρίς όμως να χάνουν σε ακρίβεια. Το context

στο JSON-LD λειτουργεί με τον ίδιο τρόπο. Επιτρέπει σε δύο εφαρμογές να επικοινωνούν μεταξύ τους πιο αποδοτικά χωρίς έλλειψη ακρίβειας. Στην ουσία το context χρησιμοποιείται ώστε να αντιστοιχίσει τους όρους στα IRIs τους. Για το προηγούμενο παράδειγμα το context θα μπορούσε να είναι το:

```
{
  "@context":
  {
    "name":  "http://schema.org/name",     ← This means that 'name' is shorthand for 'http://schema.org/name'
    "image": {
      "@id":  "http://schema.org/image",    ← This means that 'image' is shorthand for 'http://schema.org/image'
      "@type": "@id"   ← This means that a string value associated with 'image' should be interpreted as an identifier that is an IRI
    },
    "homepage": {
      "@id":  "http://schema.org/url",    ← This means that 'homepage' is shorthand for 'http://schema.org/url'
      "@type": "@id"   ← This means that a string value associated with 'homepage' should be interpreted as an identifier that is an IRI
    }
  }
}
```

Όπως παρατηρούμε η τιμή του ορισμού ενός όρου μπορεί να είναι είτε ένα απλό string που ένωνε τον όρο με το IRI είτε ένα JSON object.

Το context μπορεί είτε να γραφτεί άμεσα μέσα στο έγγραφο είτε να κληθεί. Αν υποθέσουμε ότι το context του προηγούμενου παραδείγματος βρίσκεται στο http://json-ld.org/contexts/person.jsonld μπορεί να κληθεί προσθέτοντας στο JSON-LD έγγραφο μία μόνο σειρά όπως φαίνεται στο παράδειγμα:

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Έγγραφα JSON μπορούν να ερμηνευθούν ως JSON-LD χωρίς να τροποποιηθούν απλώς προθέτοντας ένα context σαν HTTP Link Header.

Στα έγγραφα JSON-LD το context μπορεί να οριστεί και μέσα στον κώδικα. Αυτό έχει το πλεονέκτημα ότι το έγγραφο μπορεί να επεξεργασθεί ακόμα και αν δεν υπάρχει σύνδεση με το δίκτυο.

```
{
  "@context":
  {
    "name": "http://schema.org/name",
    "image": {
      "@id": "http://schema.org/image",
      "@type": "@id"
    },
    "homepage": {
      "@id": "http://schema.org/url",
      "@type": "@id"
    }
  },
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

4.1.2.4 IRIs

Τα IRIs είναι δομικά στοιχεία των Linked Data καθώς αυτά ορίζουν τι είναι κάθε κόμβος και κάθε ιδιότητα. Στο JSON-LD τα IRIs μπορούν να αναπαρασταθούν ως απόλυτα ή σχετικά IRIs. Ένα απόλυτο IRI ορίζεται να περιέχει ένα scheme μαζί με ένα path και προαιρετικά πεδία query και fragment. Ένα σχετικό IRI είναι ένα IRI που είναι σχετικό ως προς κάποιο απόλυτο IRI. Στο JSON-LD όλα τα σχετικά IRIs είναι σχετικά ως προς το base IRI.
Ένα string ερμηνεύεται ως IRI όταν είναι τιμή του @id:

```
{
...
  "homepage": { "@id": "http://example.com/" }
...
}
```

Τιμές που ερμηνεύονται ως IRIs μπορούν να εκφραστούν και σαν σχετικά IRIs. Για παράδειγμα αν υποθέσουμε ότι το ακόλουθο έγγραφο βρίσκεται στο http://example.com/about/ , το σχετικό IRI ../ θα πήγαινε στο http://example.com/ .

```
{
...
  "homepage": { "@id": "../" }
```

```
...
}
```

Απόλυτα IRIs μπορούν να εκφραστούν άμεσα στην θέση του κλειδιού όπως:

```
{
...
  "http://schema.org/name": "Manu Sporny",
...
}
```

Στο προηγούμενο παράδειγμα το κλειδί `http://schema.org/name` ερμηνεύεται ως απόλυτο IRI.

4.1.2.5 Αναγνωριστικά κόμβων

Για να είναι δυνατό να αναφερθούμε εξωτερικά στους κόμβους ενός γράφου είναι σημαντικό ο καθένας να έχει το αναγνωριστικό του. Τα IRIs είναι βασικό στοιχείο των Linked Data ώστε οι κόμβοι να είναι πραγματικά συνδεδεμένοι. Η κλήση ενός αναγνωριστικού θα πρέπει να καταλήγει στην αναπαράσταση του κόμβου του. Αυτό μπορεί να επιτρέπει σε μία εφαρμογή να ανακτήσει περισσότερες πληροφορίες για έναν κόμβο.
Στο JSON-LD, ένας κόμβος ταυτοποιείται χρησιμοποιώντας την λέξη κλειδί @id:

```
{
  "@context":
  {
    ...
    "name": "http://schema.org/name"
  },
  "@id": "http://me.markus-lanthaler.com/",
  "name": "Markus Lanthaler",
  ...
}
```

Το παράδειγμα αυτό περιέχει έναν κόμβο που χαρακτηρίζεται από το IRI `http://me.markus-lanthaler.com/`.

4.1.2.6 Ορίζοντας τον Τύπο

Ο τύπος ενός συγκεκριμένου κόμβου μπορεί να οριστεί χρησιμοποιώντας την λέξη κλειδί @type. Στα Linked Data οι τύποι ορίζονται μοναδικά με ένα IRI.

```
{
...
  "@id": "http://example.org/places#BrewEats",
  "@type": "http://schema.org/Restaurant",
...
}
```

Ένας κόμβος μπορεί να έχει πάνω από έναν τύπο και για να οριστεί κάτι τέτοιο χρησιμοποιούμε πίνακα

```
{
...
  "@id": "http://example.org/places#BrewEats",
  "@type": [ "http://schema.org/Restaurant", "http://schema.org/Brewery" ],
...
}
```

Η τιμή ενός κλειδιού @type μπορεί επίσης να είναι ένας όρος που ορίζεται στο context:

```
{
  "@context": {
    ...
    "Restaurant": "http://schema.org/Restaurant",
    "Brewery": "http://schema.org/Brewery"
  }
  "@id": "http://example.org/places#BrewEats",
  "@type": [ "Restaurant", "Brewery" ],
  ...
}
```

4.2 SWRL

Η SWRL (Semantic Web Rule Language) δημιουργήθηκε από τον συνδυασμό των γλωσσικών υποκατηγοριών της OWL (OWL DL και Lite) και της RuleML (Unary/Binary Datalog).

Η γλώσσα κατατέθηκε στο W3C (World Wide Web Consortium), το οποίο αποτελεί την Κοινοπραξία του Παγκοσμίου Ιστού και είναι η κύρια οργάνωση διεθνών προτύπων για το Παγκόσμιο Ιστό, τον Μάιο του 2004 και αποτέλεσε μία συνεργασία του Εθνικού Ερευνητικό Συμβουλίου του Καναδά, του Πανεπιστήμιο του Στανφορντ καθώς και άλλων επιτροπών .

Σε σύγκριση με προγράμματα περιγραφικής λογικής (Description Logic Programs, DLP), που αποτελούν μια αρχική πρόταση για υλοποίηση περιγραφών και κανόνων του Horn από μια ομάδα συγγραφέων, η SWRL ακολουθεί μια αντίθετη λογική υλοποίησης. Ενώ τα DLP μπορούν να οριστούν ως η διασταύρωση μεταξύ της περιγραφικής λογικής και των κανόνων του Horn, η SWRL θεωρείται ουσιαστικά ένας τρόπος συνένωσης τους. Το αποτέλεσμα που προκύπτει μέσω του DLP είναι μία γλώσσα η οποία δεν είναι πολύ χρηστική καθώς η μορφή των εξαγόμενων της είναι σχετικά περίπλοκη, χωρίς να μπορεί κανείς να διακρίνει εύκολα τους περιορισμούς της που προέρχονται από τον μετασχηματισμό Lloyd-Topor. Από την άλλη μεριά έχει τη δυναμικότητα της OWL DL, όμως έχει το μειονέκτημα της προσθήκης κανόνων που πολλές φορές δεν οδηγούν σε κάποιο αποτέλεσμα ή έχουν ελλιπή υλοποίηση. Παρόλα αυτά η πλατφόρμα υλοποίησης της SWRL (SWRL Tab of Protégé) έχει γίνει αρκετά δημοφιλής.

Οι κανόνες που δημιουργούνται με τη βοήθεια της SWRL έχουν μια μορφή σύνδεσης μεταξύ μιας προϋπάρχουσας συνθήκης(κυρίως σώμα- body) και ενός αποτελέσματος-συνέπειας(κεφαλή- head), η λογική έκφραση που ορίζει έναν κανόνα μπορεί να υπάρχει και στις δύο πλευρές. Ακολουθείται δηλαδή η κλασσική λογική πρώτης τάξεως, η οποία εκφράζεται ως: όταν ισχύουν οι συνθήκες που ορίζονται από το κυρίως σώμα τότε πρέπει να ισχύουν και οι συνθήκες που προσδιορίζονται από την κεφαλή.

4.3 Schema.org

Στο κεφάλαιο 3.3.1 ήδη κάναμε λόγω για τις τεχνολογίες του σημασιολογικού ιστού, το RDF και κάναμε αναφορά σε προβλήματα που ακόμα εκκρεμούν. Ένα από αυτά τα προβλήματα είναι η αμφισημία που καμιά φορά προκύπτει από την χρήση των IRIs ως αναγνωριστικών για τα διάφορα αντικείμενα. Ένα χαρακτηριστικό παράδειγμα είναι η αδυναμία να προσδιοριστεί αν ένα IRI αναφέρεται στην αναπαράσταση ενός αντικειμένου ή στην αφηρημένη έννοια. Διάφορες προσεγγίσεις έχουν προταθεί για να λυθεί αυτό το πρόβλημα, όπως για παράδειγμα η χρήση συμφραζομένων (context) με σκοπό να αποφασιστεί σε ποιό από όλα τα πιθανά διαφορετικά πράγματα αναφέρεται το IRI.

Οι προγραμματιστές του web επομένως, όχι μόνο πρέπει να προσδιορίζουν τα αντικείμενα και ιδέες μέσω ενός IRI, αλλά και να διαχωρίζουν τις οντότητες που περιέχουν πληροφορία, όπως π.χ. τα κείμενα, και αυτές που δεν έχουν, όπως π.χ. τα άτομα, με σκοπό να διαλέγουν κάθε φορά τη σωστή μορφή του IRI (ή να στέλνουν σε μία άλλη διεύθυνση). Σε αυτή την κατεύθυνση έχουν γίνει μεγάλες εμπορικές προσπάθειες που προσπαθούν να λύσουν αυτό το πρόβλημα υπολογιστικά, με σημαντικότερη ίσως το Schema.org.

Το Schema.org αναπαριστά ένα λεξιλόγιο για ένα ευρύ φάσμα εφαρμογών που επεκτείνεται από εκδηλώσεις και συνταγές έως προϊόντα και ανθρώπους. Το Schema.org είναι μία κοινή προσπάθεια ανάμεσα στην Google, τη Microsoft, τη Yahoo! και τη Yanex. Όλες οι προαναφερθείσες εταιρίες πρόσθεσαν στις μηχανές αναζήτησής τους το schme.org με σκοπό να εξάγουν δομημένη πληροφορία από ιστοσελίδες ώστε να βελτιστοποιούν τα αποτελέσματά τους και να τα αναπαριστούν με καλύτερο τρόπο. Οι προγραμματιστές από την άλλη μεριά επωφελούνται κι αυτοί με τη σειρά τους, με το να εμφανίζονται καλύτερα οι σελίδες τους και να έχουν μεγαλύτερη επισκεψιμότητα.

Ας πάμε όμως ένα βήμα πίσω να καταλάβουμε ακριβώς τι είναι το Schema.org, γιατί είναι σημαντικό και πως σχετίζεται με τον σημασιολογικό ιστό. Όπως ήδη αναφέρθηκε το Schme.org είναι ένα λεξιλόγιο που σκοπό έχει να δημιουργήσει και να υποστηρίξει ένα κοινό σύνολο από «σχήματα» (schemas) για δομημένη σήμανση δεδομένων στο web. Τι σημαίνει όμως μεταδεδομένα και πως τα εντάσσω σε μία ιστοσελίδα; Ας δούμε ένα πολύ απλό παράδειγμα. Οι περισσότεροι προγραμματιστές που ασχολούνται με την ανάπτυξη εφαρμογών στο διαδίκτυο είναι εξοικειωμένοι με την έννοια των HTML tags. Έστω λοιπόν ότι έχουμε την γραμμή <h1> Avatar </h1> μέσα στον HTML κώδικά μας. Αυτό σημαίνει ότι στην επικεφαλίδα 1 (header 1) εμφανίζεται η συμβολοσειρά “Avatar” και κατ’ επέκταση ο browser θα δει και θα μεταφράσει σωστό την γραμμή αυτή του κώδικα, δείχνοντας τη λέξη Avatar στην μορφή επικεφαλίδας. Παρότι το κείμενο εμφανίστηκε σωστά και ο χρήστης έλαβε το μήνυμα, ωστόσο τόσο ο browser όσο και η εκάστοτε μηχανή αναζήτησης δεν έχουν ιδέα σε τι αναφέρεται το περιεχόμενο της επικεφαλίδας. Από όσο γνωρίζουν θα μπορούσε να αναφέρεται σε κάποια φωτογραφία προφίλ ενός χρήστη κοινωνικού δικτύου ή στην επιτυχημένη 3D ταινία. Επομένως αν κάποιος χρήστης στην άλλη μεριά του internet ζητήσει πληροφορίες για την ταινία Avatar η μηχανή αναζήτησης δεν ξέρει αν η συγκεκριμένη σελίδα είναι σχετική ή όχι. Το Schema.org λοιπόν παρέχει έναν λεξιλόγιο στους προγραμματιστές να επισημάνουν τις σελίδες τους με έναν τρόπο που είναι κατανοητός από τις μηχανές αναζήτησης.

Αν έπρεπε να δώσουμε έναν ορισμό για τα μεταδεδομένα θα λέγαμε ότι είναι τα δεδομένα που παρέχουν πληροφορία για μία ή περισσότερες πτυχές των δεδομένων, κάνοντας έτσι τα δεδομένα πιο εύκολα εντοπίσιμα και εύκολα στην επεξεργασία. Η πιο δημοφιλής πρόταση

είναι η χρήση του schema.org με Microdata, RDFa ή JSON-LD formats για την σήμανση του περιεχομένου ιστοσελίδων ώστε να περιλαμβάνουν πλέον και μεταδεδομένα (metadata) σχετικά με τον εαυτό τους. Έστω λοιπόν ότι έχουμε τον παρακάτω κώδικα:

```
<div>
 <h1>Avatar</h1>
 <span>Director: James Cameron (born August 16, 1954)</span>
 <span>Science fiction</span>
 <a href="../movies/avatar-theatrical-
trailer.html">Trailer</a>
</div>
```

Αν κάναμε χρήση των Microdata tags (εισήχθησαν στην HTML5) θα είχαμε το ακόλουθο αποτέλεσμα:

```
<div itemscope itemtype="http://schema.org/Movie">
  <h1>Avatar</h1>
  <span>Director: James Cameron (born August 16, 1954)</span>
  <span>Science fiction</span>
  <a href="../movies/avatar-theatrical-
trailer.html">Trailer</a>
</div>
```

Με τον παραπάνω τρόπο απλά δηλώσαμε ότι αυτό που περιγράφεται στο ακόλουθο div είναι μία ταινία, όπως περιγράφεται στο schema.org, και άρα πλέον οι μηχανές αναζήτησης ξέρουν ότι η σελίδα αυτή αφορά την ταινία.

Επιπλέον προσθήκες θα μπορούσαν να γίνουν στον παραπάνω κώδικα βάζοντας κι άλλα μεταδεδομένα στην σελίδα, όπως:

```
<div itemscope itemtype ="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director" itemscope
itemtype="http://schema.org/Person">
  Director: <span itemprop="name">James Cameron</span> (born
<span itemprop="birthDate">August 16, 1954</span>)
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar-theatrical-trailer.html"
itemprop="trailer">Trailer</a>
</div>
```

Σε γενικές γραμμές, όσο περισσότερα μεταδεδομένα προστεθούν τόσο το καλύτερα. Ωστόσο, σαν γενικός κανόνας ακολουθείται ότι το μόνο περιεχόμενο που αξίζει να σημανθεί είναι ό,τι είναι εμφανές στους χρήστες και έχει νόημα για αυτούς και ότι το κρυφό περιεχόμενο, όπως π.χ. τα κρυφά divs, κλπ.

Για περισσότερες λεπτομέρειες και παραδείγματα αξίζει κανείς να επισκεφθεί το http://schema.org/ και να πλοηγηθεί ανάμεσα στα πολυάριθμα αντικείμενα που έχουν κατηγοριοποιηθεί και προτυποποιηθεί.

4.4 DeepGraphs specification

Παρατηρώντας κανείς το internet όπως είναι σήμερα, θα έλεγε ότι είναι πλήρως αυτοματοποιημένο. Σελίδες που ενσωματώνουν πολυάριθμες υπηρεσίες, εγγενείς (εξυπηρετούνται από τον ίδιο web server) ή εξωτερικές (εξυπηρετούνται από τρίτους servers), που με την σειρά τους μπορεί να συνδέονται και να επικοινωνούν με άλλα συστήματα και να πυροδοτούν μία αλυσίδα από αντιδράσεις σε όλα τα μέρη του κόσμου. Αυτά μπορούν να υλοποιηθούν σήμερα, με υπάρχουσες τεχνολογίες. Παρά τις τεράστιες προόδους της τεχνολογίας ωστόσο, ακόμα εντοπίζονται κενά και επομένως υπάρχει χώρος για έρευνα, βελτίωση και επέκταση.

Με βάση τα όσα γνωρίζουμε μέχρι τώρα για τις web τεχνολογίες, η ανάγκη των χρηστών για αλληλεπίδραση με τα διάφορα services, χωρίς όμως να ξέρει εκ των προτέρων την πραγμάτωσή τους (πως ακριβώς λειτουργούν) ή ακόμα και η συνεργασία διάφορων services ταυτόχρονα είναι σε καλό στάδιο. Αντίθετα, η τεχνολογία σε ότι αφορά την δημιουργία τεχνητών πρακτόρων, οι οποίοι θα μπορούσαν αυτόνομα να εκτελούν εργασίες στο web για τις οποίες θα μπορούσα αξιόπιστα και με δυναμικό τρόπο να αποφασίσουν – πάντοτε με βάση τους στόχους που τους έχουν αναθέσει – είναι το λιγότερο ανεπαρκής. Προσπάθειες που είχαν γίνει για χρήση της OWL και OLW 2 δεν είχαν ικανοποιητικά αποτελέσματα στην κλίμακα του Web μιας και η πολυπλοκότητα των reasoners τους είναι πολλαπλά εκθετική, λόγω της εγγενούς αναδρομικότητας των αλγορίθμων.

Σε αυτή την παράγραφο θα μιλήσουμε για το DeepGraphs πρότυπο, το οποίο βασίζεται στις τεχνολογίες του σημασιολογικού ιστού που περιγράψαμε παραπάνω (Κεφάλαιο 3 & 4) και απώτερο σκοπό έχει να εξυπηρετήσει το τεχνολογικό κενό που αναδείξαμε σχετικά με τους αυτόνομους πράκτορες στο web. Το DeepGraphs specification αποτελεί ουσιαστικά έναν επίσημο τρόπο να περιγράφει κανείς Restful Web APIs τα οποία βασίζονται πάνω στο JSON-LD και πηγαίνει πέρα από την σημερινή κατάσταση με το να προτείνει την δυνατότητα

συντονισμού πολλαπλών clients, να αναλύει και να αποφασίζει σχετικά με νέες δραστηριότητες σε πραγματικό χρόνο.

Η λογική του προτύπου αυτού είναι η δημιουργία έξυπνων πρακτόρων που να μπορούν να ακολουθούν αυτόματα υπερσυνδέσμους με τον ίδιο τρόπο όπως το κάνουμε και εμείς οι άνθρωποι. Οι πράκτορες αυτοί θα πρέπει να έχουν όλη την πληροφορία που χρειάζεται για να επιτύχουν έναν στόχο και έχοντας APIs τα οποία εκθέτουν την πληροφορία τους, εμπλουτισμένη με σημασιολογικά στοιχεία, να επικοινωνούν με άλλα συστήματα και να πλοηγούνται αυτόνομα στο internet. Απώτερος στόχος δηλαδή του προτύπου είναι να αυτοματοποιήσει πλήρως βήματα και διαδικασίες στο internet με σκοπό την απλούστερη αλληλεπίδραση του χρήστη με νέες υπηρεσίες. Στις μέρες μας, σενάρια που αφορούν την αλληλεπίδραση του ανθρώπου με νέες υπηρεσίες υλοποιούνται από την εκάστοτε υπηρεσία, όπου στην πράξη ο προγραμματιστής αυτής έχει αναλάβει την πρόκληση να υλοποιήσει με κώδικα όλες τις πιθανές περιπτώσεις που έχει προεπιλέξει να αφήσει στην πλατφόρμα του. Ας δούμε τι σημαίνει αυτό μέσα από ένα σύντομο παράδειγμα. Ένας χρήστης μπαίνει για πρώτη φορά σε μία υπηρεσία και πρέπει να εγγραφεί σε αυτήν. Αντί να εισάγει όλα του τα στοιχεία από την αρχή ο χρήστης μπορεί να προτιμήσει να δώσει τα στοιχεία του μέσω μίας άλλης υπηρεσίας, για παράδειγμα να χρησιμοποιήσει “Google signup” ή “Facebook signup” και να πάρει όλα τα στοιχεία του (όνομα, επώνυμο, φύλλο, κλπ) αλλά και φωτογραφίες κατευθείαν από εκεί. Στο σενάριο αυτό σήμερα, ο χρήστης έχει όσες επιλογές του δίνει ο προγραμματιστής, εφόσον δεν είναι δυνατόν να προβλέψει όλα τα πιθανά σενάρια αλλά και να υλοποιήσει κάθε πιθανή σύνδεση με τις υπηρεσίες αυτές. Χρησιμοποιώντας το DeepGraphs προσπαθούμε να απλοποιήσουμε ακριβώς τέτοιες καταστάσεις, αυτοματοποιώντας τες όσο γίνεται περισσότερο.

Ένα ακόμα σενάριο που τώρα φαίνεται αδύνατο να λειτουργήσει, αλλά είναι μία από τις βασικές καινοτομίες που σκοπεύει να εισάγει το DeepGraphs είναι ο κάθε αυτόνομος πράκτορας στο web να μπορεί δυναμικά να αλληλεπιδράσει με νέα affordances χωρίς να έχει πρότερη γνώση για αυτά. Παίρνοντας από τον server ότι νέα πληροφορία χρειάζεται και ακολουθώντας κάποια συνήθη βήματα, όπως αυτά που περιγράφονται αμέσως μετά, θα είναι σε θέση να αντιμετωπίσει κάθε κατάσταση

- Ποιος είναι ο στόχος? – ελέγχει με βάση την πληροφορία από τον server αν ο στόχος αυτός έχει νόημα στο συγκεκριμένο ιστοχώρο
- Ποιες είναι οι πιθανές σειρές δράσεων προς τον στόχο? – εφόσον ελεγχθεί ότι ο στόχος μπορεί να υλοποιηθεί, εξετάζεται αν υπάρχουν σαφείς οδηγίες για να ακολουθήσει ο πράκτορας
- Πως μπορούν οι δράσεις αυτές να υλοποιηθούν? – ο πράκτορας πρέπει να έχει έναν οδηγό σχετικά με το πως να υλοποιήσει το κάθε βήμα
- Ανάλυση αποτελεσμάτων – εξετάζεται η επιτυχία κάθε βήματος
- Επίτευξη στόχου – εξετάζεται αν ο στόχος επετεύχθει

Ακολουθώντας τον παραπάνω σκελετό, είναι φανερό ότι ο πράκτορας δεν δεσμεύεται από τίποτα και μπορεί να επεξεργαστεί ό,τι πληροφορία του δώσει ο server ή ο σημασιολογικός ιστός. Επομένως μπορεί δυναμικά να προστεθούν λειτουργικότητες σε υπάρχοντα ψηφιακά αντικείμενα. Για παράδειγμα έστω ότι από τον σημασιολογικό ιστό (π.χ. schema.org) υπάρχει η πληροφορία ότι μία φωτογραφία είναι ένα προς πώληση προϊόν (affordance "buyProduct") και έχει κανονικά τιμή, προμηθευτή, κλπ. Μία φωτογραφία ενός χρήστη είναι στο Facebook και ο πράκτοράς μου κάνει αναζήτηση για μία τέτοια φωτογραφία. Μπορεί να την αγοράσει, χωρίς το Facebook να είναι άμεσα μία πλατφόρμα αγοραπωλησιών.

Ελπίζουμε ότι στις επόμενες υποπαραγράφους το DeepGraphs specification θα γίνει ακόμα πιο ξεκάθαρο, μέσα από την παρουσίαση της μεθοδολογίας που προτείνεται και ενός αναλυτικού παραδείγματος.

4.4.1 Προτεινόμενη μεθοδολογία

Η κεντρική ιδέα είναι ότι ένας server που λειτουργεί με Hypermedia πρότυπα μπορεί να βοηθήσει τον client να εξερευνήσει τον τρόπο με τον οποίο θα μπορέσει να επιτύχει τον επιθυμητό στόχο. Θα οδηγήσει τον client μέσα από έναν χάρτη με τα σημεία εισαγωγής (entry points) και την λογική πίσω από την υλοποίηση και την σημασιολογία.

Η προσέγγιση που προτείνεται είναι βασισμένη στο Hydra-LD, όπου ο server αποστέλλει μέσα στην απάντησή του και το αυτόματο πεπερασμένων καταστάσεων με όλες τις πιθανές πράξεις σε μορφή SWRL κανόνων. Επομένως χρησιμοποιώντας συνήθη λεξικά σε RDF και OWL μπορούμε να δημιουργήσουμε γενικευμένους clients που το μόνο που γνωρίζουν είναι ο στόχος που θέλουν να επιτύχουν και το πως να διαβάζουν έναν τέτοιο χάρτη. Το Hydra επιλέχθηκε σαν δομικό στοιχείο αφενός γιατί είναι μία πολλά υποσχόμενη προσπάθεια της W3C και συνοδεύεται από μία ολοένα αναπτυσσόμενη κοινότητα, αφετέρου παρέχει το απαραίτητο λεξικό επιτρέποντας μία client-server επικοινωνία που χρησιμοποιεί JSON-LD format.

Θα προσπαθήσουμε να περιγράψουμε πως λειτουργεί βήμα προς βήμα ένας πράκτορας που βασίζεται στο πρότυπο αυτό. Αρχικά ο πράκτορας στέλνει ένα "GET" request στον server στο βασικό μονοπάτι (το οποίο για λόγους ευκολίας δηλώνεται ως αρχικό σημείο). Παρατηρείστε ότι ο πράκτορας δεν έχει εκ των προτέρων καμία σχετική γνώση εκτός από το URL που πρέπει να στείλει το αίτημα και τι θέλει να κάνει. Οτιδήποτε άλλο του είναι τελείως άγνωστο, ακόμα και το αν λειτουργεί ο server ή όχι το αγνοεί πλήρως. Ο server (αν και εφόσον λειτουργεί και το αίτημα ήταν σωστό) στέλνει τον hypermedia χάρτη με κάποια από τα URLs που περιλαμβάνει μαζί με την περιγραφή τους, έτσι ώστε ο client να ξέρει όλο ή μέρος του τοπίου του συστήματος. Ο client τώρα στον χάρτη που έλαβε πρέπει να διαβάσει μία συγκεκριμένη και κατανοητή ροή εργασιών. Αυτό συμβαίνει με το αντίστοιχο SWRL που

ο server στέλνει, καθώς στην δική μας προσέγγιση οι SWRL κανόνες είναι μία σειρά από βήματα που μπορεί ο client να καταλάβει και να ακολουθήσει. Ανεξάρτητα με τις τεχνολογίες που χρησιμοποιούνται αυτό που είναι σημαντικό να ξεκαθαριστεί είναι ότι ο client αυτό που κάνει είναι να ακολουθεί ουσιαστικά ένα FSM (για αυτό άλλωστε έχουν επιλεχθεί hypermedia web τεχνολογίες, εφόσον από τη φύση τους περιλαμβάνουν την έννοια της αυτόματης μηχανής). Το SWRL επιλέχθηκε σαν γλώσσα σειριοποίησης για αυτά τα FSMs γιατί είναι ένα W3C πρότυπο και συνδυάζει την λογική των κανόνων με την OWL.

Στον χάρτη ο πράκτορας βρίσκει το σημείο έναρξης και κάθε φορά πλέον που μία κατάσταση γίνεται αληθής μετά από μία συναλλαγή (με τον server) ο client προχωράει στην επόμενη κατάσταση μέσω του hypermedia χάρτη του API και ανακαλύπτει τα επόμενα βήματα που πρέπει να εκτελέσει και πως. Στέλνει λοιπόν το επόμενο αίτημα στον server, όπως έχει περιγραφεί στον χάρτη ότι πρέπει να γίνει το αίτημα, και πλέον είναι στην ευχέρεια του server να βγάλει τα συμπεράσματά του. Έτσι κάπως θα κυλήσει η διαδικασία μέχρι κάποια στιγμή ο πράκτορας να φτάσει τον στόχο του (να φτάσει κάποιο τελικό σημείο – end point) ή να αρνηθεί ο server να τον εξυπηρετήσει για κάποιον λόγο (π.χ. bad request, server unavailable, unauthorized access, κλπ).

Το DeepGraphs είναι κυρίως ένα RDFS λεξικό και είναι σχεδιασμένο για να είναι απλό και επαναχρησιμοποιήσιμο για τους developers. Οι DeepGraphs συμβάσεις περιστρέφονται γύρω από 4 απλές έννοιες:

- Affordances: κλάσεις που αναπαριστούν τις πιθανές δράσεις που ένας πράκτορας μπορεί στην πραγματικότητα να εκτελέσει ως προς ένα ψηφιακό πόρο
- Processes: ιδιότητα που ορίζει σχετικά πότε τα affordances μπορούν να εκτελεστούν παράλληλα και πότε σειριακά
- Constraints: κανόνες που περιγράφουν την ακεραιότητα των περιορισμών των συναλλαγών και τα βήματα αυτών των συναλλαγών, δηλαδή το FSM. Αυτά τα FSM αποτελούνται από τις σειρές βημάτων για όλα τα πιθανά σενάρια και για κάθε κατάσταση. Η διαχείριση σφαλμάτων σε μη προβλεπόμενες καταστάσεις, επίσης περιγράφεται σε αυτό το κομμάτι του αρχείου
- Duration: ιδιότητα για τα processes. Το web είναι ένας πολύ κομματιασμένος χώρος πράγμα που σημαίνει ότι μία συναλλαγή που συμβαίνει σε ένα μέρος σε μία συγκεκριμένη χρονική ζώνη είναι πάρα πολύ πιθανό κομμάτια της να εκτελεστούν με έναν διαμοιρασμένο τρόπο σε ένα άλλο μέρος με άλλη χρονική ζώνη. Με λίγα λόγια, μιλάμε για την έννοια του σχετικού χρόνου στο ίντερνετ που προστέθηκε στο HTTP 1.1 και η οποία είναι απαραίτητη για την παραλληλοποίηση των processes

Σύμφωνα με την ανωτέρω μεθοδολογία οι πόροι και οι δράσεις περιγράφονται με βάση την ορολογία των συνδεδεμένων δεδομένων και τις αρχές του σημασιολογικού ιστού, επομένως ούτε ο client ούτε ο server είναι δεμένοι με κάποια συγκεκριμένη υλοποίηση ή κάποιο σύνολο πράξεων. Οπότε νέα affordances μπορούν να προστεθούν δυναμικά σε υπάρχοντες πόρους και να αποφασίζουν την συμπεριφορά τους με βάση νέους ή υπάρχοντες κανόνες. Αν για παράδειγμα, προστίθεται ένα καινούργιο affordance σε έναν πόρο, π.χ. η δράση «αναζήτηση» σε μία υπάρχουσα λίστα, ή απλά επεκτείνει μία υπάρχουσα γενικευμένη δράση, τότε ο πράκτορας έχει τουλάχιστον ένα σημείο έναρξης για αυτό. Με την προσέγγιση αυτή νέα affordances μπορούν να προστεθούν είτε από developers είτε από την κοινότητα προτυποποίησης και ο server μπορεί να επιλέξει αν θα τα εκτελέσει ή όχι. Φυσικά οι αποφάσεις θα παίρνονται ανάλογα με το πως είναι εκ των προτέρων προγραμματισμένος ο server να αντιδρά σε τέτοιες καταστάσεις. Για παράδειγμα θα μπορούσε να είναι έτσι φτιαγμένος ώστε δυναμικά να προσπαθεί να λύσει οποιαδήποτε νέα προσθήκη affordance σύμφωνα με τους ορισμούς του. Εναλλακτικά θα μπορούσε οτιδήποτε είναι πέρα από τα γνωστά του affordances να το απορρίπτει σαν αίτημα και να μην το εξυπηρετεί.

4.4.2 Παράδειγμα DeepGraphs

Ας προσπαθήσουμε να συγκεντρώσουμε όσα είπαμε μέχρι τώρα για το DeepGraphs specification σε ένα παράδειγμα για να γίνει κατανοητό τι ακριβώς είναι, πως λειτουργεί και τι προσφέρει στο web. Για να το κάνουμε αυτό θα επιλέξουμε ένα παράδειγμα που είναι οικείο σε όλους μας. Θα δούμε ένα απλό παράδειγμα μίας ηλεκτρονικής αγοράς ενός βιβλίου. Συγκεκριμένα, θα λέμε ότι ο χρήστης Γιάννης θέλει να αγοράσει από το ηλεκτρονικό κατάστημα Foo το βιβλίο Moby Dick.

Πριν ξεκινήσουμε με το πως υλοποιείται αυτή η διαδικασία στο DeepGraphs specification, ας δούμε πως γίνεται στο web που όλοι γνωρίζουμε και χρησιμοποιούμε σήμερα, με τις υπάρχουσες τεχνολογίες, με την ελπίδα να γίνει στην συνέχεια ξεκάθαρα η αντιδιαστολή.

Τα βήματα που πρέπει σήμερα να ακολουθήσει ένας χρήστης για να βρει και να αγοράσει ένα βιβλίο από το ηλεκτρονικό κατάστημα Foo είναι ακριβώς τα ακόλουθα (και με την σειρά που παρουσιάζονται):

1. Αναζήτηση βιβλίου με βάση κάποια λέξη-κλειδί στο ηλεκτρονικό κατάστημα (π.χ. *"Moby Dick"*)
2. Επιλογή βιβλίου από την λίστα αποτελεσμάτων
3. Εισαγωγή στο καλάθι
4. Checkout του καλαθιού
5. Επιλογή ταχυδρομικής αποστολής
6. Συνέχεια και πληρωμή
7. Επιλογή τρόπου πληρωμής
8. Αναμονή για επιβεβαίωση συναλλαγής

9. Ανάκτηση απόδειξης

Στα παραπάνω βήματα είναι προφανές ότι θα μπορούσαν να προστεθούν να αφαιρεθούν κάποια ανάλογα με το ηλεκτρονικό κατάστημα. Παράδειγμα, ένα άλλο κατάστημα μπορεί να απαιτούσε από τον χρήστη να συνδεθεί με λογαριασμό πριν προχωρήσει στην πληρωμή, ενώ σε κάποιο άλλο μπορεί να μην είναι απαραίτητα τα στάδια που αφορούν το καλάθι. Στο κατάστημα Foo πάντως τα πράγματα κυλούν έτσι και ο Γιάννης θα χρειαστεί να εκτελέσει ένα προς ένα όλα τα 9 βήματα για να πάρει το βιβλίο και την απόδειξη πληρωμής του. Παρακάτω ακολουθεί και μία φωτογραφία που δείχνει περίπου μία αντίστοιχη διαδικασία σε μορφή UML.

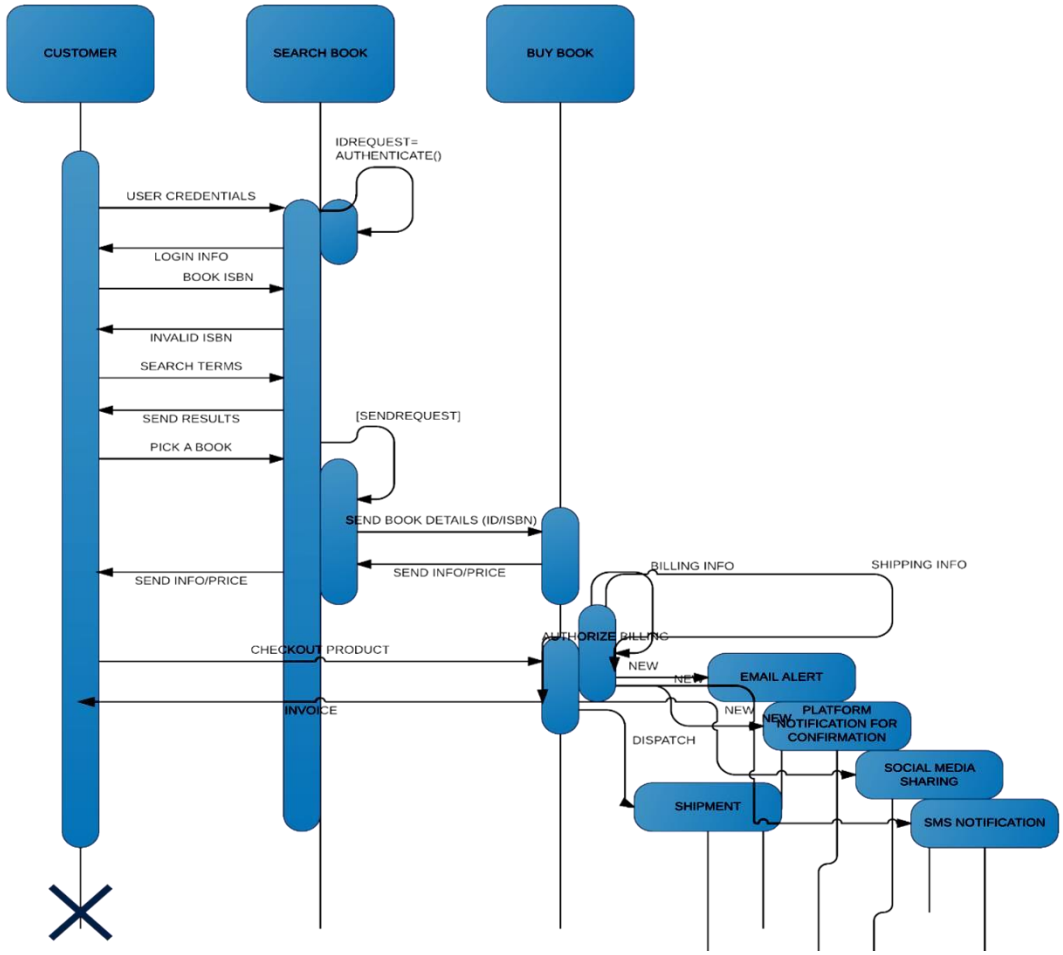


Figure 5 Ακολουθιακό διάγραμμα πριν το DeepGraphs specification

Ας δούμε τώρα τι θα έκανε αντίστοιχα ο πράκτορας που δουλεύει στο DeepGraphs οικοσύστημα. Έστω ότι ο Γιάννης λέει στον πράκτορα να ψάξει το βιβλίο Moby Dick από το κατάστημα Foo. Ο πράκτορας έχει αρκετή πληροφορία για να φέρει ένα ικανοποιητικό αποτέλεσμα. Ξέρει ότι ψάχνει ένα βιβλίο χρησιμοποιώντας την φράση "Moby Dick" από ένα ηλεκτρονικό κατάστημα που λέγεται Foo. Τα βήματα που θα ακολουθήσει είναι τα εξής:

1. Αποστολή "GET" αιτήματος για να πάρει τον αρχικό χάρτη
2. Επιβεβαίωση ότι υπάρχει μονοπάτι για αυτή την δράση (affordance)

3. Αναζήτηση με βάση κάποια λέξη-κλειδί στο ηλεκτρονικό κατάστημα
4. Φιλτράρισμα μόνο με βάση το schema.org/Book αντικείμενο
5. Επιλογή αντικειμένου από τη λίστα
6. Εισαγωγή στο καλάθι
7. Checkout του καλαθιού
8. Επιλογή ταχυδρομικής αποστολής
9. Συνέχεια και πληρωμή
10. Επιλογή τρόπου πληρωμής
11. Αναμονή για επιβεβαίωση συναλλαγής
12. Ανάκτηση απόδειξης

Με βάση τα παραπάνω βλέπουμε ότι ο πράκτορας κάνει κάποια ελάχιστα επιπλέον βήματα σε σχέση με τον χρήστη, ωστόσο τα περισσότερα βήματα παραμένουν τα ίδια και ο πράκτορας αλληλεπιδρά με το κατάστημα σαν να ήταν ο ίδιος ο Γιάννης. Εκ μέρους του παίρνει δυναμικά αποφάσεις (π.χ. επιλογή αποστολής ή επιλογή τρόπου πληρωμής, κλπ) με σκοπό να επιτύχει τον στόχο του, δηλαδή τη δουλειά που του ανέθεσε ο χρήστης. Το ακόλουθο διάγραμμα επίσης βοηθάει στην οπτική απεικόνιση όσων είπαμε.

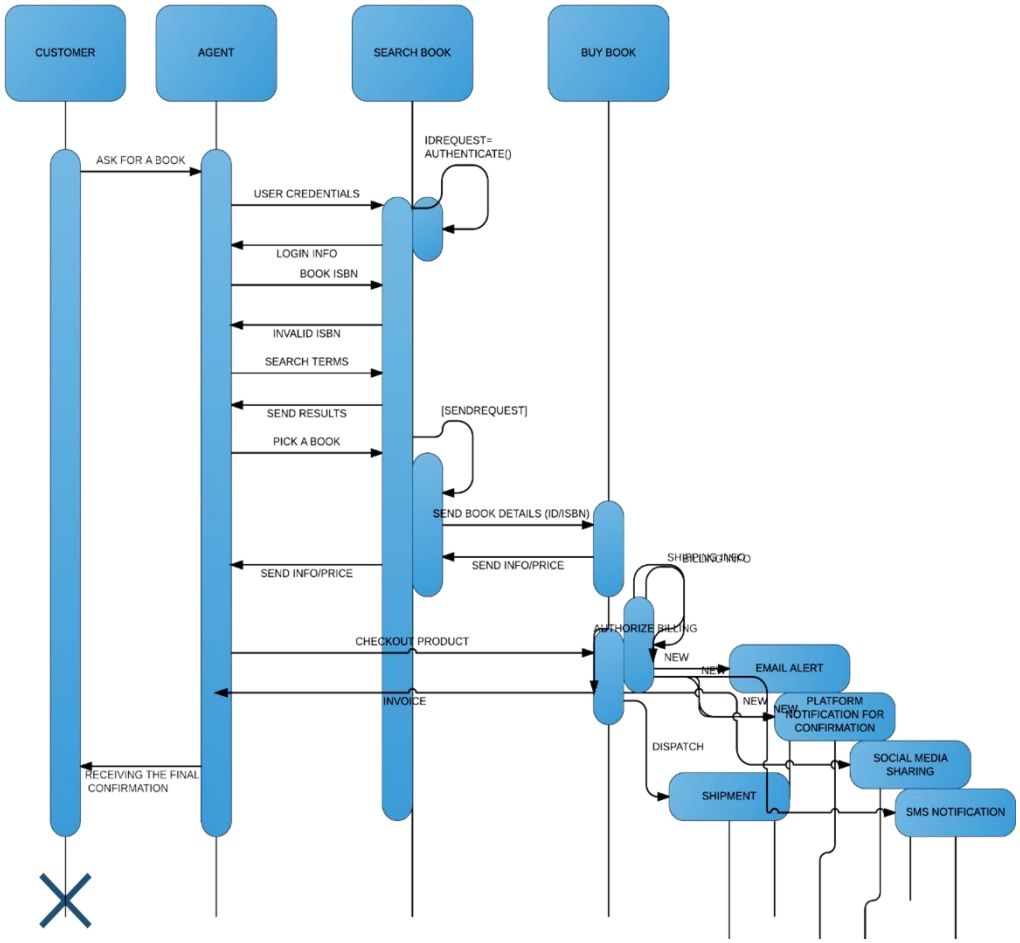


Figure 6 Ακολουθιακό διάγραμμα με το DeepGraphs specification

Αξίζει να σημειωθεί ότι η διαδικασία που περιγράψαμε περιλαμβάνει πόρους που είναι εύκολα αναγνωρίσιμοι, όπως Βιβλίο, Λίστα Βιβλίων (ή Προϊόντων), Διεύθυνση, Παραγγελία,

κλπ, και λεπτομερώς περιγεγραμμένοι στον σημασιολογικό ιστό. Ένα παράδειγμα που αναφέρεται στο http://schema.org/Book σε JSON-LD είναι το ακόλουθο:

```
<script type="application/ld+json">
{
  "@context":  "http://schema.org/",
  "@id": "#record",
  "@type": "Book",
  "additionalType": "Product",
  "name": "Le concerto",
  "author": "Ferchault, Guy",
  "offers":{
      "@type": "Offer",
      "availability": "http://schema.org/InStock",
      "serialNumber": "CONC91000937",
      "sku": "780 R2",
      "offeredBy": {
          "@type": "Library",
          "@id": "http://library.anytown.gov.uk",
          "name": "Anytown City Library"
      },
      "businessFunction": "http://purl.org/goodrelations/v1#LeaseOut",
      "itemOffered": "#record"
    }
}
</script>
```

Το ίδιο βέβαια συμβαίνει και με τις δράσεις (actions ή affordances). Ενέργειες όπως «Ψάξε ένα Αντικείμενο» είναι πλήρως προσδιορισμένες στο http://schema.org/SearchAction. Στην συνέχεια δίνουμε το πιο απλό παράδειγμα της ενέργειας αυτής, όπως κάναμε και παραπάνω:

```
<script type="application/ld+json">
  // John searched for 'What is the answer to life the universe and everything?'.
{
```

```
4.        "@context": "http://schema.org",
5.        "@type": "SearchAction",
6.        "agent": {
7.          "@type": "Person",
8.          "name": "John"
9.        },
10.       "query": "What is the answer to life the universe and
   everything?"
11.     }
12.     </script>
```

Τέλος αξίζει να δώσουμε και μία γραφική αναπαράσταση του γράφου που ο πράκτορας θα ακολουθήσει κατά γράμμα για να εκτελέσει τον σκοπό του.

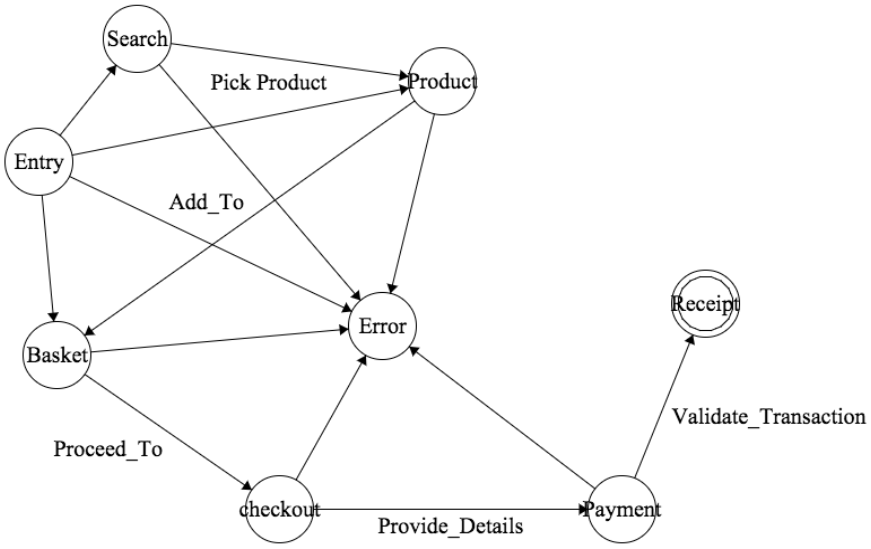


Figure 7 FSM για το παράδειγμα του ηλεκτρονικού καταστήματος

4.4.3 Συμπεράσματα

Μετά τα παραπάνω είναι ίσως κατανοητό πως έχει βελτιωθεί η συνολική εμπειρία του χρήστη με την εισαγωγή του DeepGraphs specification. Συγκεκριμένα αν κοιτάξει κανείς το δεύτερο ακολουθιακό διάγραμμα του προηγούμενου παραδείγματος (Figure 6), που αφορά την αγορά ενός βιβλίου από ηλεκτρονικό κατάστημα μέσω ενός αυτοματοποιημένου πράκτορα που δρα στα πλαίσια του DeepGraphs, είναι αρκετά προφανές το πόσο λιγότερο αλληλεπιδρά ο χρήστης πλέον με την όλη διαδικασία.

Πέρα από αυτό όμως, στο παραπάνω παράδειγμα κερδίσαμε κάτι ακόμα που μπορεί να μην φαίνεται άμεσα στον χρήστη. Ο πράκτορας εφόσον δρα αυτόνομα μπορεί να εκτελέσει παράλληλα κάποιες εργασίες, εφόσον αυτό επιτρέπεται. Μπορεί να αποφασίσει δηλαδή

κάποια αιτήματα να τα παραλληλοποιήσει, εάν δεν υπάρχουν εξαρτήσεις ανάμεσά τους, όπως θα μπορούσε να είναι για παράδειγμα η επιλογή αποστολής και η πληρωμή.

Φυσικά, μπορεί κανείς να αντιτάξει ότι συνολικά το DeepGraphs επιφέρει κάποιας μορφής καθυστέρηση, καθώς προσθέτει ένα επιπλέον επίπεδο επικοινωνίας (βλ. Figure 6), και φόρτο στο δίκτυο, καθώς επιπρόσθετα αρχεία πρέπει να μετακινούνται τακτικά για να γίνεται η ανταλλαγή των FSM. Στο πρώτο επιχείρημα θα μπορούσαμε να απαντήσουμε ότι, καθώς ο πράκτορας είναι αυτόνομος και ανεξάρτητος από τον χρήστη όχι μόνο μπορεί να αυτοματοποιήσει τις διαδικασίες και άρα να τις εκτελέσει πολλαπλάσιες φορές γρηγορότερα από τον πιο εξοικειωμένο χρήστη μίας πλατφόρμας, αλλά και να παραλληλοποιήσει διαδικασίες, μειώνοντας σημαντικά τον χρόνο της όλης διαδικασίας. Όσο για το δεύτερο επιχείρημα, είναι αλήθεια ότι το πρότυπο αυτό βασίζεται στην μετακίνηση των χαρτών, που μέχρι τώρα δεν χρειαζόταν. Ωστόσο, το μέγεθος των αρχείων αυτών δεν είναι σημαντικό ώστε να αποτελεί μεγάλο πρόβλημα ή απειλή για το δίκτυο. Αν αναλογιστεί κανένας ότι το πιο περίπλοκο site μπορεί να περιλαμβάνει έως και 100 affordances τα οποία μπορεί να θέλουν περίπου 10 γραμμές το καθένα να περιγραφούν, μιλάμε για αρχεία που έχουν το πολύ 1000 γραμμές όλες κι όλες. Άρα στις ακραίες περιπτώσεις κάνουμε λόγο για μερικά KB. Και πάλι όμως αν θεωρηθεί ότι υπάρχει λόγος ανησυχίας μπορούμε πολύ εύκολα να καταφύγουμε σε παραδοσιακές μεθόδους μείωσης μεγέθους των δεδομένων, όπως είναι η συμπίεση των αρχείων, ή η λύση της επαναχρησιμοποίησης δεδομένων με τεχνικές caching. Οι υπάρχουσες REST υποδομές επιτρέπουν να κρατώνται στην μνήμη απαντήσεις ή μέρος αυτών στον client ή σε ενδιάμεσους proxies, έτσι ώστε να μπορούν να χρησιμοποιηθούν στο μέλλον εάν χρειαστεί.

5

Μεθοδολογία

Το παρόν κεφάλαιο είναι ίσως το κομβικότερο της παρούσας εργασίας. Αρχικά αποσαφηνίζεται το γιατί χρειάζεται ή είναι επιθυμητή η δυνατότητα των γενικευμένων πρακτόρων που δρουν στα πλαίσια του DeepGraphs να διαπραγματεύονται, χρησιμοποιώντας συγκεκριμένα παραδείγματα από την πραγματική ζωή. Εξηγείται ακόμα το γιατί και πως μπορεί μία διαπραγματευτική διαδικασία να ενταχθεί στο DeepGraphs specification, αναπτύσσοντας προσεκτικά μία μεθοδολογία.

5.1 Διαπραγμάτευση στα πλαίσια του DeepGraphs specification

Μετά από όλα τα προηγούμενα κεφάλαια που θέτουν το γνωσιακό επίπεδο που απαιτείται για την κατανόηση της παρούσας εργασίας, μπορούμε προχωρήσουμε σε πιο πρακτικά θέματα. Ίσως μέχρι τώρα κάποια πράγματα είναι ξεκάθαρα, όπως για παράδειγμα το γιατί είναι απαραίτητη η ύπαρξη ενός προτύπου όπως το DeepGraphs για την εκτεταμένη διεξαγωγή διαπραγματεύσεων ανάμεσα σε αυτοματοποιημένους πράκτορες στα πλαίσια του διαδικτύου.

Ήδη στο κεφάλαιο 2 που παρουσιάσαμε την διαπραγμάτευση σαν δομημένη και συμπαγή διαδικασία, κάναμε λόγο για τις πολυάριθμες έρευνες και εργασίες που έχουν γίνει τα τελευταία λόγια στον τομέα της αυτοματοποιημένης διαπραγμάτευσης. Πολλές από αυτές τις

ερευνητικές δουλειές συνοδεύονταν μάλιστα και από υλοποιήσεις, πράγμα που σημαίνει ότι έχουν υπάρξει στο παρελθόν πράκτορες (υπό μορφή προγράμματος) που η δουλειά τους ήταν να διαπραγματεύονται με άλλους αυτοματοποιημένους πράκτορες. Κάποιες είχαν και πολλά υποσχόμενα αποτελέσματα, κάτι που δεν είναι καθόλου ασήμαντο ειδικά όταν μιλάμε για έρευνες που γίνονταν για πολύπλευρες διαπραγματεύσεις (περισσότερα από 2 αντίπαλα στρατόπεδα), για διαπραγματεύσεις που επέτρεπαν την αλλαγή στρατηγικής ενδιάμεσα στην διαδικασία (η επιλογή στρατηγικής είναι μία ευαίσθητη και πολύπλοκη διαδικασία, που μπορεί να έχει σοβαρό αντίκτυπο, θετικό ή αρνητικό, στην έκβαση της διαπραγμάτευσης) ή επέβαλλαν την ανταλλαγή επιχειρημάτων. Οι παραπάνω περιπτώσεις που μόλις αναφέρθηκαν είναι διαδικασίες εξαιρετικά περίπλοκες ήδη στην ανθρώπινη διαπραγμάτευση, πόσο μάλλον σε αυτοματοποιημένες διαπραγματεύσεις χωρίς την μεσολάβηση του ανθρώπινου παράγοντα.

Γιατί λοιπόν όλη αυτή η γνώση δεν χρησιμοποιήθηκε ποτέ στην παραγωγή, έστω και σε πειραματικό στάδιο? Υπάρχουν πολλοί λόγοι που θα μπορούσε κανένας να πει, ωστόσο ο σημαντικότερος είναι ότι έως τώρα δεν υπήρχαν οι κατάλληλες τεχνολογίες για να στηρίξουν ένα τέτοιο εγχείρημα. Έστω ότι φτιάχναμε έναν πράκτορα που αποκλειστικός του στόχος είναι να διαπραγματεύεται τιμές, ένας πράκτορας αρκετά ειδικού σκοπού, αλλά ταυτόχρονα γενικευμένος σε βαθμό να του επιτρέπει να διεκδικεί πάνω από ένα αντικείμενα και σε πάνω από μία σελίδες – για την ακρίβεια σε όλο το internet –, χωρίς την ύπαρξη του DeepGraphs. Ο πράκτορας αυτός θα έπρεπε από πριν να ξέρει ακριβώς πως να πλοηγηθεί μέσα στο internet (πώς να καλέσει τις σελίδες και τι να ζητήσει από αυτές) αλλά και πως να αντιμετωπίσει τους αντίπαλους πράκτορες. Κάτι τέτοιο είναι αδύνατο να συμβεί στο internet όπως το ξέρουμε τώρα, αρχικά γιατί οι σελίδες δεν είναι αρκετά ώριμες ώστε να προσφέρουν τέτοια λειτουργικότητα (δεν μπορούν να καθοδηγήσουν έναν πράκτορα) αλλά και επειδή δεν υπάρχει ενιαίος και συμφωνημένος (προτυποποιημένος) τρόπος για να διεξάγεται μία τέτοια διαδικασία. Η κάθε σελίδα ακολουθεί και θέτει τους δικούς της κανόνες για το πως θα πραγματοποιούνται οι ενέργειες σε αυτήν, είτε αυτό είναι η αναζήτηση, η αγορά ενός αντικειμένου ή η διαπραγμάτευση μίας τιμής.

Στα επόμενα κεφάλαια θα παρουσιάσουμε λεπτομερώς το πως έχουμε σκεφτεί και υλοποιήσει μία απλή διαδικασία διαπραγμάτευσης, κάνοντας πάντα χρήση του DeepGraphs specification. Θα μιλήσουμε αναλυτικά για τα βήματα που πραγματοποιούνται, τις υποθέσεις που κάναμε για να διευκολύνουμε την ανάλυσή μας και απλουστεύσουμε το πολύπλοκο κόσμο του διαδικτύου και τέλος κάποιες πιθανές επεκτάσεις που θα είχαν νόημα στο παράδειγμά μας.

5.2 Παραδείγματα διαπραγμάτευσης στο DeepGraphs specification

Στο κεφάλαιο 4 που παρουσιάσαμε για πρώτη φορά το DeepGraphs πρότυπο, περιγράψαμε ένα απλό αλλά περιεκτικό παράδειγμα ενός πράκτορα που θα μπορεί να πλοηγηθεί αυτόνομα σε ένα ηλεκτρονικό κατάστημα, να αναζητήσει, να επιλέξει και να αγοράσει ένα προϊόν μόνο του και χωρίς την παρέμβαση ενός χρήστη. Ας δούμε λοιπόν γιατί η επέκταση του πρωτοκόλλου είναι χρήσιμη έτσι ώστε να καλύπτει και να διαχειρίζεται σωστά καταστάσεις που απαιτούν ο πράκτορας να διεκδικήσει κάτι από άλλους όμοιούς του.

Πολλά από τα ηλεκτρονικά καταστήματα αυτή τη στιγμή στον παγκόσμιο ιστό αφήνουν ανοιχτή την τιμή και επιτρέπουν την δυνατότητα στον χρήστη να κάνει τις δικές του προσφορές στο σύστημα. Έπειτα συνήθως το σύστημα επικοινωνεί με τον πάροχο (provider), ο οποίος αποδέχεται ή απορρίπτει την τιμή. Στη συνέχεια ο χρήστης μπορεί να επιλέξει να κάνει νέα πρόταση εάν το επιθυμεί, με βάση τα νέα δεδομένα, ελπίζοντας να επιτύχει αυτή την φορά. Η διαδικασία αυτή μπορεί να επαναληφθεί πολλές φορές και ίσως να μην επιτύχει και ποτέ. Περιγράφοντας αυτή την διαδικασία είναι ίσως φανερό ότι είναι αρκετά δυσάρεστη και κουραστική, καθώς ο χρήστης μπορεί μόνο ηλεκτρονικά να επικοινωνήσει με τον πάροχο και μόνο μέσω της πλατφόρμας (δεν έχει άμεση επικοινωνία μαζί του). Φυσικά οι πλατφόρμες προσπαθούν να κάνουν την διαδικασία όσο γίνεται πιο φιλική, ωστόσο ο χρήστης θα προτιμούσε σε κάθε περίπτωση αυτά να γίνονται αυτοματοποιημένα και χωρίς την δική του μεσολάβηση.

Υποθέτουμε λοιπόν πως ο πράκτορας του παραδείγματος του κεφαλαίου 4 καταλήγει σε ένα κατάστημα σαν αυτό που περιγράφηκε προηγουμένως, δηλαδή αφήνει την τιμή ανοιχτή για τον χρήστη. Τότε μπορεί να συμβούν 2 πράγματα, είτε ο πράκτορας θα πρέπει να το αγνοήσει γιατί δεν μπορεί να καταλάβει τον γράφο (περιλαμβάνει ένα στάδιο που δεν μπορεί να εκτελέσει) στην περίπτωση που το μαγαζί είναι συμβατό με το DeepGraphs specification είτε αυτά τα καταστήματα θα εξαιρεθούν τελείως από το πρωτόκολλο και θα εξακολουθούν να λειτουργούν συμβατικά ως προς τους χρήστες, σαν να είναι αόρατα για τους αυτόματους πράκτορες. Κάτι τέτοιο φυσικά δεν μπορεί να συμβεί γιατί θα άφηνε εκτός της οικονομίας ένα μεγάλο μέρος καταστημάτων και άρα είναι απαραίτητη η ένταξη τέτοιων διαδικασιών στο πρωτόκολλο αυτό.

Πέρα από το παραπάνω πολύ προφανές και κατανοητό παράδειγμα που ο πράκτορας καλείται να διαπραγματευτεί άμεσα με άλλους πράκτορες του διαδικτύου για ένα πολύ συγκεκριμένο και μετρήσιμο πράγμα (τιμή), υπάρχουν κι άλλες λιγότερο ενστικτώδεις ή προφανείς εφαρμογές. Ένα παράδειγμα πολύ αντιπροσωπευτικό είναι η διεκδίκηση μεγαλύτερου εύρους

δικτύου. Έστω ότι σε ένα δημόσιο δίκτυο είναι συνδεδεμένοι διάφοροι χρήστες και εκτελούν κάποιες ενέργειες ο καθένας. Για να γίνει ακόμα πιο χειροπιαστό θα κάνουμε το σενάριο ακόμα πιο συγκεκριμένο, λέγοντας ότι έχουμε 3 χρήστες (ο αριθμός είναι τελείως τυχαία επιλεγμένος απλά και μόνο για τις ανάγκες του παραδείγματος) όπου ο ένας βλέπει ένα βίντεο στο κινητό του, ο άλλος γράφει ένα κείμενο στον υπολογιστή του σε ένα τοπικό πρόγραμμα και ο τρίτος στέλνει ένα email. Και οι τρεις χρήστες κάνουν χρήση του δικτύου ανεξάρτητα με το αν το χρησιμοποιούν συνειδητά ή όχι. Για παράδειγμα ο χρήστης που φαίνεται να μην κάνει καμία χρήση του δικτύου εκείνη την στιγμή εφόσον χρησιμοποιεί ένα desktop application, μπορεί να έχει φαινομενικά μικρότερες απαιτήσεις δικτύου σε σχέση με εκείνον που παρακολουθεί ένα βίντεο, ωστόσο μπορεί στο background να τρέχουν δουλειές, όπως updates ή queued workloads που είχαν μείνει παγωμένες μέχρι να βρει η συσκευή πρόσβαση σε δίκτυο. Επομένως, παίρνουμε σαν παραδοχή ότι και οι τρεις έχουν ανάγκη από δίκτυο. Είναι ξεκάθαρο όμως ότι αν μοιράζουμε το εύρος στα 3 θα είναι κακή η εξυπηρέτηση για εκείνον που έχει τις μεγαλύτερες και πιο άμεσες απαιτήσεις δικτύου. Αν από την άλλη το αφήσουμε τελείως ανοιχτό τότε υπάρχει ο φόβος εκείνος που έχει τις μεγαλύτερες ανάγκες να καταναλώνει όλο το δίκτυο, με αποτέλεσμα οι υπόλοιποι να μην εξυπηρετούνται καθόλου. Σαν συνέπεια, οι δουλειές των άλλων χρηστών, σημαντικές ή όχι, βαριές ή όχι δεν θα εκτελεστούν ποτέ (π.χ. το mail του τρίτου χρήστη αν και είναι μία εύκολη και «ελαφριά» δουλειά μπορεί να καθυστερήσει ασυνήθιστα πολύ). Κάτι τέτοιο δεν είναι καθόλου παράλογο, εφόσον γνωρίζουμε ότι υπάρχουν τέτοιες εφαρμογές και διαδικασίες που όντως καταναλώνουν όσους πόρους μπορούν να βρουν χωρίς σεβασμό προς τις άλλες διεργασίες ή χρήστες του δικτύου (π.χ. Netflix).

Μία τέτοια κατάσταση θα μπορούσαν να διαχειριστούν οι αυτοματοποιημένοι πράκτορες των χρηστών εάν και εφόσον είναι ενεργοποιημένοι στις συσκευές των χρηστών. Οι πράκτορες χωρίς να ενοχλήσουν τον χρήστη, καταλαβαίνουν ότι βρίσκονται σε δημόσιο δίκτυο και μοιράζονται πόρους. Βγάζουν λοιπόν μία προσέγγιση για το τι ανάγκες έχουν σε internet και «βγαίνουν» προς τα έξω για να διεκδικήσουν εκείνο που θέλουν. Αν στο παραπάνω παράδειγμα είχαμε μόνο τους 2 τελευταίους χρήστες, το πιθανότερο είναι ότι οι πράκτορες θα έπαιρναν ακριβώς το μερίδιο που θα επεδίωκαν με την πρώτη κιόλας προσπάθεια (χωρίς αντι-προτάσεις) εφόσον τα συμφέροντά τους δεν έρχονται σε αντίθεση – οι απαιτήσεις δικτύου και των δύο είναι πολύ χαμηλές και μπορούν άνετα να εξυπηρετηθούν ταυτόχρονα από την ίδια γραμμή χωρίς πρόβλημα. Με την εισαγωγή του τρίτου χρήστη όμως το τοπίο αλλάζει τελείως. Οι πράκτορες πλέον έρχονται σε άμεση σύγκρουση συμφερόντων και μπαίνουν πλέον σε κανονική διαδικασία διαπραγμάτευσης προσπαθώντας να βρουν την βέλτιστη για όλους λύση, έτσι ώστε να μην αλλοιωθεί όσο γίνεται η εμπειρία κανενός χρήστη. Στο παράδειγμα αυτό το σύστημα κάποια στιγμή θα ισορροπήσει σε μία κατάσταση

που θα έχει γίνει αποδεκτή από όλους. Το σύστημα θα βγει εκτός ισορροπίας, αν η ποιότητα δικτύου για κάποιο λόγο αλλάξει (έστω και προσωρινά), αν κάποιος συμμετέχων αποχωρίσει ή μπει στο δίκτυο ή αν κάποιος αλλάξει τις απαιτήσεις του (π.χ. κάποιος ξεκινάει να ακούσει μουσική από το internet). Στις παραπάνω περιπτώσεις οι πράκτορες θα πρέπει να μπουν και πάλι σε διαδικασία διαπραγμάτευσης.

Θα μπορούσε κανένας να αντιπροτείνει ότι αυτό είναι μία αχρείαστη εφαρμογή, εφόσον έχουμε ήδη αλγορίθμους και τρόπους να μοιράζουμε το bandwidth. Ωστόσο κανένας δεν είναι αρκετά εξελιγμένος ώστε να είναι ad hoc ρυθμιζόμενος και μάλιστα τόσο σωστά προσαρμοσμένα στις απαιτήσεις των χρηστών. Ακόμα αξίζει να σημειωθεί ότι όλα αυτά γίνονται ερήμην του χρήστη, δηλαδή ο πράκτορας από μόνος του ανακαλύπτει την ανάγκη και την επιλύει χωρίς να «ενοχλεί» τον χρήστη και χωρίς να επιβαρύνει το μηχάνημα σημαντικά υπολογιστικά.

Το πως λειτουργεί μία διαπραγμάτευση στα πλαίσια του DeepGraphs αλλά και το ότι είναι υπαρκτή η ανάγκη ενσωμάτωσής της μέσα στο πρωτόκολλο έγινε κατανοητό μέσα από τα παραπάνω παραδείγματα. Παρόλα αυτά θα ήθελα να αναφέρω ένα τελευταίο παράδειγμα που μοιάζει να βγαίνει από επιστημονική φαντασία, ωστόσο αποτελεί μία απλή εφαρμογή του γνωστού Internet of Things (IoT) που μπορεί να μην απέχει και πολύ από την σημερινή εποχή.

Ένα πολύ τακτικό φαινόμενο στην καθημερινότητα και σίγουρα όλοι το έχουμε αντιμετωπίσει είναι ο διαμοιρασμός των σπιτικών εργασιών. Στις συμβατικές οικογένειες το μοίρασμα των εργασιών γίνεται με βάση κάποιον απλό αλγόριθμο που προσπαθεί να είναι όσο γίνεται πιο δίκαιος. Ας πάρουμε σαν παράδειγμα την ακόλουθη οικογένεια για να κυλήσει το παράδειγμα πιο εύκολα: Μενέλαος (πατέρας), Αφροδίτη (μητέρα), Ευαγγελία (κόρη), Χρήστος (γιος). Στην παραπάνω οικογένεια, θα εξετάσουμε μία μόνο «αγγαρεία» πως θα μπορούσε να κατανάμεται μέσα στην εβδομάδα και συγκεκριμένα το πλύσιμο των πιάτων: Δευτέρα-Τρίτη-Πέμπτη τα πιάτα πλένει ο Μενέλαος, Τετάρτη η Ευαγγελία και Παρασκευή-Σάββατο-Κυριακή η μητέρα. Ο παραπάνω αλγόριθμος είναι ότι πιο δίκαιο μπορούσαν να υλοποιήσουν, λαμβάνοντας υπόψιν την ηλικία των ατόμων, τις υποχρεώσεις τους και φυσικά τις ικανότητες τους (π.χ. ο Χρήστος είναι πολύ μικρός για να πλένει τα πιάτα, επομένως εξαιρείται από αυτή την δουλειά). Το πρόγραμμα αυτό μπορεί να λειτουργεί στο 90% των περιπτώσεων σωστά για την οικογένεια, ωστόσο υπάρχει ένα 10% όπου τα πράγματα μπορεί να παρεκκλίνουν από το φυσιολογικό και να προκύψουν έκτακτες καταστάσεις που κανένας δεν είχε προβλέψει. Στην περίπτωση αυτή θα μπορούσε να υπάρχει εφεδρικό πλάνο ή τα κενά να καλύπτονται τυχαία και ανάλογα με την κρίση του καθενός. Βλέπουμε όμως ότι το πως κατανέμεται μία και μόνο εργασία από όλες που μπορεί να έχει ένα σπίτι, παρουσιάζει

πολυπλοκότητες και δυσκολίες που τα άτομα της οικογένειας μπορεί να μην είναι σε θέση ή να μην θέλουν να λύσουν.

Σκεφτείτε τώρα στο παραπάνω παράδειγμα αν το πρόγραμμα αυτό δεν προκύπτει από τα ίδια τα μέλη της οικογένειας, αλλά από την συνεργασία (ή καλύτερα την διαπραγμάτευση) αυτοματοποιημένων πρακτόρων. Οι πράκτορες δηλαδή του καθενός (personal agents) προσπαθούν να προασπιστούν τα συμφέροντα των χρηστών τους και έτσι μέσα από μία διαδικασία διαπραγμάτευσης να προκύπτει ένα κοινώς αποδεκτό πρόγραμμα. Κάθε φορά που θα προκύπτει κάτι απροσδόκητο καλούνται οι πράκτορες πάλι να «συνεδριάσουν» για να προκύψει από κοινού μία απόφαση για το ποιος πρέπει να αναλάβει να καλύψει το κενό που δημιουργήθηκε.

Είναι φανερό ότι η διαπραγμάτευση αυτή προφανώς πρέπει να λαμβάνει υπόψιν της όλα όσα αναφέραμε παραπάνω σχετικά με περιορισμούς που αφορούν τις ικανότητες των ατόμων ή το πόσο χρόνο συνολικά θα πρέπει να αφιερώνουν σε αγγαρείες, το σταθερό πρόγραμμα των χρηστών, κλπ. Καταλαβαίνουμε λοιπόν ότι το σενάριο περιλαμβάνει πολύ περισσότερους περιορισμούς σε σχέση με τα προηγούμενα που παρουσιάσαμε και επομένως είναι αρκετά πιο περίπλοκο. Υπάρχουν όμως κι άλλοι λόγοι που το καθιστούν πολύ πιο δύσκολο πρόβλημα σε σχέση με τα υπόλοιπα 2 που προηγήθηκαν και δεν αναφερόμαστε μόνο ως προς την πολυπλοκότητα του αλγορίθμου. Πρώτα από όλα έννοιες όπως «ανικανότητα του χρήστη να πλύνει τα πιάτα λόγω ηλικίας» ή ακόμα καλύτερα «ακαταλληλότητα του χρήστη να πλύνει τα πιάτα λόγω ηλικίας» είναι κάτι δύσκολο να αντιληφθεί και να κωδικοποιήσει ένα πρόγραμμα, ενώ είναι κάτι πολύ ενστικτώδες και οικείο για έναν άνθρωπο. Επίσης, το εβδομαδιαίο πρόγραμμα από μόνο του ενώ μπορεί να ποσοτικοποιηθεί και να χωριστεί σε ώρες και συγκεκριμένες εργασίες είναι σαφώς λιγότερο μετρήσιμο σε σχέση με τα προηγούμενα μεγέθη που εξετάσαμε, όπως ήταν η τιμή και το διαθέσιμο bandwidth, και άρα δυσκολότερο να διαπραγματευτείς επ'αυτού. Τέλος, ίσως το δυσκολότερο από όλα είναι ότι η διαπραγμάτευση αυτή πρέπει να γίνει αποδεκτή από όλα τα μέλη της οικογένειες και να ακολουθείται. Πρόκειται για κάτι που επηρεάζει άμεσα τα μέλη της οικογένειας, την καθημερινή τους δράση και κατά κάποιο τρόπο τείνει να τους περιορίσει. Οι άνθρωποι σπάνια είναι δεκτικοί στο να αναλαμβάνουν επιπλέον δουλειές ή να υπακούνε σε εντολές τρίτων, οπότε είναι εξαιρετικά σημαντικό να νιώσουν ότι μπορούν να εμπιστευτούν το αποτέλεσμα και άρα να ξέρουν ότι η κατανομή έγινε ακριβοδίκαια.

Το παράδειγμα αυτό όπως είπαμε, φαντάζει μακρινό, κι αυτό γιατί η έννοια του προσωπικού πράκτορα δεν είναι ακόμα αρκετά διαδεδομένη. Η αλήθεια είναι ότι παρά τις προσπάθειες που γίνονται (συμπεριλαμβανομένης και της παρούσας εργασίας) υπάρχουν ακόμα κάποια θέματα τεχνικής φύσης που πρέπει ακόμα να λυθούν για να φτάσουμε στην ευρύτατη χρήση των πρακτόρων με την μορφή που περιγράφηκαν παραπάνω. Χωρίς να μπούμε σε πολλές

λεπτομέρειες μπορούμε απλά να αναφέρουμε κάποια παραδείγματα, πατώντας πάνω στο παράδειγμα της διανομής των σπιτικών εργασιών. Το πρώτο πράγμα που μπορεί να παρατηρήσει κανένας είναι το ποιος θα ενημερώνει τους πράκτορες έτσι ώστε να στείλουν τις απαραίτητες πληροφορίες και να αρχίσουν να διαπραγματεύονται. Θα είναι κάποιος κεντρικός πράκτορας «του σπιτιού»? Επίσης που θα εκτελείται ο υπολογισμός, δηλαδή η διαδικασία της διαπραγμάτευσης? Θα υπάρχει κάποιος server μέσα στο σπίτι που θα εκτελεί τέτοιου είδους υπολογισμούς ή θα γίνεται κάπου στο cloud? Όλα αυτά, και άλλα ακόμα, είναι κενά στην τεχνολογία που πρέπει να απαντηθούν και να λυθούν προκειμένου κάποια στιγμή να μπορούν αυτά που τώρα περιγράφουμε θεωρητικά να χρησιμοποιηθούν μαζικά από τους χρήστες.

5.3 Μεθοδολογία για το DeepGraphs specification

Στο κεφάλαιο 4.3, έγινε μία προσπάθεια περιγραφής του DeepGraphs specification. Στα πλαίσια της καλύτερης κατανόησης του προτύπου, κρίθηκε προτιμότερο να εντάξουμε τόσο την ανάπτυξη της μεθοδολογίας όσο και το σχετικό παράδειγμα σε εκείνη την ενότητα. Επομένως, οι τεχνολογίες που προτείνονται και η μεθοδολογία που ακολουθείται στο DeepGraphs specification γενικά αλλά και ειδικότερα στην υλοποίηση που επιχειρήσαμε και θα περιγράψουμε στο επόμενο κεφάλαιο, θεωρείται ότι καλύπτεται πλήρως στο αναφερόμενο κεφάλαιο (Κεφάλαιο 4.3) και ο αναγνώστης παραπέμπεται εκεί για περισσότερες λεπτομέρειες.

5.4 Μεθοδολογία διαπραγμάτευσης στο DeepGraphs specification

Έχοντας κατά νου την μεθοδολογία του DeepGraphs specification είναι μάλλον προφανές ότι ο πράκτορας που περιγράφεται, αν και γενικευμένος, είναι απόλυτα καθοδηγούμενος από τους ονομαζόμενους «χάρτες». Με πιο απλά λόγια οι πράκτορες δεν αυτοσχεδιάζουν, αλλά λαμβάνουν σαφείς οδηγίες για το τι πρέπει να κάνουν κάθε φορά για να επιτύχουν το στόχο τους, και άρα το μόνο που πραγματικά χρειάζεται είναι να έχουν τις απαραίτητες «γνώσεις» και μηχανισμούς (να είναι κατάλληλα προγραμματισμένοι ώστε να είναι συμβατοί με το DeepGraphs specification) για να: αλληλεπιδρά με τον server (να κάνει το αρχικό “GET” request), αναγνωρίζουν και να πλοηγούνται σωστά χρησιμοποιώντας το χάρτη. Με την λογική αυτή για να μπορέσουμε να σκεφτούμε πως μία διαπραγμάτευση θα μπορούσε να

ενταχθεί στο πρότυπο αυτό, αρκεί να περιγράψουμε την διαπραγμάτευση σαν διακριτά βήματα ενός γράφου πολύ σαφώς καθορισμένα.

Πρώτο βήμα λοιπόν είναι να εντάξουμε τη διαδικασία «διαπραγμάτευση» μέσα σε έναν χάρτη. Προφανώς το βήμα αυτό πρέπει να γίνεται αφού ο πράκτορας έχει επιλέξει το προϊόν και πριν πληρώσει για αυτό, επομένως τα βήματα του πράκτορα θα είναι τα ακόλουθα:

1. Αποστολή "GET" αιτήματος για να πάρει τον αρχικό χάρτη
2. Επιβεβαίωση ότι υπάρχει μονοπάτι για αυτή την δράση (affordance)
3. Αναζήτηση με βάση κάποια λέξη-κλειδί στο ηλεκτρονικό κατάστημα
4. Φιλτράρισμα μόνο με βάση το schema.org/Book αντικείμενο
5. Επιλογή αντικειμένου από τη λίστα
6. Διαπραγμάτευση τιμής
7. Εισαγωγή στο καλάθι
8. Checkout του καλαθιού
9. Επιλογή ταχυδρομικής αποστολής
10. Συνέχεια και πληρωμή
11. Επιλογή τρόπου πληρωμής
12. Αναμονή για επιβεβαίωση συναλλαγής
13. Ανάκτηση απόδειξης

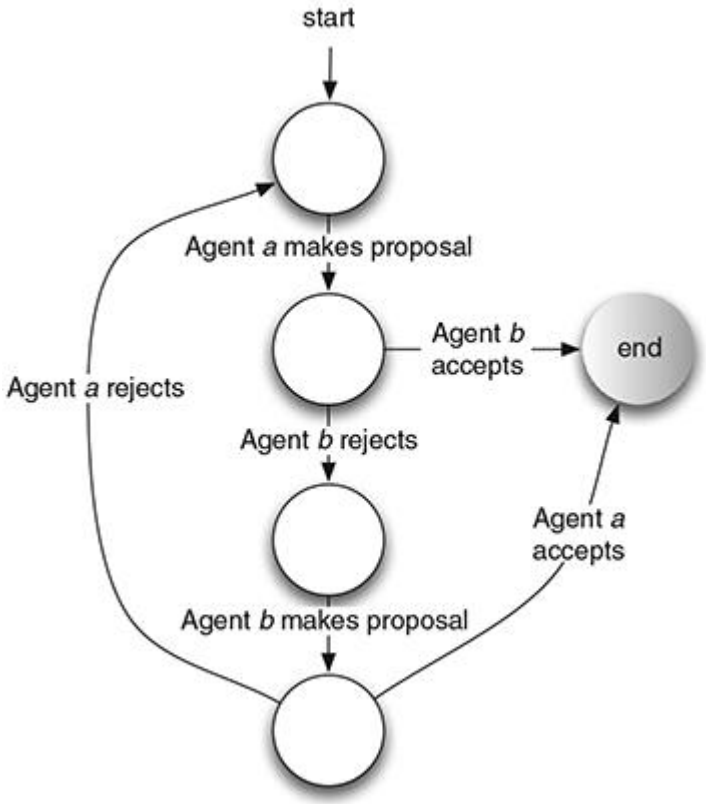
Το παραπάνω παράδειγμα φυσικά παίρνει σαν δεδομένο ότι η διαπραγμάτευση ήταν επιτυχής για να προχωρήσει στα επόμενα βήματα.

Σε κάθε περίπτωση παρατηρούμε ότι η διαπραγμάτευση δεν ήταν τίποτα παραπάνω από ένα επιπλέον βήμα που πρέπει να εκτελέσει ο agent. Απλώς στον γράφο που στέλνει ο server μπήκε ένας επιπρόσθετο κόμβος. Φαίνεται επομένως ότι η διαπραγμάτευση δεν είναι ούτε ξένη ούτε ασύμβατη με το DeepGraphs πρωτόκολλο, αντιθέτως ενσωματώνεται εύκολα σαν έννοια.

Το δύσκολο κομμάτι είναι το αμιγώς κομμάτι της διαπραγμάτευσης, πως δηλαδή οι πράκτορες θα κάνουν προσφορές και αντιπροτάσεις, πως θα διεκδικούν, πως θα επιχειρηματολογούν, κλπ. Για την προτυποποίηση αυτή βασιστήκαμε σε μία πολύ συγκεκριμένη και περιορισμένη μαθηματική προτυποποίηση που έχει τις βάσεις της στην θεωρία παιγνίων και ονομάζεται «ανυπόμονων παικτών». Θα δούμε στην συνέχεια τι λέει το συγκεκριμένο μοντέλο, αλλά για αρχή πάμε να δούμε με ποια κριτήρια επιλέχθηκε.

Κάθε διαπραγμάτευση -ανάμεσα σε δύο μέρη- ακολουθεί μία πορεία όπως αυτή που φαίνεται στον ακόλουθο γράφο (Εικόνα 1). Ο συμμετέχων Α προτείνει κάτι, όπου ο Β θα επεξεργαστεί. Αν ο Β το αποδεχτεί η διαπραγμάτευση κλείνει επιτυχώς. Αν ο Β αρνηθεί τότε κάνει μία αντιπρόταση και πλέον είναι στην ευχέρεια του Α να αρνηθεί ή να δεχτεί την πρόταση. Είναι φανερό ότι τα συμφέροντα των Α και Β είναι αντίθετα (αν δεν υπήρχαν αντίπαλα συμφέροντα δεν θα υπήρχε και ανάγκης για διαπραγμάτευση), επομένως ο Α θα μπορούσε συνέχεια να αρνείται οποιαδήποτε πρόταση του Β μέχρι να περάσει το δικό του.

Το ίδιο θα μπορούσε να ακολουθεί και ο Β, με αποτέλεσμα η διαπραγμάτευση να μην τερματίζει ποτέ.



Εικόνα 1 Γράφος διαπραγμάτευσης

Θεωρητικά αυτό είναι κάτι που γίνεται. Πράγματι αν αφήσεις 2 πράκτορες λογισμικού με σαφείς εντολές να υπερασπιστούν όσο καλύτερα γίνεται τα συμφέροντα του χρήστη, η διαδικασία θα κατέληγε κάπως έτσι. Στον πραγματικό κόσμο όμως, το αποτέλεσμα της διαπραγμάτευσης δεν είναι το μόνο που έχει αξία. Ο χρόνος από την στιγμή που ξεκινάει η διαπραγμάτευση μέχρι την στιγμή που τελειώνει είναι εξίσου σημαντικό με το αποτέλεσμα. Στο παράδειγμά μας δηλαδή, τόσο ο Α όσο και ο Β προτιμούν να βρουν μία κοινώς αποδεκτή λύση γρήγορα, ακόμα και αν είναι υποβέλτιστη, παρά να τραβήξει πολύ σε χρόνο.

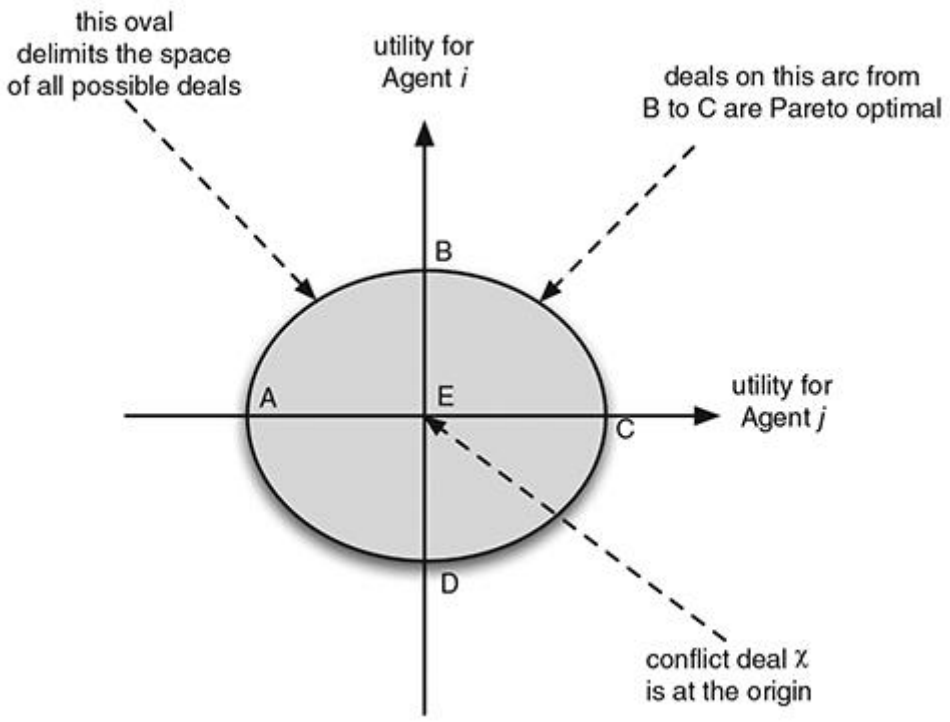
Το μοντέλο που αναφέραμε παραπάνω περιγράφει ακριβώς αυτό το πράγμα, πως δηλαδή όσο περνάει η ώρα το αντικείμενο «χάνει» αξίας και για τις δύο πλευρές. Συγκεκριμένα το μοντέλο προβλέπει έναν παράγοντα Δέλτα (συμβολίζεται με «δ») που αντιπροσωπεύει το κατά πόσο μειώνεται η αξία για κάθε μέρος της διαπραγμάτευσης σε κάθε γύρο.

Ας δούμε τώρα τα μαθηματικά πίσω από αυτό που περιγράφηκε παραπάνω.

Έστω ότι έχουμε δύο συμμετέχοντες σε μία διαπραγμάτευση, τον πωλητή S και τον αγοραστή Β. Ο πρώτος θέτει την κατώτερη τιμή που μπορεί να δεχτεί να πουλήσει το προϊόν, έστω ότι είναι v. Ο δεύτερος θέτει την ανώτερη τιμή που προτίθεται να δώσει για να πάρει αυτό το

αγαθό και είναι V. Η διαφορά ανάμεσα στις τιμές είναι x = V-v. Προφανώς η διαπραγμάτευση έχει νόημα αν V>v, αν V=v τότε άμεσα συμφωνούν και τα δύο μέρη στην ίδια τιμή, ενώ αν είναι V<v δεν υπάρχει λόγος να μπουν καν στην διαδικασία διαπραγμάτευσης (τα V και v θεωρούνται αυστηρά όρια – hard limits) . Η διαφορά στην τιμή x ισοδυναμεί με το βάρος που θα επωμιστεί κάθε μέρος. Για παράδειγμα αν ο S αναλάβει όλο το x, τότε σημαίνει ότι από το θεωρητικό κέρδος x που θα μπορούσε να είχε χάνει το 100%. Ενώ αν το βάρος αυτό το επωμιστεί ο B, τότε αυτός χάνει x χρήματα αφού ο S θα ήταν ευχαριστημένος και με v. Το πρόβλημα αυτό μοιάζει με το πρόβλημα διαμοιραμού μίας πίτας, όπου πρέπει να γίνει όσο γίνεται πιο δίκαια για να είναι όλοι ευχαριστημένοι.

Αξίζει να σημειωθεί ότι όλες οι λύσεις αυτού του μαθηματικού προβλήματος είναι βέλτιστες κατά Pareto. Pareto optimal (ή Pareto efficiency) ονομάζεται η κατάσταση στην διαδικασία διαμοιρασμού πόρων, όπου όποια αλλαγή και να κάνεις για να ευνοήσεις κάποιο μέρος θα γίνει πάντα σε βάρος κάποιου άλλου μέρους -τουλάχιστον ένα. Η εικόνα που φαίνεται παρακάτω περιγράφει με πολύ καθαρό οπτικό τρόπο την έννοια του Pareto optimal. Πάνω στον κύκλο είναι όλες οι πιθανές λύσεις του προβλήματος της πίτας που αναφέρθηκε παραπάνω. Κάθε σημείο πάνω στον κύκλο ανήκει στο σύνολο καταστάσεων που είναι pareto optimal, αλλά κάθε φορά καθώς κινείται πάνω στον κύκλο, όσο αυξάνεται η τιμή στον άξονα j π.χ. τόσο μειώνεται η τιμή στον άξονα i και αντίστροφα.



Εικόνα 2 The negotiation set

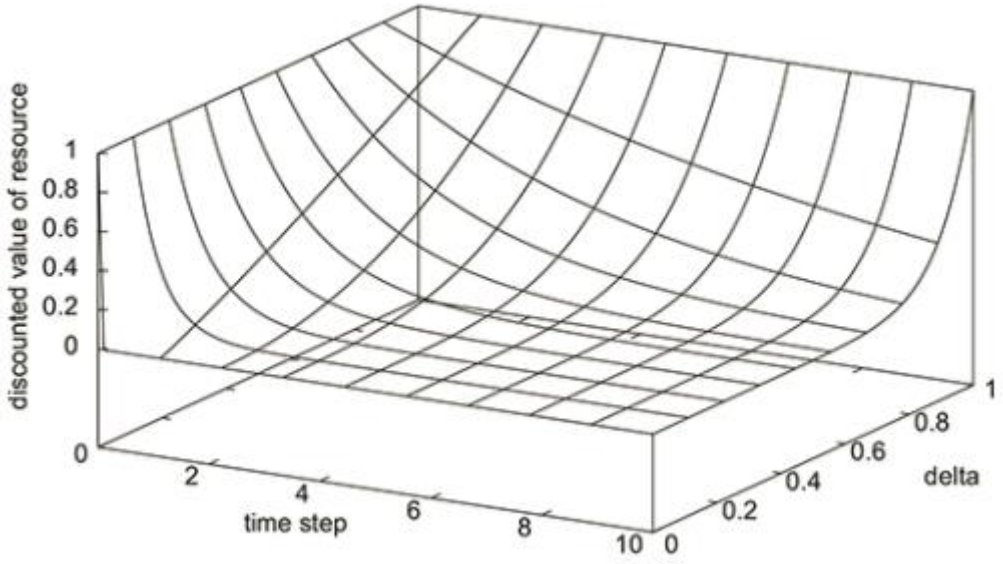
Αφού καταφέραμε να μοντελοποιήσουμε το κέρδος για καθένα μέρος, αρκεί τώρα να προτυποποιήσουμε μέσα σε αυτό και τον χρόνο.

Συνεχίζοντας στο παραπάνω σενάριο (με τους S και B), υποθέτουμε ότι πρώτος κάνει προσφορά ο B. Αν ο B προτείνει αυτό που πραγματικά θέλει, δηλαδή να προτείνει v, τότε ο S προφανώς θα απορρίψει την προσφορά. Όταν ο S θα κάνει προσφορά όμως, ο χρόνος είναι t+1, επομένως η αξία του προϊόντος για το B έχει πέσει, και άρα ο S θα κάνει προσφορά την καλύτερη πιθανή (για εκείνον) τιμή, που είναι $[(1-\delta_B)x+v]$. Παρατηρούμε όμως ότι όσο πάει η αξία και για τους δύο μειώνεται, οπότε όσο δεν αποδέχονται την λύση τόσο μικρότερο θα είναι το κέρδος.

Αν περιορίσουμε την παραπάνω διαδικασία σε συγκεκριμένους επιτρεπόμενους γύρους, τότε έχουμε το εξής σενάριο – υποθέτουμε 2 γύρους:

ο B που θα προτείνει πρώτος ξέρει ότι έχει μπροστά του 2 γύρους. Έστω ότι M είναι η μέγιστη τιμή που μπορεί να πάρει στο τέλος του δεύτερου γύρου. Άρα θα κάνει τους ακόλουθους συλλογισμούς, αν M είναι η πρόταση που παίρνω στον γύρο t. Αυτό σημαίνει ότι στον γύρο t-1 ο S έκανε πρόταση $(1- \delta_B *M)$. Επομένως στο αρχικό βήμα t-2 είχα $(1 – \delta_S(1 – \delta_B M))$. Αν θέλουμε λοιπόν να είναι $M = 1 – \delta_S(1 – \delta_B M)$ Προκύπτει ότι $M = (1- \delta_S) / (1 - \delta_S \delta_B)$. Αν κάνουμε το ίδιο και για το minimum θα βρούμε την ίδια τιμή, πράγμα που σημαίνει ότι αυτό είναι αποδεδειγμένα το μόνο ποσό που μπορεί να πάρει (μοναδικό equilibrium – σημείο ισορροπίας). Άρα μετά από 2 γύρους η μόνη λύση που μπορεί να βγει από αυτό το μαθηματικό μοντέλο είναι αυτή και άρα αν ο B από τον πρώτο γύρο υπολογίσει αυτό το μέγεθος και το προτείνει, τότε ο S θα το δεχτεί αυτόματα.

Η παραπάνω μεθοδολογία προφανώς εφαρμόζεται και για διαφορετικά βάθη χρόνου (π.χ. 3 ή 4 γύροι) και κάθε φορά θα δίνει μία μοναδική λύση. Άρα ακολουθώντας αυτή την τακτική μπορούμε χωρίς καθόλου κόστος χρονικά να έχουμε διαπραγματεύσεις που συγκλίνουν κάθε φορά. Αυτό που αλλάζει ανάλογα την διαπραγμάτευση και τα μέρη που συμμετέχουν είναι τα Δέλτα που παίζουν πολύ σημαντικό ρόλο. Το Δέλτα παίρνει πάτνα τιμές ανάμεσα στο 0 και το 1. Μεγάλες τιμές του δ δείχνουν μεγάλη ανυπομονησία, εφόσον η αξία μειώνεται κατά μεγάλο ποσοστό. Αντίθετα μικρότερες τιμές είναι ένδειξη μεγαλύτερης υπομονής. Στο γράφημα που ακολουθεί φαίνεται ενδεικτικά το πως η αλλαγή της τιμής του δ επηρεάζει την τιμή.



Εικόνα 3 Μεταβολή τιμής σε συνάρτηση με την τιμή του δ (0<δ<1)

6

Υλοποίηση διαπραγμάτευσης

Σε αυτό το κεφάλαιο θα αναλυθεί τεχνικά η υλοποίηση του αυτοματοποιημένου πράκτορα που περιγράφηκε αφαιρετικά στο κεφάλαιο της μεθοδολογίας (κεφάλαιο 5). Αρχικό βήμα για το Proof of Concept (PoC) που δημιουργήσαμε ήταν η δημιουργία ενός γενικευμένου πράκτορα, όπως αυτός θα λειτουργούσε στα πλαίσια του DeepGraphs. Αφιερωθήκαμε στο παράδειγμα που περιγράψαμε στο κεφάλαιο 4.4.2, πως δηλαδή ένας αυτοματοποιημένος πράκτορας παίρνει εντολή από τον χρήστη να αγοράσει ένα βιβλίο από ένα ηλεκτρονικό κατάστημα και φέρει σε πέρας όλη την αγορά μόνος του, χωρίς την παρέμβαση του χρήστη.

Στην συνέχεια επεκτείναμε τον ίδιο πράκτορα ώστε να μπορεί να διαπραγματεύεται με άλλα προγράμματα και τεχνητούς πράκτορες του διαδικτύου, πάλι πατώντας σε όλες τις αρχές και την μεθοδολογία που περιγράφηκε στο προηγούμενο κεφάλαιο, και πάντα μιλώντας για το συγκεκριμένο σενάριο.

6.1 Υλοποίηση πράκτορα στο DeepGraphs specification

6.1.1 Τεχνολογίες που χρησιμοποιήθηκαν

Στα πλαίσια της θεωρίας του DeepGraphs και της μεθοδολογίας που παρουσιάσαμε σε προηγούμενα κεφάλαια, κάναμε λόγο για τεχνολογίες όπως Hydra specification, JSON-LD και άλλες state of the art (κορυφαίες και σύγχρονες) τεχνολογίες. Οι τεχνολογίες αυτές είναι

απόλυτα συμβατές με το DeepGraphs πρότυπο από κάθε άποψη και αυτό με τη σειρά του χρησιμοποιεί τη λειτουργικότητά τους στο έπακρο. Δεν θα ήταν εύκολο δηλαδή να σκεφτεί κανείς ένα τέτοιο πρότυπο να υπάρχει στο web έξω από τα πλαίσια του σημασιολογικού ιστού, ούτε θα ήταν εύκολο να μεταφέρει πληροφορία χωρίς την χρήση του JSON-LD που είναι μία αρκετά στοχευμένη μέθοδος κωδικοποίησης συνδεδεμένων δεδομένων.

Στα πλαίσια όμως αυτού του πιλοτικού παραδείγματος που φτιάξαμε οι περισσότερες από αυτές τις τεχνολογίες παραλείφθηκαν, καθώς κρίθηκε ότι προσθέτουν μάλλον παρά αφαιρούν πολυπλοκότητα στην διαδικασία. Αυτό είναι λογικό να συμβαίνει σε μικρή κλίμακα και σε ένα περιορισμένο περιβάλλον όπου υπάρχει απόλυτη εμπιστοσύνη ανάμεσα στα συστήματα που επικοινωνούν (π.χ. server, βάση δεδομένων, clients, κλπ). Στο περιβάλλον αυτό είναι απόλυτα εφικτό και αποδεκτό να ανταλλάσσονται για παράδειγμα αρχεία Prolog (”.pl” files) -όπως θα δούμε ότι όντως χρησιμοποιήθηκαν παρακάτω- ενώ είναι απαγορευτικό να συμβαίνει στο ελεύθερο internet. Κάθε Prolog αρχείο είναι ένα εκτελέσιμο αρχείο, οπότε θα μπορούσε αν πέσει στα χέρια ενός υστερόβουλου χρήστη στο internet (π.χ. man in the middle attack) να το τροποποιήσει με τρόπο που να κάνει κακό τόσο στον άλλο χρήστη (π.χ. να υποδείξει στον πράκτορα να καταθέσει ένα χρηματικό ποσό) όσο και στα ίδια τα συστήματα (π.χ. να προκαλέσει τεχνητό μπλοκάρισμα στον client ή στον server). Αντίθετα με την μεταφορά JSON-LD αρχείων, ακόμα και να πέσουν σε λάθος χέρια, οι κίνδυνοι είναι σαφώς μικρότεροι.

Αυτό ήταν μόνο ένα παράδειγμα γιατί επιλέξαμε να αποκλείσουμε κάποιες από τις τεχνολογίες από την υλοποίησή μας. Κάποιες άλλες τεχνολογίες απλώς θεωρήθηκε ότι δεν προσφέρουν κάτι περισσότερο σε ότι αφορά την ανάδειξη της αξίας της ιδέας που προσπαθούμε να εξηγήσουμε και έτσι πολύ απλά εγκαταλείφθηκαν.

Οι τεχνολογίες που χρησιμοποιήθηκαν είναι οι ακόλουθες:

- SWI-Prolog – ο αυτοματοποιημένος πράκτορας γράφηκε σε γλώσσα Prolog. Η επιλογή έγινε με μόνο κριτήριο ότι το reasoning και η διάσχιση του γράφου γίνεται «φυσικά» και εύκολα χρησιμοποιώντας αυτή τη γλώσσα. Παρότι δεν είναι εξαιρετικά δημοφιλείς σε web εφαρμογές[2] για την συγκεκριμένη εργασία πληρούσε ικανοποιητικά -και με το παραπάνω- τις απαιτήσεις.
  Χωρίς να μπούμε σε πολλές λεπτομέρειες σχετικά με τον λογικό προγραμματισμό καλό θα ήταν να αναφέρουμε κάποια βασικά προχωρήσουμε στην παρουσίαση της υλοποίησης.
  Ο λογικός προγραμματισμός λειτουργεί χρησιμοποιώντας σχέσεις (relations) και ο

[2] Η Prolog είναι μία από τις δημοφιλέστερες και πιο ευρέως χρησιμοποιούμενες οικογένειες γλωσσών λογικού προγραμματισμού.

υπολογισμός γίνεται «τρέχοντας» ένα ερώτημα (query) πάνω στις σχέσεις αυτές. Η Prolog έχει ένα μόνο τύπο δεδομένων, τον όρο (term), πάνω στον οποίο δομούνται οι σχέσεις και τα ερωτήματα. Οι όροι μπορούν να είναι άτομα (atoms), νούμερα, μεταβλητές και σύνθετοι όροι (compound terms), όπως είναι π.χ. οι λίστες. Από αυτά αξίζει να αναλύσουμε λίγο την έννοια των ατόμων στην Prolog. Άτομα είναι γενικού σκοπού ονόματος που δεν έχουν κάποιο συγκεκριμένο νόημα για τον κώδικα και αντιμετωπίζονται κάπως σαν σταθερές. Παραδείγματα ατόμων είναι τα ακόλουθα: x, atom, ‘foo’, κλπ. Οι σχέσεις ορίζονται από προτάσεις (clauses). Οι προτάσεις είναι δύο ειδών, οι κανόνες (rules) και τα γεγονότα (facts). Οι κανόνες είναι της μορφής “Head :- Body” και διαβάζονται ως εξής: το head είναι αλήθεια αν το body είναι αλήθεια. Αν το Body της πρόταση δεν υπάρχει, π.χ. “Head.” τότε κάνουμε λόγο για γεγονός. Ένα γεγονός ταυτίζεται με έναν κανόνα της μορφής “Head :- true”, όπου “true” είναι το άτομο που έχει πάντα αληθή τιμή στην Prolog (αντίστοιχα υπάρχει και το “false”). Το Head μπορεί να έχει μόνο ένα κατηγόρημα (που παίρνει ένα ή περισσότερα ορίσματα), ενώ το Body μπορεί να αποτελείται από πολλαπλά ορίσματα που ενώνονται λογικά είτε με σύζευξη είτε με διάζευξη. Για παράδειγμα στην ακόλουθη πρόταση: “father(X) :- hasSon(X); hasDaughter(X).” λέμε ότι κάποιος είναι πατέρας αν έχει είτε κάποιον γιο είτε κάποια κόρη. Χρησιμοποιήσαμε την διάζευξη λοιπόν για να δώσουμε περισσότερα από ένα κατηγορήματα και αν κάποιο από αυτά είναι αληθές τότε ο X είναι πατέρας.

Τέλος αξίζει να κάνουμε αναφορά στον τρόπο που κάνει reasoning η Prolog. Η Prolog δέχεται ερωτήματα στα οποία προσπαθεί να ρωτήσει true ή false. Τα ερωτήματα μπορεί να είναι της μορφής “father(tom)”, όπου γίνεται σαφής ερώτηση για το αν ο Tom είναι πατέρας ή όχι, είτε της μορφής “father(X)”, όπου σε αυτή την περίπτωση η Prolog ψάχνει να βρει όλες τις τιμές που κάνουν αυτή την πρόταση αληθή. Και στις δύο περιπτώσεις ο τρόπος που αντιδρά η Prolog είναι ο ίδιος. Ψάχνει μέσα στα γεγονότα της και προσπαθεί να ενοποιήσει μεταβλητές με άτομα. Η ενοποίηση (unification) είναι το βασικότερο ίσως εργαλείο της Prolog και αυτό που της επιτρέπει να είναι τόσο type-agnostic. Θα δώσουμε ένα σύντομο παράδειγμα Prolog προγράμματος, απλά και μόνο για να γίνει ξεκάθαρος ο τρόπος που δουλεύει η Prolog.

Έστω ότι έχουμε την πρόταση που είδαμε πριν “father(X) :- hasSon(X); hasDaughter(X).” Αν ρωτήσουμε την Prolog “father(tom).” ή “father(X).” η Prolog θα απαντήσει false γιατί δεν ξέρει αν υπάρχει κάποιο άτομο στον κόσμο της που ικανοποιεί αυτές τις σχέσεις, δεν υπάρχουν γεγονότα (που είναι όπως ήδη είπαμε αδιαμφισβήτητες αλήθειες για την Prolog) για να μπορεί να συμπεράνει κάτι. Αν

προσθέσουμε στα παραπάνω το γεγονός “hasSon(tom).” (λέμε δηλαδή ότι ο tom έχει γιο) και κάνουμε τις ίδιες ερωτήσεις με πριν τότε η Prolog engine θα απαντήσει “true” και “X=tom” αντίστοχα. Στην πρώτη περίπτωση με την ερώτηση father(tom), η Prolog πήγε κατευθείαν στον κανόνα, ο οποίος για να ισχύει πρέπει να ισχύει κάτι από τα δύο, είτε hasSon(tom) είτε hasDaughter(tom) (παρατηρείτε ότι στην ίδια πρόταση η μεταβλητή X έχει παντού την ίδια τιμή). Παίρνει το πρώτο κατηγόρημα hasSon(tom) και ρωτάει αν ισχύει. Βρίσκει το γεγονός και απαντάει αυτόματα true. Αξίζει να παρατηρήσουμε ότι η Prolog δεν θα αναζητήσει καν το δεύτερο κατηγόρημα, εφόσον είναι διάζευξη και το πρώτο κατηγόρημα είναι αληθές. Στην δεύτερη ερώτηση η Prolog καλείτε να απαντήσει το father(X). Αναζητά το κατηγόρημα αυτό και ανατρέχει στον κανόνα. Πάλι εξετάζει το πρώτο κατηγόρημα (hasSon(X)) και ψάχνει στα γεγονότα της να βρει τιμές που κάνουν αυτό το κατηγόρημα αληθές. Βρίσκει το γεγονός hasSon(tom) και επιστρέφει αυτή την τιμή. Εδώ αξίζει η να σημειωθεί ότι η Prolog τερματίζει μεν στη εύρεση τιμής, αλλά αν της πει ο χρήστης να συνεχίζει να ψάχνει τιμές, θα συνεχίσει να κοιτάζει στα γεγονότα της και θα εξετάσει όλες τις εναλλακτικές μέχρι να μην έχει άλλες λύσεις. Στην δική μας περίπτωση π.χ. θα κοίταζε άλλη μία φορά το hasSon(X) και αν τυχόν υπάρχουν άλλα γεγονότα, μετά θα εξέταζε αν υπάρχουν hasDaughter(X) γεγονότα και μετά θα τερμάτιζε.

Η Prolog είναι μία πολύ δυνατή και γενικού σκοπού γλώσσα. Δεν υπάρχει κάτι που δεν μπορεί να γραφεί σε γλώσσα Prolog. Ωστόσο λόγω της ιδιαιτερότητάς της δεν είναι εξαιρετικά δημοφιλής πέρα από κάποιες εφαρμογές, όπως είναι για παράδειγμα εφαρμογές ανάλυσης γλώσσας (language parsing). Αξίζει να παρατηρήσει κανείς πόσο καλά δομημένη είναι, πράγμα που συμβαίνει κυρίως επειδή έχει τις βάσεις της στην Πρωτοβάθμια Λογική (First Order Logic).

- Schema.org – τα αντικείμενα που περιγράφονται προς πώληση στο σενάριο έχουν δημιουργηθεί με βάση την αντίστοιχη περιγραφή του Schema.org (παρακάτω θα δούμε και ένα παραδείγμα). Αυτό έγινε διότι ήταν αδύνατο να δημιουργήσουμε έναν γενικευμένο πράκτορα χωρίς μία σαφή και προκαθορισμένη μορφή των αντικειμένων.
- JSON-LD – το κάθε αντικείμενο που περιγράφεται από το Schema.org ο client το λαμβάνει υπό την μορφή JSON-LD.

6.1.2 Υλοποίηση

Ο agent έχει γραφτεί σε Prolog και ο κώδικας είναι διαθέσιμος στο ακόλουθο public github repository για όποιον θέλει να το επισκεφτεί[3]: https://github.com/petyhaker/agent .

Ας ξεκινήσουμε αναλύοντας το τι κάνει ο agent. Ο agent που φτιάξαμε αυτή τη στιγμή μπορεί να εκτελέσει μία από τις ακόλουθες λειτουργίες:

- buyProduct – να αγοράσει δηλαδή ένα προϊόν από κάποιο online μαγαζί
- browseProduct – να παρουσιάζει ένα προϊόν από κάποιο online μαγαζί
- checkoutBasket – να αδειάσει το καλάθι, δηλαδή να αγοράσει ό,τι υπάρχει στο καλάθι ενός ηλεκτρονικού μαζαγιού
- browseBasket – να δείξει στον χρήστη το καλάθι του σε κάποιο online μαγαζί

Οι λειτουργίες αυτές δεν είναι ενσωματωμένες στον agent, καθώς όπως είπαμε θέλουμε ο πράκτορας να είναι γενικευμένος και να είναι σε θέση να εκτελεί ότι χάρτη λαμβάνει από τον εκάστοτε server, στην δική μας περίπτωση τον χάρτη από το ηλεκτρονικό κατάστημα. Επομένως ο agent μας το μόνο που ξέρει να κάνει είναι να διασχίζει το γράφο που λαμβάνει από το ηλεκτρονικό κατάστημα και να φέρνει σε πέρας τα πράγματα που του έχουν ανατεθεί ακολουθώντας αναλυτικές οδηγίες.

Πάμε να δούμε βήμα προς βήμα τι συμβαίνει. Αρχικά η κλήση του προγράμματος γίνεται ως εξής:

```
agent(Affordance, Scope, Request_Parameters, Request_Parameters_Values).
```

Το Affordance είναι αυτό που καθορίζει τον στόχο του agent. Το Scope είναι τα URLs των καταστημάτων που θα πρέπει να στείλει αίτημα και μπορεί να είναι ένα ή περισσότερα ή και all. Στην περίπτωση που η συνάρτηση κληθεί με Scope==all ο agent ψάχνει στο αρχείο του trusted_stores.txt[4] και τα βάζει όλα σε μία λίστα για να τα εξετάσει ένα προς ένα.

Τα Request_Parameters και Request_Parameters_Values είναι προφανώς οι παράμετροι και οι αντίστοιχες τιμές τους που θα συνοδεύουν το αίτημα στο προαναφερθέν URL. Ένα παράδειγμα κλήσης π.χ. θα ήταν το ακόλουθο:

```
agent(browseBasket, ["http://www.amazon.com/api/", end_of_file],
["username", "password"], ["vasiliki", "verysecurepassword"]).
```

[3] Οδηγίες χρήσης βρίσκονται αναλυτικά στο README.md file

[4] Είναι απλό .txt αρχείο καθώς δεν υπάρχει κάποια πληροφορία με συγκεκριμένη δομή (π.χ. πεδία, ταμπέλες, κλπ) που να επιβάλει την χρήση πιο περίπλοκης μορφής. Θα μπορούσε εξίσου εύκολα να είχε γραφεί σε κάποιο .pl ή .json αρχείο.

Σε αυτή την περίπτωση ζητάμε από τον agent να δείξει το καλάθι του χρήστη στο ηλεκτρονικό κατάστημα Amazon και παίρνει σαν παραμέτρους το username και το password.

Το πρώτο πράγμα που κάνει ο agent είναι να καλέσει την agent_inner, μία εσωτερική συνάρτηση που είναι υπεύθυνη για 3 πράγματα. Αρχικά να πάρει τον γράφο του καταστήματος, δεύτερον να εξετάσει αν το Affordance που έχει οριστεί έχει νόημα στο παρόν κατάστημα και τρίτον να δει αν υπάρχει μονοπάτι που μπορεί να ακολουθήσει για να επιτύχει τον στόχο. Το αίτημα στον server γίνεται με την συνάρτηση getGraph(X), όπου στην παρούσα υλοποίηση αντιστοιχεί απλά σε φόρτωση αρχείου. Στον πραγματικό κόσμο αυτό θα αντιστοιχούσε με ένα HTTP request στο URL http://www.amazon/api. Όσο για την δεύτερη λειτουργία της συνάρτησης, ουσιαστικά διαβάζει τον χάρτη που λαμβάνει και εξετάζει βιαστικά αν υπάρχει κάπου το ζητούμενο Affordance δηλωμένο. Για παράδειγμα, αν ζητήσεις buyProduct είναι έγκυρο για το κατάστημα Amazon εφόσον αναφέρεται ρητά στον χάρτη του καταστήματος, αλλά αν ζητήσεις makeCookies δεν είναι έγκυρο. Παρακάτω ακολουθεί ένα παράδειγμα του χάρτη του καταστήματος amazon που σχεδιάσαμε και αφορά τα valid affordances:

```
valid_intension(buyProduct).
valid_intension(browseProduct).
valid_intension(browseBasket).
valid_intension(checkoutBasket).
```

Ο γράφος κάθε καταστήματος είναι ένα .pl αρχείο, οπότε είναι επίσης γραμμένος υπό την μορφή Prolog κανόνων και γεγονότων.

Αν όλα τα παραπάνω έχουν γίνει με επιτυχία, το τελευταίο πράγμα που εξετάζει η συνάρτηση είναι αν υπάρχει έγκυρο μονοπάτι που μπορεί ο agent να ακολουθήσει για να ολοκληρώσει τον στόχο του. Αυτό ελέγχεται στην traverse_verification συνάρτηση. Η συνάρτηση αυτή έχει εξαιρετικό ενδιαφέρον, κυρίως γιατί είναι απόλυτα γενικευμένη (agnostic) ως προς το ηλεκτρονικό κατάστημα αλλά και τον τελικό στόχο.

```
traverse_verification(_, _, end).
traverse_verification(Intension, Request, State) :- path(Intension, State, Request, _, _),next_state(Intension, State, Request, NextState), path(Intension, NextState, NextRequest, _, _), traverse_verification(Intension, NextRequest, NextState), !.
```

Η συνάρτηση αυτή καλείται ως εξής traverse_verification(Affordance, Request, State), όπου State αντιστοιχεί σε ένα node του γράφου. Όταν καλείται επομένως από την agent_inner καλείται με entry στην θέση του State, ένα αναγνωριστικό που υποδηλώνει την έναρξη του συγκεκριμένου στόχου. Αντίστοιχο αναγνωριστικό υπάρχει και για την λήψη, το end. Η

επιλογή τον αναγνωριστικών είναι αποκλειστικά συμβάσεις (θα μπορούσε π.χ. να είναι start αντί για entry) αλλά για πρακτικούς λόγους η χρήση τους είναι απαραίτητη και η καθολική τους χρήση επιβεβλημένη (δεν μπορεί π.χ. ένας χάρτης να χρησιμοποιεί το start και ένας άλλος το entry). Η συνάρτηση αυτή τερματίζει επιτυχημένα όταν από ένα entry point φτάσει μέσα από ένα μονοπάτι σε ένα end point (το γεγονός traverse_verification(_,_,end) σημαίνει ότι η συνάρτηση επιστρέφει true μόνο όταν φτάσει σε end point). Είναι σημαντικό να παρατηρήσουμε ότι στην συνάρτηση αυτή δεν εκτελείται κανένα request, δεν αναλύεται δηλαδή αν τα βήματα αυτά μπορούν να εκτελεστούν με επιτυχία. Το μόνο που κάνει είναι να ελέγχει αν υπάρχει πλήρες μονοπάτι που μπορεί να ικανοποιήσει αυτό το στόχο (affordance) μέσα από έναν πεπερασμένο αριθμό κόμβων στο γράφο. Αυτός ο έλεγχος γίνεται προληπτικά, για να μην εμφανιστεί στην συνέχεια (π.χ. μετά από 10 βήματα, 15 requests και συνολικά 2 λεπτά χρόνο) ότι ο γράφος είναι σπασμένος και δεν μπορεί να προχωρήσει η διαδικασία.

Ο γράφος για το browseProduct που αναφέραμε σαν παράδειγμα στην αρχή στο amazon online store είναι ο ακόλουθος:

```
next_state(browseProduct, entry,"http://www.amazon.com/api/", search).
next_state(browseProduct, search,"http://www.amazon.com/api/search",
product).
next_state(browseProduct, product,"http://www.amazon.com/api/product",
end).
```

Το συγκεκριμένο affordance για να υλοποιηθεί επομένως χρειάζεται 3 βήματα, εκ των οποίων το πρώτο είναι το entry point και το τελευταίο το end point. Το ενδιάμεσο βήμα είναι το product. Σε κάθε στάδιο όπως θα δούμε στην συνέχεια εκτελείται και κάτι διαφορετικό από τον agent - αλλά στην παρούσα συνάρτηση δεν είναι αυτός ο σκοπός – για αυτό και παρέχονται τα κατάλληλα URLs.

Για διαφορετικά affordances στον ίδιο γράφο μπορεί τα στάδια να είναι λιγότερα ή περισσότερα, ανάλογα την ανάγκη:

```
next_state(buyProduct, entry,"http://www.amazon.com/api/", search).
next_state(buyProduct, search,"http://www.amazon.com/api/search", product).
next_state(buyProduct, product,"http://www.amazon.com/api/product",
basket).
next_state(buyProduct, basket,"http://www.amazon.com/api/basket",
checkout).
next_state(buyProduct, checkout,"http://www.amazon.com/api/checkout",
payment).
next_state(buyProduct, payment,"http://www.amazon.com/api/payment", end).
```

Το παραπάνω παράδειγμα δείχνει τον αντίστοιχο γράφο για το affordance buyProduct. Είναι φανερό ότι έχει πολύ περισσότερα ενδιάμεσα στάδια (περνάει και από το στάδιο product), αλλά σε γενικές γραμμές ακολουθεί την ίδια λογική (το κατηγόρημα next_state/3 είναι το ίδιο και επίσης κάνει χρήση των entry/end).

Σε άλλα καταστήματα το ίδιο affordance μπορεί να έχει διαφορετικά ενδιάμεσα βήματα (περισσότερα ή λιγότερα). Το DeepGraphs specification όπως το έχουμε φανταστεί δεν περιορίζει την εκάστοτε ιστοσελίδα να προσαρμόσει την λειτουργικότητα και την δομή της σε κάτι καινούργιο. Απλώς δίνει έναν δομημένο τρόπο να περιγράψουν τον τρόπο που ήδη λειτουργούν. Επομένως μπορεί τα βήματα και τα affordances μιάς ιστοσελίδας (ή ενός ηλεκτρονικού καταστήματος) να είναι τελείως διαφορετικά αλλά ο χάρτης θα πρέπει να μοιάζει με αυτόν που περιγράψαμε ήδη. Για παράδειγμα ο παραπάνω agent μπορεί να διασχίσει τον ακόλουθο χάρτη για το affordance buyProduct ενός άλλου μαγαζιού:

```
next_state(buyProduct, entry,"http://www.amazon.com/api/", search).
next_state(buyProduct, search,"http://www.amazon.com/api/search", product).
next_state(buyProduct, product,"http://www.amazon.com/api/product",
basket).
next_state(buyProduct, basket,"http://www.amazon.com/api/basket",
checkout).
next_state(buyProduct, checkout,"http://www.amazon.com/api/checkout",
payment).
next_state(buyProduct, payment,"http://www.amazon.com/api/payment",
postage).
next_state(buyProduct, postage,"http://www.amazon.com/api/postage", end).
```

Στον καινούργιο αυτό χάρτη υπάρχει ένα νέο βήμα, που αφορά τα στοιχεία αποστολής. Στην περίπτωση αυτή ο agent μας μπορεί και πάλι να διασχίσει τον γράφο χωρίς κανένα πρόβλημα.

Εφόσον ο agent βεβαιωθεί ότι ο σκοπός που του έχει ανατεθεί συνάδει με το ηλεκτρονικό κατάστημα και μπορεί να επιτευχθεί, καλεί την επόμενη συνάρτηση που είναι η traverse. Η συνάρτηση αυτή καλείται ως εξής traverse(Affordance, Stores, Parameter, Value) και στόχος της είναι πλέον να εκτελέσει τα διάφορα στάδια. Η συνάρτηση αυτή παίρνει το URL που είναι στην αρχή της λίστας Stores και προετοιμάζει ότι χρειάζεται για να εκτελέσει τον στόχο. Για παράδειγμα αν το buyProduct έχει scope all, η traverse είναι υπεύθυνη να ξεχωρίσει ποια καταστήματα έχουν το προϊόν που ζητήθηκε από τον χρήστη, ποιο έχει την καλύτερη τιμή και αφού καταλήξει στο 1 κατάστημα που θα συνεχίσει να καλέσει την complete_action που είναι και η συνάρτηση που είναι υπεύθυνη να κάνει τα απαραίτητα http requests στον server για να αγοράσει το προϊόν.

Στο σημείο αυτό μπορεί κανένας να παρατηρήσει ότι η traverse δεν είναι τόσο agnostic όπως ήταν η traverse_verification. Αυτό συμβαίνει γιατί όντως κάποια πράγματα ο agent πρέπει να τα κοιτάζει ανά περίπτωση (π.χ. άλλο φιλτράρισμα απαιτείται στο browseBasket και άλλη στο browseProduct). Στην συγκεκριμένη υλοποίηση κάποιες πολύ βασικές λειτουργίες τις εντάξαμε στα πλαίσια του γενικευμένου πράκτορα, θεωρώντας ότι δεν υπάρχει ηλεκτρονικό κατάστημα χωρίς τα βασικά affordances που είδαμε. Αν κάποιο κατάστημα θέλει να προσθέσει επιπλέον, τότε μπορεί πολύ απλά να προσθέσει επιπλέον κανόνες στον γράφο του και ο πράκτορας θα μπορέσει να τα εκτελέσει. Ίσως πάλι στα πλαίσια του DeepGraphs να ήταν καλύτερα ο agent να ήταν εντελώς γενικευμένος και άρα να μετακινήσουμε οποιαδήποτε εξειδικευμένη πληροφορία στον χάρτη. Οποιαδήποτε από τις δύο προσεγγίσεις είναι αποδεκτή και συμβατή με το DeepGraphs. Εμείς επιλέξαμε την πρώτη προσέγγιση στην δική μας υλοποίηση. Αυτό που αξίζει να αναφέρουμε ωστόσο είναι ότι στην traverse γίνεται αυτή τη στιγμή το «φιλτράρισμα» των μαγαζιών και η τελική επιλογή στην περίπτωση που ο χρήστης δεν έχει δηλώσει σαφώς το scope (π.χ. έχει βάλει την επιλογή all). Άρα ένα σημαντικό κομμάτι του reasoning του agent γίνεται εδώ. Στο δικό μας παράδειγμα η επιλογή γίνεται αυστηρά κάνοντας sort στις τιμών των προϊόντων (τα προϊόντα μπορεί να είναι είτε από το ίδιο είτε από διαφορετικά καταστήματα), αλλά θα μπορούσαμε να έχουμε υλοποιήσει και άλλα κριτήρια ορισμένα από τον χρήστη.

Οι δύο τελευταίες συναρτήσεις για τις οποίες θα μιλήσουμε είναι η complete_actions και η collect_prices. Είναι οι δύο μόνες συναρτήσεις που επικοινωνούν με τον server και αναλαμβάνουν το πιο εκτελεστικό κομμάτι της διαδικασίας. Η collect prices είναι υπεύθυνη να ζητήσει προϊόντα και συγκεκριμένα τις τιμές αυτών από όλα τα stores που έχουν οριστεί από τον χρήστη. Η συνάρτηση αυτή δεν κάνει κάτι εξαιρετικό, απλά συλλέγει τις τιμές και τις επιστρέφει σε λίστα στην traverse. Αυτό που αξίζει να σημειωθεί είναι ότι η collect_prices για να αποσπάσει την τιμή από το προϊόν κάνει χρήση του Schema.org. Όταν ζητάει από τον server το προϊόν (η search στο παράδειγμά μας είναι μία απλή φόρτωση αρχείου, και όχι ένα πραγματικό http request) αυτό που λαμβάνει είναι ένα JSON αρχείο με όλα τα προϊόντα που είναι διαθέσιμο. Αν το προϊόν δεν είναι στην καθορισμένη Schema.org μορφή ο agent δεν θα μπορούσε ποτέ να πάρει την πληροφορία που χρειάζεται. Ένα παράδειγμα προϊόντος που έχουμε βάλει στην βάση δεδομένων του καταστήματος Amazon μας είναι το ακόλουθο, το βιβλίο (υποκατηγορία του Thing) Moby Dick:

```
{
    "@context": "http://schema.org",
    "@type": "Product",
    "productId": "foo",
    "description": " lsgoiesg",
```

```
    "image": "foo",
    "name":"Moby Dick",
    "offers": {
      "@type": "Offer",
      "availability": "yes",
      "price": "55.00",
      "priceCurrency": "USD"
    },
    "review": [
      {
      "@type": "Review",
      "author": "Ellie",
      "datePublished": "2011-04-01",
      "description": "Great story but the translation was poor.",
      "name": "Not a happy camper",
      "reviewRating": {
        "@type": "Rating",
        "bestRating": "5",
        "ratingValue": "1",
        "worstRating": "1"
        }
     }
   ]
}
```

Να προσθέσουμε ακόμα ότι η collect_prices εξετάζει και την διαθεσιμότητα του βιβλίου, με αντίστοιχο τρόπο όπως και την τιμή, κάνοντας πάντα χρήση του Schema.org προτύπου.

Τέλος η complete_action είναι αυτή που εκτελεί όλα τα κατάλληλα requests για να επιτύχει τον επιθυμητό στόχο. Φτιάχνει πρώτα τις κατάλληλες παραμέτρους με την fix_parameters και καλεί την http_request για να πάρει το αποτέλεσμα της κλήσης, να επιβεβαιώσει ότι το αίτημα ήταν επιτυχές και να προχωρήσει στα επόμενα στάδια. Να τονίσουμε εδώ ότι τόσο τα στάδια όσο και οι παράμετροι που πρέπει να οριστούν για να γίνει ένα έγκυρο αίτημα στον server διαβάζονται αποκλειστικά και μόνο από τον χάρτη. Ένα παράδειγμα του πως υποδεικνύονται τα σωστά αιτήματα φαίνεται παρακάτω, πάλι ένα απόσπασμα από τον χάρτη του Amazon store:

```
path(buyProduct, basket, "http://www.amazon.com/api/basket",post,
["usename", "password"]).
path(browseBasket, basket, "http://www.amazon.com/api/basket",get,
["usename", "password"]).
path(checkoutBasket, basket, "http://www.amazon.com/api/basket",get,
["usename", "password"]).
path(_ , search, "http://www.amazon.com/api/search",get, ["name"]).
path(_, product, "http://www.amazon.com/api/product",get, ["productId"]).
path(_, checkout, "http://www.amazon.com/api/checkout",get, ["basketId"]).
path(_, payment, "http://www.amazon.com/api/payment",post,
["creditCard_number", "postage_info"]).
path(_, entry, "http://www.amazon.com/api/", get, "").
path(_, end, "http://www.amazon.com/api/", get, "").
```

6.1.3 Αποτελέσματα

Παρακάτω ακολουθούν κάποια παραδείγματα εκτέλεσης του προγράμματος, ενδεικτικά των περισσότερων σεναρίων που έχουμε καλύψει με τον agent που φτιάξαμε.

6.1.3.1 Αίτημα για αγορά προϊόντος από όλα τα διαθέσιμα καταστήματα και από ένα κατάστημα.

Το βιβλίο “Moby Dick” είναι διαθέσιμο μόνο στην βάση δεδομένων του Amazon store – για αυτό και στο κατάστημα της Beneton δείχνει μη διαθέσιμο. Το αποτέλεσμα δίνει ένα μόνο προϊόν, εκείνο με την καλύτερη τιμή. Το output δίνει το ProductID (στην περίπτωσή μας “foo”) την τιμή και το URL από όπου το αγοράσαμε.

```
17 ?- agent(buyProduct, all, "name", "Moby Dick").

foo 55.0 http://www.amazon.com/api/
true
```

Εικόνα 4 Buy product named "Moby Dick" from all available stores

```
 26 ?- agent(buyProduct, ["http://www.amazon.com/api/", end_of_file], "n
ame", "Moby Dick").
foo 55.0 http://www.amazon.com/api/
true
```

Εικόνα 5 Buy product named "Moby Dick" from Amazon store

```
27 ?- agent(buyProduct, ["http://www.beneton.com/", end_of_file], "name"
, "Moby Dick").
No items match your search
true
```

Εικόνα 6 Buy product named "Moby Dick" from Beneton store

Το αποτέλεσμα που προκύπτει για το “Foo Book” από εδώ και κάτι, είναι ότι δεν υπάρχουν διαθέσιμα προϊόντα με αυτό το όνομα. Όντως τέτοιο βιβλίο δεν υπάρχει σε καμία βάση κανενός καταστήματος.

```
 18 ?- agent(buyProduct, all, "name", "Foo Book").

No items match your search
true
```

Εικόνα 7 Buy Product named "Foo Book" from all stores

6.1.3.2 *Αίτημα για εξέταση προϊόντος από όλα τα διαθέσιμα καταστήματα και από ένα κατάστημα.*

Όπως ήδη είπαμε, το βιβλίο “Moby Dick” είναι διαθέσιμο μόνο στην βάση δεδομένων του Amazon store. Το αποτέλεσμα δίνει ένα μόνο προϊόν, εκείνο με την καλύτερη τιμή. Το output δίνει το ProductID (στην περίπτωσή μας “foo”) την τιμή και το URL από όπου το αγοράσαμε.

```
25 ?- agent(browseProduct, all, "name", "Moby Dick").

foo 55.0 http://www.amazon.com/api/
true
```

Εικόνα 8 Browse product named "Moby Dick" from all stores

```
 24 ?- agent(browseProduct, all, "name", "Foo Book").

No items match your search
true
```

Εικόνα 9 Browse product named "Foo Book" from all stores

```
 23 ?- agent(browseProduct, ["http://www.amazon.com/api/", end_of_file],
 "name", "Foo Book").
No items match your search
true
```

Εικόνα 10 Browse product named "Foo Book" from Amazon store

```
 29 ?- agent(browseProduct, ["http://www.amazon.com/api/", end_of_file]
 "name", "Moby Dick").
foo 55.0 http://www.amazon.com/api/
true
```

Εικόνα 11 Browse product named "Moby Dick" from Amazon store

6.1.3.3 Αίτημα για εξέταση καλαθιού από ένα κατάστημα

Το αίτημα αυτό το έχουμε περιορίσει σε ένα κατάστημα (το scope δεν μπορεί να είναι πάνω από ένα κατάστημα την φορά). Το αποτέλεσμα δίνει ένα απλό true, που σημαίνει ότι η εκτέλεση έληξε επιτυχώς και το καλάθι επιστρέφει στον χρήστη. Δυστυχώς στα πλαίσια της υλοποίησης δεν ασχοληθήκαμε με περισσότερες λεπτομέρειες που αφορούν τον χρήστη, όπως στοιχεία χρήστη, καλάθι, κλπ. Περισσότερες λεπτομέρειες αναλύονται στην επόμενη υποενότητα των Επεκτάσεων.

```
 19 ?- agent(browseBasket, ["http://www.amazon.com/api/", end_of_file],
["username", "password"], ["vasiliki", "verysecurepassword"]).
true
```

Εικόνα 12 Browse basket from Amazon store

6.1.3.4 Αίτημα για ολοκλήρωση αγοράς καλαθιού σε ένα κατάστημα

Όπως ακριβώς και πριν το checkoutBasket έχει περιοριστεί σε ένα μόνο κατάστημα.

```
22 ?- agent(checkoutBasket, ["http://www.amazon.com/api/", end_of_file],
 ["username", "password"], ["vasiliki", "verysecurepassword"]).
true
```

Εικόνα 13 Checkout basket from amazon store

Σε όλα τα παραπάνω παραδείγματα που είδαμε υποθέσαμε ότι όλα τα http requests επέστρεφαν με status code 200, που σημαίνει επιτυχία. Στο παράδειγμα που ακολουθεί θέσαμε το status code του checkout σε 500 (υποδηλώνει πρόβλημα στην μεριά του server) και το αποτέλεσμα είναι το ακόλουθο.

```
 37 ?- agent(browseProduct, ["http://www.amazon.com/api/", end_of_file],
 "name", "Moby Dick").
foo 55.0 http://www.amazon.com/api/
false.
```

Πράγματι, ο agent βρήκε το προϊόν, αλλά απέτυχε η όλη διαδικασία επειδή κάποιο http request δεν επέστρεψε σωστά.

6.1.4 Περιορισμοί και επεκτάσεις

Ο agent όπως είδαμε μπορεί να εκτελέσει έναν αριθμό από λειτουργίες και αποκρίνεται με αρκετά καλό τρόπο και σε σενάρια που δεν έχουν προβλεφθεί εξ αρχής. Ωστόσο, εξακολουθεί να παραμένει ένα περιορισμένο εγχείρημα με αρκετές δυνατότητες επέκτασης και βελτίωσης. Στην ενότητα αυτή ακολουθούν κάποιες ιδέες που θα μπορούσαν να αναβαθμίσουν τεχνικά το παρόν project.

Μία πρώτη και απαραίτητη αρχή είναι να δημιουργήσουμε ένα wrap application σε python το οποίο θα καλεί το Prolog πρόγραμμα σαν εξωτερικό reasoner. Ο λόγος που θέλουμε να γίνει αυτό, είναι κυρίως γιατί απώτερος στόχος είναι ο agent να επικοινωνεί με φυσική γλώσσα στον χρήστη. Αφενός η Python είναι μία γλώσσα με πολλές δυνατότητες σε αυτόν τον τομέα, αφετέρου είναι φανερό ότι δεν δόθηκε ιδιαίτερη σημασία στην αλληλεπίδραση με τον χρήστη σε αυτή την φάση ανάπτυξης του agent και τόσο τα inputs όσο και τα outputs είναι δύσχρηστα. Αυτό θα ήταν μία σημαντική ποιοτική αναβάθμιση του εγχειρήματος.

Σε δεύτερη φάση θα ήταν ωφέλιμο για την πληρότητα του σεναρίου να υλοποιήσουμε έναν server που να δέχεται και να απαντάει στα http_requests. Όπως ήδη αναφέραμε μέχρι τώρα τα http requests είτε τα προσομοιώναμε με φόρτωση αρχείου (π.χ. στο αίτημα για τον χάρτη) είτε με ένα απλό γεγονός που επιβεβαίωνε ότι το αίτημα έγινε και επιστρέφει 200. Ήδη έγινε μία προσπάθεια για υλοποίηση ενός server σε Node.js, η οποία όμως δεν προχώρησε καθώς στην πορεία θεωρήθηκε ότι εισήγαγε επιπλέον κόπο και πολυπλοκότητα στο σύστημα που ήταν άσχετη με την πραγματική αξία της προσπάθειας που υλοποιούμε. Σε επόμενα στάδια θα μπορούσαμε να το κάνουμε να επικοινωνεί με την εξωτερική Python εφαρμογή, αντί για τον Prolog agent κατευθείαν.

Τέλος, σε μελλοντικά βήματα θα πρέπει όλη η επικοινωνία να γίνεται με JSON αρχεία (ιδανικά να προέρχονται από databases) και να καταργηθούν όλα τα .txt και .pl αρχεία.

6.2 Υλοποίηση διαπραγμάτευσης στο DeepGraphs specification

6.2.1 Τεχνολογίες που χρησιμοποιήθηκαν

Η υλοποίηση του agent που παρουσιάστηκε στην προηγούμενη ενότητα επεκτάθηκε σε αυτή την ενότητα έτσι ώστε να περιλαμβάνει και την ικανότητα να διαπραγματεύεται τιμές. Επομένως οι τεχνολογίες που περιγράφονται στην υποενότητα 6.1 αφορούν και αυτό το κομμάτι της υλοποίησης.

6.2.2 Υλοποίηση

Όπως περιγράψαμε ήδη και έχουμε πει σε πολλά σημεία της εργασίας ο πράκτορας είναι γενικευμένος επομένως μπορεί να διαβάσει έναν χάρτη λίγο διαφορετικό από τον προηγούμενο. Στην προκειμένη περίπτωση αλλάξαμε τον χάρτη του θεωρητικού ηλεκτρονικού καταστήματος Foo Store ως εξής:

```
next_state(buyProduct, entry, "http://www.foo.com/api/", search).
next_state(buyProduct, search, "http://www.foo.com/api/search", product).
next_state(buyProduct,product, "http://www.foo.com/api/product",
negotiate).
next_state(buyProduct,negotiate,"http://www.foo.com/api/negotiate",
basket).
next_state(buyProduct, basket, "http://www.foo.com/api/basket", checkout).
next_state(buyProduct,checkout, "http://www.foo.com/api/checkout",
payment).
next_state(buyProduct, payment, "http://www.foo.com/api/payment", end).
```

Η αλλαγή αφορά αποκλειστικά το Affordance buyProduct. Οποιαδήποτε άλλη λειτουργία δεν αλλάζει ακόμα και αν η τιμή είναι διαπραγματεύσιμη. Αξίζει επίσης να παρατηρήσουμε ότι η traverse_verification μπορεί χωρίς πρόβλημα να διαβάσει τον παραπάνω γράφο.

Το μόνο πράγμα που προσθέσαμε επομένως στην παραπάνω υλοποίηση είναι μία παραπάνω συνάρτηση που καλείται μόνο όταν στον γράφο μέσα υπάρχει το στάδιο “negotiate”. Τότε η traverse καλεί την αντίστοιχη συνάρτηση (negotiate) που επεξεργάζεται το προϊόν και τις τιμές που δίνονται από τον αγοραστή και τον πωλητή, υπολογίζει την προσφορά που θα δώσει και περιμένει απάντηση. Αν η απάντηση είναι ναι από τον προμηθευτή τότε προχωράει χωρίς πρόβλημα στα επόμενα βήματα, αλλιώς ξαναπροσπαθεί υπολογίζοντας εκ νέου προσφορά. Σε περίπτωση που η ελάχιστη τιμή είναι μεγαλύτερη από την μέγιστη τιμή η διαπραγμάτευση καταρέει. (η συνάρτηση επιστρέφει false).

Αξίζει να δώσουμε λίγη προσοχή στον υπολογισμό της προσφοράς που ακολουθεί το μοντέλο που περιγράψαμε στο αντίστοιχο κεφάλαιο της μεθοδολογίας:

```
calc_offer(MinPrice, DeltaP, Result) :- getUser_Delta(DeltaB),
getUser_Price(MaxPrice), Result1 is (MaxPrice - MinPrice), Result2 is ( (1
- DeltaP) / (1 - (DeltaP * DeltaB)) ),Result is (Result1*Result2)+MinPrice.
```

Η παραπάνω συνάρτηση πιστή στην μεθοδολογία που περιγράψαμε παίρνει σαν είσοδο την μικρότερη τιμή που θεωρεί αποδεκτή ο πωλητής και την μέγιστη τιμή που προτίθεται να πληρώσει ο αγοραστής, καθώς και τα Δελτα και των δύο πλευρών για να υπολογίσει μία προσφορά με χρονικό βάθος 2. Η επιλογή 2 είναι τελείως αυθαίρετη, αλλά την κρατήσαμε

σταθερή για να απλοποιήσει την υλοποίηση. Η προσφορά που κάνει τώρα ο αγοραστής είναι φτιαγμάνη με τέτοιον τρόπο που είναι βέλτιστη για και για τους δύο, επομένως αν δεν συντρέχουν άλλοι λόγοι (π.χ. τεχνικά προβλήματα) ο πωλητής θα πρέπει να την αποδεχτεί σαν βέλτιστη. Όπως ήδη αναφέραμε και στην περιγραφή της μεθοδολογίας το μοντέλο αυτό ευνοεί τον αγοραστή, καθώς του δίνει το πρώτο βήμα.

6.2.3 Αποτελέσματα

Όπως είδαμε το μόνο από τα καταστήματα που έχει το στάδιο negotiate είναι το Foo store, επομένως σε αυτό βάλαμε ένα Moby Dick βιβλίο και τώρα αν ζητήσουμε από τον agent να μας αγοράσει το βιβλίο αυτό από το κατάστημα Foo η τιμή που υπολογίζει, προτείνει και «κλείνει» την συμφωνία φαίνεται παρακάτω.

```
 41 ?- agent(buyProduct, ["http://www.foo.com/api/", end_of_file], "name", "Moby Dick").
foo 56.666666666666664
true
```

Στο προηγούμενο παράδειγμα οι τιμές είχαν οριστεί σαν:

- Ελάχιστη τιμή (πωλητή): 50
- Μέγιστη τιμή (αγοραστή): 60
- Δελτα πωλητή: 0.5
- Δέλτα αγοραστή: 0.5

Όσο για τις υπόλοιπες λειτουρίες δεν αλλάζουν κάπως στο πρόγραμμα, εξακολουθούν να λειτουργούν όπως ακριβώς και πριν. Η μόνη διαφορά που παρατηρείται που οφείλεται σε δική μας κακή σχεδίαση του προϊόντος που προσθέσαμε στο Foo Store είναι το ακόλουθο:

> Δεν προβλέψαμε να βάλουμε την προτεινόμενη τιμή του αγοραστή σε καινούργιο πεδίο στο JSON αρχείο, αλλά γεμίσαμε το πεδίο τιμή (price) του προϊόντος. Με τον τρόπο αυτό όταν έχουμε παραπάνω από ένα καταστήματα να ελέγξουμε τιμή πλέον πάνω πάνω βγαίνει το προϊόν από το Foo store γιατί η ελάχιστη τιμή του πωλητή είναι 50 (ενώ στο Amazon Store είναι 55)
>
> ```
>  86 ?- agent(browseProduct, all, "name", "Moby Dick").
>
> foo 50.0 http://www.foo.com/api/
> true
> ```
>
> Στην πραγματικότητα όμως ο αγοραστής θα αναγκαστεί να πληρώσει τελικά 56.7 όπως είδαμε πριν, πράγμα που είναι παραπλανητικό και χρήζει διόρθωσης.

6.2.4 Περιορισμοί και επεκτάσεις

Όπως ήδη έχουμε αναφέρει, η υλοποίηση αυτή είναι μία αρχική προσπάθεια και επομένως υπάρχει αρκετός χώρος για βελτίωση. Τεχνικά περιγράψαμε κάποιες ιδέες στην ενότητα 6.1.4 που μπορούν να εφαρμοστούν και να υλοποιηθούν άμεσα. Σε ότι αφορά το αμιγώς κομμάτι της διαπραγματευτικής ικανότητας του πράκτορα υπάρχουν ακόμα κάποιες προβληματικές, όπως το browseProduct που αναφέραμε παραπάνω. Αυτό θα μπορούσε να λυθεί επεκτείνοντας το Shema.org που χρησιμοποιήσαμε κάνοντας σαφές ότι η τιμή αυτή είναι εικονική.

7

Επίλογος

Φτάνοντας στο τέλος της εργασίας, μπορούμε να κάνουμε μία ανασκόπηση της δουλειάς που έγινε. Αρχικά, μιλήσαμε για την διαπραγμάτευση σαν φαινόμενο και σαν διαδικασία αρκετά αναλυτικά. Στη συνέχεια, μελετήσαμε μία προσπάθεια για επέκταση του Hydra specification, το DeepGraphs specification. Το πρότυπο αυτό προτείνει τον συνδυασμό τεχνολογιών, όπως το SWRL και το Shema.org, με το Hydra specification έτσι ώστε οι υπηρεσίες να μπορούν να κινούνται αυτόνομα πλέον στον web. Αφού αναλύσαμε τεχνικά και αξιολογήσαμε την πρόταση, προσπαθήσαμε να επεκτείνουμε το DeepGraphs πρότυπο έτσι ώστε να επιτρέπει την διαπραγμάτευση σαν μία ακόμα διαδικασία που οι πράκτορες θα μπορούν να εκτελούν φυσικά (όπως και όλες τις υπόλοιπες). Στα πλαίσια αυτής της θεωρητικής μελέτης, αναπτύχθηκε και μία μεθοδολογία, γενικά για το DeepGraphs αλλά και για το πως θα εντασσόταν η διαπραγμάτευση μέσα σε αυτό.

Η εργασία αυτή συνοδεύεται και από μία υλοποίηση της προαναφερθείσας μεθοδολογίας. Αρκετά κομμάτια της προτεινόμενης μεθοδολογίας παραλείπονται σκόπιμα από την υλοποίηση, καθώς θεωρήθηκε ότι δεν προσέφεραν κάτι παραπάνω στην προσπάθεια. Σε κάθε περίπτωση όμως ήταν μία αρχική προσέγγιση του DeepGraphs specification και την διαπραγμάτευσης μέσα σε αυτό, με αρκετά ελπιδοφόρα αποτελέσματα.

Τόσο η θεωρητική μελέτη όσο και η υλοποίηση, είναι διαθέσιμα στην κοινότητα ανοιχτού λογισμικού. Ελπίζουμε τα πρώτα βήματα που κάναμε να εμπνεύσουν και να βοηθήσουν ερευνητές προς αυτή την κατεύθυνση. Ιδανικά πατώντας πάνω σε αυτά που ήδη αναλύσαμε,

θα μπορούσαν να επεκτείνουν την ιδέα μας με ακόμα πιο πολλές και ευρείες λειτουργίες - και την αντίστοιχη υλοποίηση επίσης. Για παράδειγμα θα μπορούσε κάποιος να σκεφτεί λύσεις ώστε να μπορούν οι πράκτορες να είναι ακόμα πιο ευέλικτοι σε ότι αφορά την διαπραγμάτευση, όπως το να τους επιτρέπει να διαλέγουν δυναμικά στρατηγική ανάλογα την περίπτωση. Θα μπορούσε ακόμα να τους επιτρέπει να χρησιμοποιούν επιχειρήματα, ή να προτείνει έναν αποτελεσματικό τρόπο για πολύπλευρη διαπραγμάτευση. Συμβουλές και προτάσεις θα μπορούσαν γίνουν και στο κομμάτι του DeepGraphs αμιγώς, είτε επεκτείνοντας την λειτουργία του είτε προτείνοντας νέες προσεγγίσεις τεχνολογικά.

Δεν ξέρουμε αν το DeepGraphs θα υιοθετηθεί ποτέ σε μεγάλη ή μικρή κλίμακα, ούτε αν θα μπορέσει ποτέ να επιτρέψει την απόλυτη και ανενόχλητη ελευθερία των υπηρεσιών στο internet όπως την έχουμε φανταστεί. Αυτό εξαρτάται από πολλούς παράγοντες, που δεν αφορούν μόνο την τεχνολογική αρτιότητα μίας πρότασης. Πολλές προτάσεις που ήταν αρτιμελής και πολλά υποσχόμενες «σκόνταψαν» γιατί δεν είχαν την αποδοχή του κόσμου. Χαρακτηριστικό παράδειγμα είναι η RDF, μία πρόταση που ανέπτυξε ο ίδιος ο ιδρυτής του internet και παρόλα αυτά έμεινε στην αφάνεια για πολύ καιρό, καθώς θεωρήθηκε περίπλοκη από τους προγραμματιστές. Αντίθετα, τεχνολογίες ημιτελείς μπορεί καμιά φορά να βγουν ακόμα και σε παραγωγικά συστήματα.

Επομένως το αν θα γίνει αποδεκτό από την κοινότητα και θα εφαρμοστεί, αν σε λίγα χρόνια το internet βασίζεται πάνω σε αυτό το πρότυπο που τώρα περιγράφουμε θεωρητικά ακόμα -ή σε κάποιο παρόμοιο- δεν το ξέρουμε και δεν μπορούμε να έχουμε ικανοποιητική απάντηση για το αν αυτό θα γίνει ή όχι. Αυτό που εμείς πιστεύουμε όμως είναι ότι το internet αργά ή γρήγορα θα οδηγηθεί προς αυτή την κατεύθυνση, και σύντομα θα προκύψει η ανάγκη για ένα αντίστοιχο πρότυπο. Όταν αυτό συμβεί, εμείς θα είμαστε περήφανοι αν με αυτή την εργασία μας έχουμε βοηθήσει έστω και λίγο στο να εξελιχθεί το internet ένα βήμα παραπάνω.

8

Βιβλιογραφία

T. Berners-Lee, “Information Management: A Proposal,” *CERN*, 1989. [Online]. Available: http://www.w3.org/History/1989/proposal.html. [Accessed: 23-Apr-2011].

R. Cailliau, “A Little History of the World Wide Web,” *W3C*, 1995. [Online]. Available: http://www.w3.org/History.html. [Accessed: 16-Oct-2013].

T. Berners-Lee, “The Original HTTP as defined in 1991,” *W3C*, 1991. [Online]. Available: http://www.w3.org/Protocols/HTTP/ AsImplemented.html. [Accessed: 17-Oct-2013].

T. Berners-Lee, “W3 Naming Schemes,” 1992. [Online]. Available: http://info.cern.ch/hypertext/WWW/Addressing/Addressing.html. [Accessed: 17-Oct-2013].

T. Berners-Lee and D. Connolly, “Hypertext Markup Language (HTML),” *Internet Engineering Task Force (IETF) Draft*, 1993. [Online].
Available: http://tools.ietf.org/html/draft-ietf-iiir-html-00.

T. Berners-Lee, “RFC1630: Universal Resource Identifiers in WWW,” *Internet Engineering Task Force (IETF) Request for Comments*, 1994. [Online]. Available: http://tools.ietf.org/html/rfc1630.

T. Berners-Lee, R. T. Fielding, and L. Masinter, “RFC2396: Uniform Resource Identifiers (URI) - Generic Syntax,” *Internet Engineering Task Force (IETF) Request for Comments*, 1998. [Online]. Available: http://tools.ietf.org/html/rfc2396.

T. Berners-Lee, R. T. Fielding, and H. Frystyk Nielsen, “RFC1945: Hypertext Transfer Protocol -- HTTP/1.0,” *Internet Engineering Task Force (IETF) Request for Comments*, 1996. [Online]. Available: http://tools.ietf.org/html/rfc1945.

R. T. Fielding, J. Gettys, J. C. Mogul, H. Frystyk Nielsen, and T. Berners-Lee, “RFC2068: Hypertext Transfer Protocol -- HTTP/1.1,” *Internet Engineering Task Force (IETF) Request for Comments*, 1997. [Online]. Available: http://tools.ietf.org/html/rfc2068.

M. Duerst and M. Suignard, “RFC3987: Internationalized Resource Identifiers (IRIs),” *Internet Engineering Task Force (IETF) Request for Comments*, 2005. [Online]. Available: http://tools.ietf.org/html/rfc3987.

R. T. Fielding, “REST APIs must be hypertext-driven,” *Untangled musings of Roy T. Fielding*, 2008. [Online]. Available: http://roy.gbiv.com/untangled/ 2008/200brest-apis-must-be-hypertext-driven. [Accessed: 02-Jun-2010].

Fernando Lopes , Michael Wooldridge and A. Q. Novais, “Negotiation among autonomous computational agents: principles, analysis and challenges”, Artificial Intelligence Review, 29, pp. 1–44, 2008

Shaheen Fatima, Sarit Kraus and Michael Woolbridge, “Principles of Automated Negatiation”, Cambirdge, 2015, ISBN 978-1-107-00254-8

Michael Fitzmoser, “Simulation of Automted Negotiation”, Springer, 2010 ISBN 978-3-7091-0132-2

O. Lassila and R. R. Swick, “Resource Description Framework (RDF) Model and Syntax Specification,” *W3C Recommendation*, 1999. [Online]. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

N. Shadbolt, W. Hall, and T. Berners-Lee, “The Semantic Web Revisited,” *Intell. Syst. IEEE*, vol. 21, no. 3, pp. 96–101, May 2006.

Lanthaler, Markus, and Christian Gütl. "Hydra: A Vocabulary for Hypermedia-Driven WebAPIs." LDOW. 2013. APA

Rigas, Emmanouil, GeorgiosMeditskos, and Nick Bassiliades. "SWRL2COOL: object-oriented transformation of SWRL in the CLIPS production rule engine." Artificial Intelligence: Theories and Applications (2012): 49-56.

Ian Horoks, Peter F Patel-Schneider, et al. “SWRL: A Semantic Web Rule Language Combining Owl and RuleML 2004: <http://www.w3.org/Submission/SWRL/.>