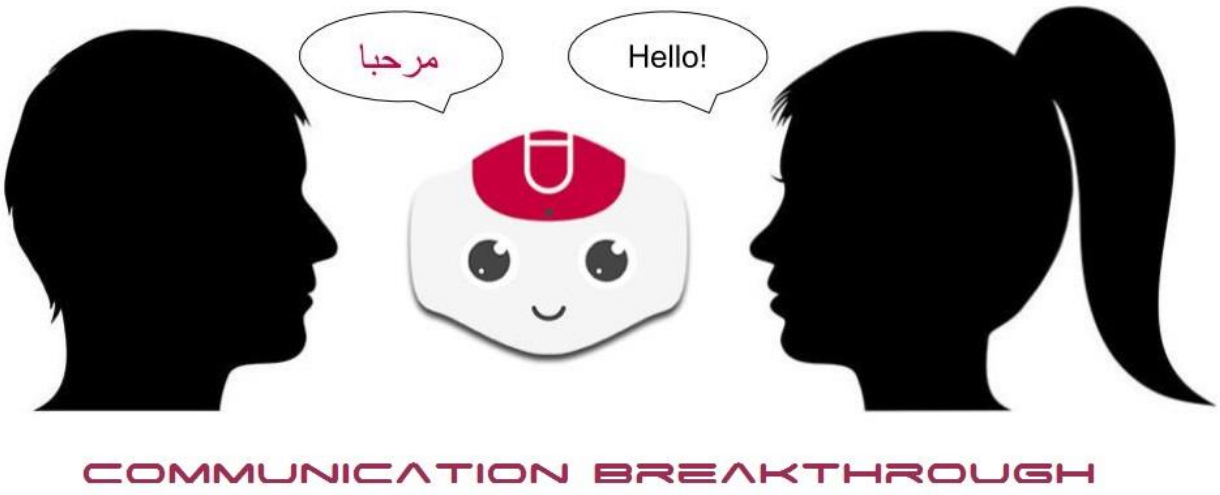
NAO Translate



Σχεδίαση Συστήματος

Del.3.1

Version 0.1.4

Βαρβαρίγγος Ιωάννης 8782 varvarip@ece.auth.gr

Γιαννακίδου Σοφία 8974 sagiannaki@ece.auth.gr

Σαχίνης Αλέξανδρος 8906 alexsach@ece.auth.gr

Χατζηχαριστού Αλεξάνδρα 8943 chatzich@ece.auth.gr

23/05/2018

Ιστορικό Αλλαγών

Όνομα	Ημερομηνία	Περιγραφή Αλλαγής	Έκδοση
Α. Συμεωνίδης	29/05/2009	Δημιουργία Εγγράφου Προσαρμογή του ESA software engineering standards guidelines (1991) και του εγγράφου SDD document, από τους Bruegge και Dutoit (2004).	0.1
NAO Translate	19/5/2018	Δυναμική Μοντελοποίηση Συστήματος και Διαγράμματα Ακολουθιών	0.2
NAO Translate	21/5/2018	Αποδόμηση Συστήματος, Έλεγχος Πρόσβασης και Ασφάλεια	0.3
NAO Translate	22/5/2018	Απεικόνιση Υλικού/Λογισμικού	0.3.2
NAO Translate	23/5/2018	Τελική Μορφοποίηση Κειμένου	0.4

Μέλη Ομάδας Ανάπτυξης

Όνομα	ΟΑ	Email
Α. Συμεωνίδης	*	asymeon@issel.ee.auth.gr
Ι. Βαρβαρίγγος	NAO Translate	varvarip@ece.auth.gr
Σ. Γιαννακίδου	NAO Translate	sagiannaki@ece.auth.gr
Α. Σαχίνης	NAO Translate	alexsach@ece.auth.gr
Α. Χατζηχαριστού	NAO Translate	chatzich@ece.auth.gr

Πίνακας Περιεχομένων

Λίστα Σχημάτων

1. Εισαγωγικά

1.1 Στόχος του Εγγράφου

Το παρόν έγγραφο αποσκοπεί στη σχεδίαση του συστήματος της εφαρμογής, μέσω της δυναμικής μοντελοποίησης αυτού και επιλογής κατάλληλης αρχιτεκτονικής. Προκειμένου να επιτευχθεί ο στόχος αυτός είναι απαραίτητο να αποδομηθεί το σύστημα σε μεμονωμένα αυτοτελή υποσυστήματα, τα οποία όμως μπορούν να συγχρονίζονται και να εδραιώνουν μία αξιόπιστη και αποτελεσματική επικοινωνία μεταξύ τους.

Για την επίδειξη των αλληλεπιδράσεων μεταξύ των υποσυστημάτων κατασκευάζονται διαγράμματα τμημάτων και διαγράμματα ανάπτυξης. Με τα διαγράμματα τμημάτων καταγράφονται τα εννοιολογικά συνεκτικά τμήματα και οι μεταξύ τους εξαρτήσεις, ενώ με τα διαγράμματα ανάπτυξης μοντελοποιούνται τα τμήματα του λογισμικού, ώστε να είναι εφικτός ο έλεγχος του συστήματος αναφορικά με την επίδοση του και την ασφάλεια που παρέχει στους χρήστες και να μπορεί να εκτιμηθεί το κόστος του έργου. Στο στάδιο αυτό είναι δυνατός ο έλεγχος της ορθότητας της επιλεχθείσας αρχιτεκτονικής για το λογισμικό του συστήματος.

Παράλληλα, η δυναμική μοντελοποίηση του συστήματος περιλαμβάνει αφηγήσεις σεναρίων χρήσης, οι οποίες παρουσιάζονται μέσω των διαγραμμάτων ακολουθιών. Τα σενάρια χρήσης που δημιουργήθηκαν στο έγγραφο των απαιτήσεων χρηστών μετασχηματίζονται σε σενάρια με απεικόνιση κλάσεων.

Έτσι τελικά, καθίσταται εφικτή η λήψη αποφάσεων για τα είδη λογισμικού που θα χρησιμοποιηθούν, τη σχεδίαση του υλικού και την ένταξη στο στάδιο της ανάπτυξης και υλοποίησης του συστήματος.

1.2 Τυπογραφικές παραδοχές

Το παρόν έγγραφο χρησιμοποιεί γραμματοσειρά Calibri μεγέθους 11 για το κυρίως κείμενο. Για τις επικεφαλίδες το μέγεθος μπορεί να είναι 16, 14 ή 12, ανάλογα με το βάθος στο οποίο βρίσκεται η κάθε μία. Υπογράμμιση χρησιμοποιείται κατά την επεξήγηση των χαρακτηριστικών των κλάσεων του συστήματος που αναλύεται. Χρησιμοποιείται bold μωβ γραφή για την αναφορά σε ιδιότητες και μεθόδους μίας κλάσης υπό τη μορφή: **κλάση::Ιδιότητα** και **κλάση::Μέθοδος(όρισμα)** και σε άλλα διαγράμματα ακολουθίας υπό τη μορφή: **όνομα διαγράμματος**. Τα διαγράμματα που παρουσιάζονται ακολουθούν τις παραδοχές της γλώσσας UML.

1.3 Αναγνωστικό κοινό

Για την πλήρη και σωστή κατανόηση του εγγράφου σχεδίασης λογισμικού, αλλά και την ορθή εκτίμηση αυτού, απαραίτητη κρίνεται η πρότερη μελέτη των εγγράφων απαιτήσεων χρηστών και λογισμικού. Το παρόν έγγραφο τηρεί μια συγκεκριμένη οργάνωση και ροή, την οποία καλούνται οι αναγνώστες να ακολουθήσουν προς διευκόλυνσή τους στην κατανόηση των περιεχομένων του.
Αναγνωστικό κοινό του εγγράφου αποτελούν οι εξής ομάδες ατόμων:

- Μηχανικοί λογισμικού που θα αναλάβουν τη συγγραφή, δοκιμή και αποσφαλμάτωση του υπό ανάπτυξη λογισμικού.

- Μηχανικοί λογισμικού που θα αναλάβουν τη συντήρηση και επέκταση του συστήματος, όταν αυτό απαιτηθεί από τις επικρατούσες συνθήκες.

1.4 Σκοπός του Έργου

Η εφαρμογή έχει σκοπό να χρησιμοποιεί το NAO ως διερμηνέα. Η χρήση της εφαρμογής θα παρέχει τη δυνατότητα διεξαγωγής διαλόγου μεταξύ ατόμων που μιλούν διαφορετικές γλώσσες. Με δυνατότητες συνομιλίας έως και 4 διαφορετικές γλώσσες ταυτόχρονα, γεφυρώνεται το χάσμα που υπήρχε ανάμεσα σε άτομα από διαφορετικές χώρες, ενώ με τη χρήση του NAO διασφαλίζεται μια ευχάριστη συζήτηση χάρη στην, κατά το δυνατόν ανθρώπινη αλληλεπίδραση του με τους ομιλητές.

Η ομάδα ανάπτυξης προσδοκεί με τη δημιουργία της εφαρμογής να φέρει τους ανθρώπους πιο κοντά και να εμβαθύνει τις διαπροσωπικές σχέσεις των χρηστών της ανεξάρτητα από την καταγωγή τους. Η χρήση της μπορεί να αποτελέσει κίνητρο γνωριμίας άλλων πολιτισμών, διεύρυνσης των οριζόντων των χρηστών και να θέσει τα θεμέλια για έναν νέο τρόπο επικοινωνίας.

2. Δυναμική μοντελοποίηση του συστήματος

2.1 Πακέτο Φωνητικών Εντολών

Το πακέτο αυτό περιλαμβάνει τη λήψη και επεξεργασία των φωνητικών εντολών του χρήστη από το σύστημα.

Η Λειτουργική Απαίτηση που σχετίζεται με αυτό το πακέτο είναι η:

- ΛΑ-2.

Αφήγηση Σεναρίου

Όταν ο χρήστης πατήσει το κεντρικό κουμπί στο κεφάλι του NAO και εκφωνήσει μία φωνητική εντολή, το NAOqi OS καλεί τη συνάρτηση **NAOqiAudioProxy::onButtonPressed()**. Αυτή καλεί τη συνάρτηση **VoiceOrderController::checkVoiceOrder(voiceOrder)** με όρισμα τη φωνητική εντολή, όπως την αντιλήφθηκε το NAOqi OS, σε μορφή συμβολοσειράς, με σκοπό να ελέγξει αν αυτή ανήκει στη λίστα των φωνητικών εντολών. Καλεί συνεπώς την **VoiceOrders::getListOfVoiceOrders()** και πραγματοποιεί αυτόν τον έλεγχο. Τονίζεται ότι σε περίπτωση που ο χρήστης δεν εκφωνήσει κάποια φωνητική εντολή το **voiceOrder** θα είναι NULL, ενώ αν εκφωνήσει κάτι που δεν κατάφερε να γίνει αντιληπτό από το NAOqi, θα επιστρέψει το String “00”. Με βάση το αποτέλεσμα του ελέγχου, καλείται η **VoiceOrderController::voiceOrderConfirmation(confirmed)**, η οποία με τη σειρά της καλεί τις **NAOqiAudioProxy::playFile(fileName)**, **NAOqiSensorsProxy::openLED(success)** με όρισμα “true” ή “false”, ανάλογα με την αναγνώριση ή μη της φωνητικής εντολής του χρήστη, ώστε να αναπαραχθεί ήχος επιβεβαίωσης ή απόρριψης και να ανάψουν τα πράσινα ή κόκκινα LED στα μάτια του NAO αντίστοιχα.

Εδώ τερματίζει το Σενάριο Χρήσης, όπως αυτό ορίστηκε στο 1ο Παραδοτέο. Ωστόσο, για την προγραμματιστική συνέχεια απαιτείται και μία ακόμα ενέργεια που εξασφαλίζει την υλοποίηση της εκφωνηθείσας φωνητικής εντολής:

Αν η φωνητική εντολή αναγνωριστεί και ανήκει στη λίστα των φωνητικών εντολών, καλείται η συνάρτηση **VoiceOrderController::implementVoiceOrder(voiceOrder: String)**, για την υλοποίησή της. Υπάρχουν οι ακόλουθες περιπτώσεις ανάλογα με την τιμή του String:

- Translate: Καλεί τη συνάρτηση **ActivateTerminateController::activation()**.
- Terminate : Καλεί τη συνάρτηση **ActivateTerminateController::termination()**.
- Add/Delete : Καλεί τη συνάρτηση **AddDeleteController:: activity(String)** με είσοδο το String ‘Add’/’Delete’.

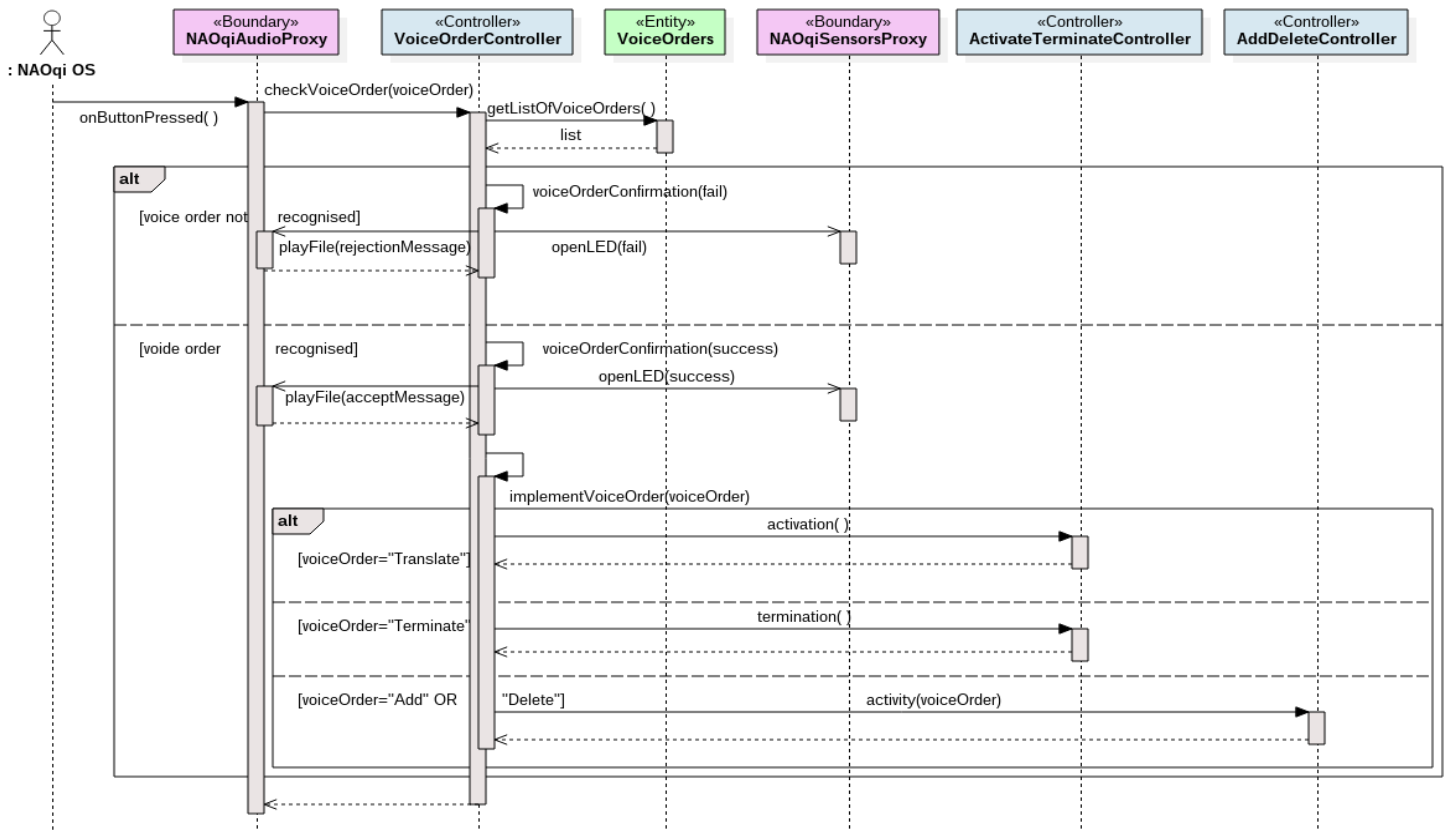


Figure 1: Voice Order Sequence Diagram

2.2 Πακέτο Μετάφρασης

Το πακέτο αυτό υλοποιεί τη μετάφραση των λεγομένων του ομιλητή σε όλους τους ακροατές. Οι λειτουργικές απαιτήσεις που σχετίζονται με αυτό το πακέτο είναι οι:

- ΛΑ-4, ΛΑ-5, ΛΑ-6, ΛΑ-8.

Αφήγηση Σεναρίου

Το NAOqi OS καλεί τη συνάρτηση **NAOqiAudioProxy::startMicrophoneRecording()**, μόλις ανιχνευθεί ήχος και ενώ το σύστημα βρίσκεται σε κατάσταση αναμονής. Η συνάρτηση αυτή καλεί με τη σειρά της την **TranslationController::setSpeaker().** Αυτή θέτει "false" τις τιμές των μεταβλητών **NAOqiPeoplePerceptionProxy::isEyeContant** και **NAOqiPeoplePerceptionProxy::isSpeechFinished**.

Έπειτα, καλεί την **NAOqiPeoplePerceptionProxy::searchForSpeaker()**

- Αν επιστρέψει NULL, τότε ο ομιλητής δεν έχει βρεθεί πριν ολοκληρωθεί ο λόγος του και καλείται η **TranslationController::errorHandle()**, για την ταυτοποίησή του. Έτσι, η τελευταία καλεί την **NAOqiAudioProxy::stopMicrophoneRecord()**, για να σταματήσει την καταγραφή λόγου και την **NAOqiPeoplePerceptionProxy::smileDetection()**, για να εντοπίσει τον ομιλητή. Αν αυτός δεν εντοπιστεί, εμφανίζεται μήνυμα σφάλματος και το σενάριο τερματίζει. Διαφορετικά, εκτελούνται οι ενέργειες του **setTranslationInfo**.Τέλος, καλείται η **TranslationController::setTranslatedSpeeches()** ώστε να συνεχιστεί η διαδικασία της μετάφρασης, όπως θα εξηγηθεί παρακάτω.
- Διαφορετικά, μόλις κάποια από τις **NAOqiPeoplePerceptionProxy::isSpeechFinished**,

NAOqiPeoplePerceptionProxy::isEyeContact γίνει “true”, καλείται η **TranslationController::stopRecording()**. Η μέθοδος αυτή καλεί την **NAOqiAudioProxy:: stopMicrophoneRecord()**, για να τερματιστεί η ηχογράφηση και στη συνέχεια εκτελούνται οι ενέργειες του **setTranslationInfo**.

Έπειτα, καλείται η **TranslationController::setTranslatedSpeeches()** η οποία καλεί την **TranslationSystemProxy::translate(speech languagesToTranslate)**, για να λάβει τα μεταφρασμένα αρχεία από το εξωτερικό σύστημα μετάφρασης. Στη συνέχεια, καλεί την **Translation:: setTranslatedSpeeches(translatedSpeeches)**, ώστε να αποθηκευτούν τα μεταφρασμένα αρχεία, ενώ τέλος, την **TranslationController::translateToParticipant()** για να ξεκινήσει η μετάφραση. Στη συνέχεια, εκτελούνται οι ενέργειες του **makeTranslation**.Η διαδικασία επαναλαμβάνεται, μέχρις ότου γίνει μετάφραση σε όλες τις γλώσσες του **Translation::languagesToTranslate**.

setTranslationInfo

Αρχικά δημιουργείται ένα αντικείμενο τύπου **Audio** δίνοντας ως ορίσματα στον constuctror το καταγεγραμμένο αρχείο ήχου με τα λεγόμενα του ομιλητή και τη γλώσσα του. Στη συνέχεια, αποθηκεύεται ο ομιλητής και ο λόγος του στο **TranslationEntity**. Τέλος, υπολογίζονται οι γλώσσες στις οποίες πρέπει να γίνει η μετάφραση, βλέποντας το ποιες είναι παρούσες στη συζήτηση γλώσσες και αφαιρώντας από αυτές τη γλώσσα του ομιλητή.

makeTranslation

Αρχικά, καθορίζονται οι γλώσσες στις οποίες πρέπει να γίνει η μετάφραση μέσω της **Translation::getLanguagesToTranslate()** και καλείται η **NAOqiPeoplePerceptionProxy:: searchForParticipant(languagesToTranslate)**, για να βρεθεί ένας συμμετέχοντας. Αμέσως μετά, η **NAOqiPeoplePerceptionProxy::eyeContact(faceID)**, ώστε να διατηρηθεί οπτική επαφή με το συμμετέχοντα που εντοπίστηκε. Τέλος, χρησιμοποιώντας τις **NAOqiMotionProxy::talkGestures(emotion)** και **Translation::getTranslatedSpeeches()** μεταφράζεται ο λόγος του ομιλητή στη γλώσσα του συμμετέχοντα με συνοδεία κατάλληλων χειρονομιών, ενώ γίνεται update στις γλώσσες στις οποίες πρέπει να γίνει η μετάφραση με την **Translation::setTranslatedSpeeches(remainingLanguages).**

Στη συνέχεια παρουσιάζονται το διάγραμμα ακολουθίας του σεναρίου της Μετάφρασης και οι αναφορές **setTranslationInfo**, **makeTranslation**.

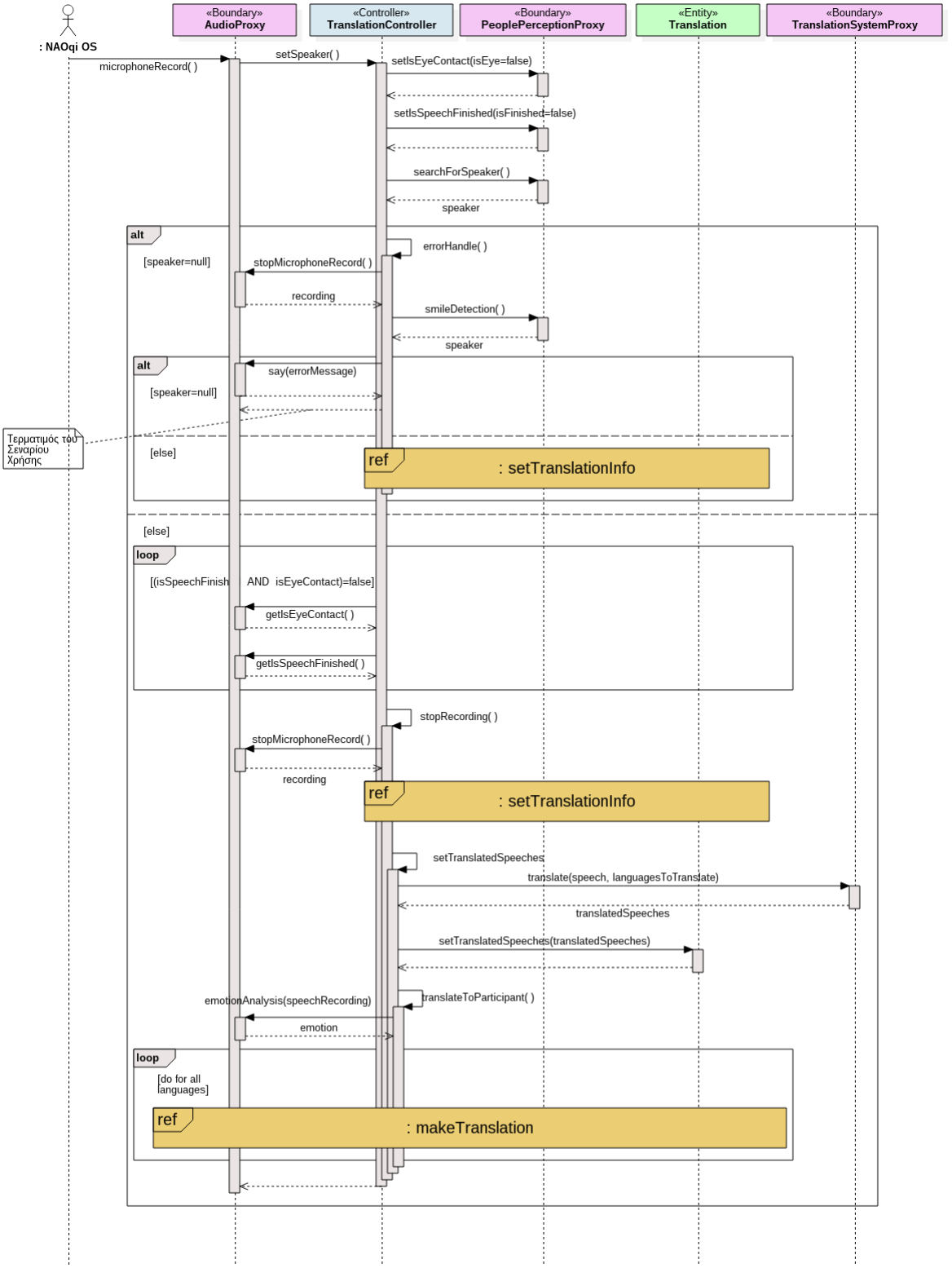


Figure 2: Translation Sequence Diagram

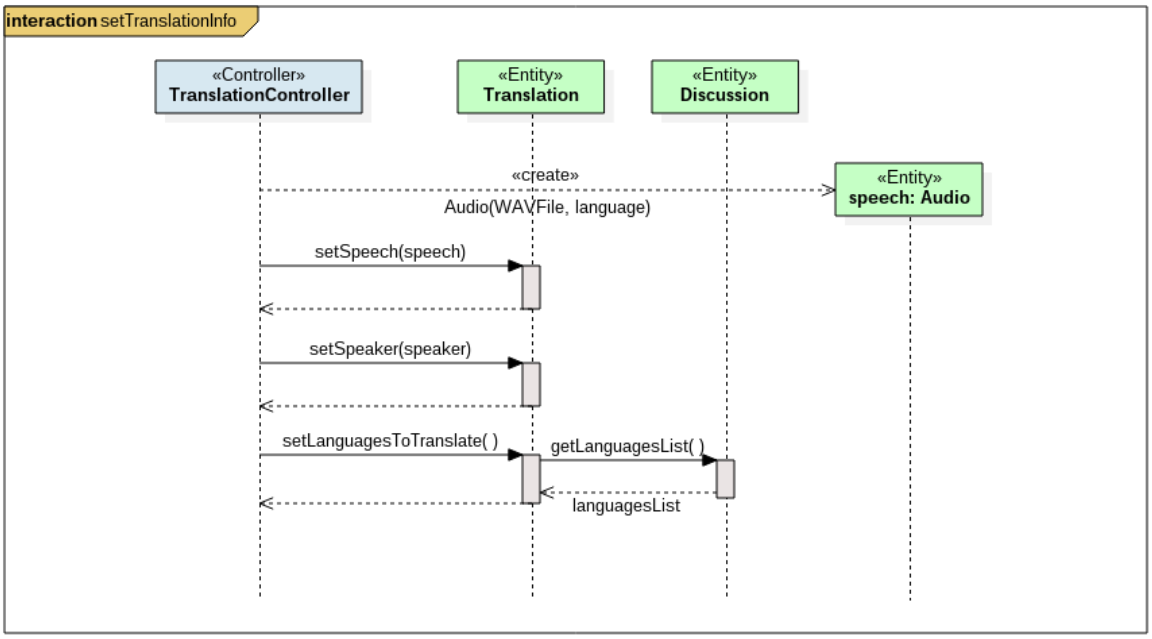


Figure 3: setTranslationInfo Reference

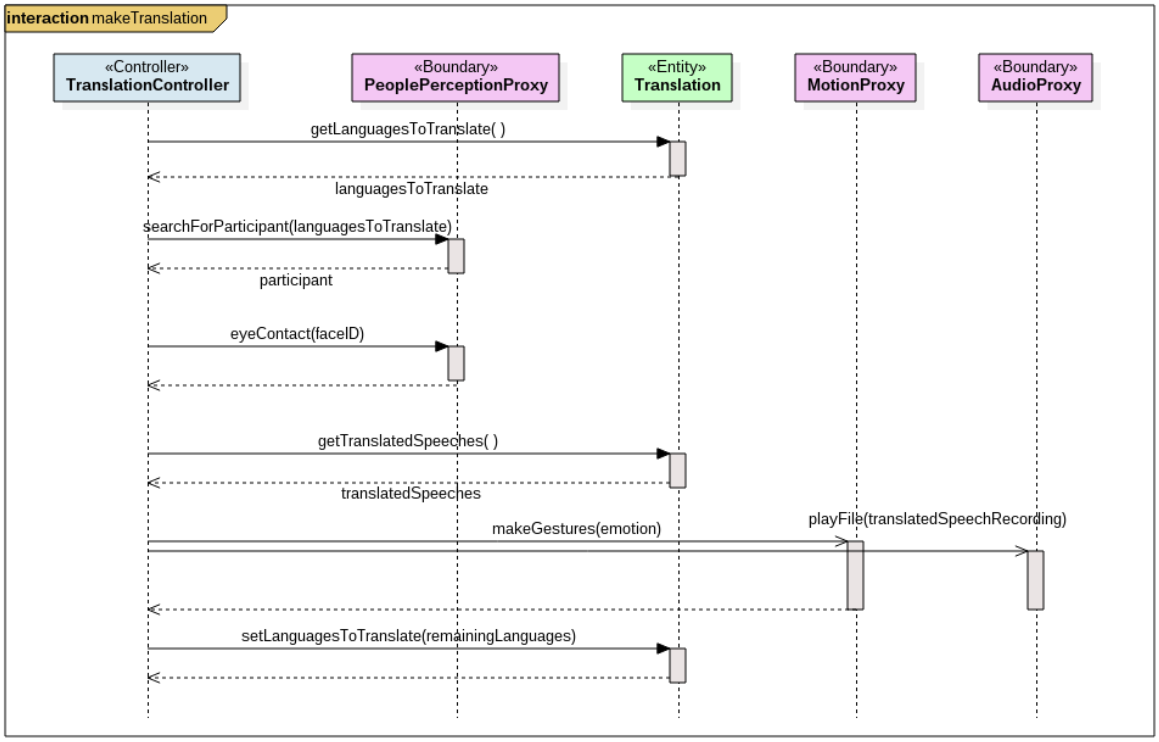


Figure 4: makeTranslation Reference

2.3 Πακέτο Γνωριμίας

Το πακέτο αυτό περιλαμβάνει τη γνωριμία των συμμετεχόντων με το σύστημα, την αποθήκευση των απαραίτητων στοιχείων τους για την πραγματοποίηση της μετάφρασης (δήλωση γλώσσας, αποθήκευση προσώπου).

Οι λειτουργικές απαιτήσεις που συνδέονται με αυτό το πακέτο είναι:

- ΛΑ-3, ΛΑ-8

Το σενάριο αυτό πυροδοτείται από την ενεργοποίηση του συστήματος. Για αυτόν τον λόγο ως "Ενεργός δράστης", στο διάγραμμα ακολουθίας, φαίνεται η κλάση ActivateTerminateController, μέσα στην οποία καλείται η IntroductionController.

Αφήγηση Σεναρίου

Αμέσως μετά την ενεργοποίηση, καλείται η συνάρτηση **IntroductionController::welcomeMessage()**, μέσω της οποίας καλούνται η **NAOqiAudioProxy::say(String)**, με όρισμα ένα μήνυμα που θα καλωσορίζει τους χρήστες, ζητώντας τους να καθίσουν σε ημικύκλιο γύρω του NAO και η **IntroductionController::searchForParticipants()**. Αυτή αποτελείται από μία επαναληπτική διαδικασία, η οποία εκτελείται όσο η **IntroductionController::endIntroduction()** είναι ψευδής Η υλοποίηση αυτής πραγματοποιείται στην **endIntroduction**. Εντός της επανάληψης καλούνται οι **NAOqiPeoplePerceptionProxy::eyeContact(fID)**, **NAOqiPeoplePerceptionProxy::smileDetection(fID)**, με δοθέν όρισμα την **NAOqiPeoplePerceptionProxy::getLastElementOfFaceIDList()**.Αν η τελευταία επιστρέψει αληθή τιμή, εκτελούνται οι ενέργειες της **handshake**. Αν η **NAOqiSensorProxy::handshakeCheck()** είναι αληθής, καλείται η **IntroductionController::validateLanguage():String** και εκτελούνται οι ενέργειες της **validateLanguage**. Η μέθοδος αυτή θέλει να επιστρέψει τη γλώσσα του χρήστη. Αφού η **NAOqiAudioProxy::languageIdentification()** επιστρέψει αποδεκτή τιμή, εξετάζεται αν αυτή είναι:

- "null": Ο χρήστης δεν έχει δηλώσει γλώσσα, καλείται η **IntroductionController:: rejectMessage()** και η συνάρτηση επιστρέφει τιμή null.
- Άλλο επιτρεπτό String: Ελέγχεται αν η **Discussion::languagesList** είναι πλήρης (περιέχει ήδη 4 διαφορετικά στοιχεία) και η γλώσσα που δηλώθηκε δεν ανήκει σε αυτά. Σε περίπτωση που ισχύει, η γλώσσα δεν είναι αποδεκτή, καλείται η **IntroductionController:: rejectLanguageMessage()** και η συνάρτηση επιστρέφει τιμή NULL. Διαφορετικά, η συνάρτηση επιστρέφει τη γλώσσα ως συμβολοσειρά.

Αν η **IntroductionController::validateLanguage()** επιστρέψει αποδεκτή γλώσσα, καλείται η **IntroductionController::saveParticipant(participantID, language)**. Η μέθοδος αυτή, δημιουργεί αντικείμενα της κλάσης **Participant**, ενώ παράλληλα ενημερώνει τις **Discussion::participantsList, Discussion::languagesList**. Τέλος, ελέγχεται η ξανά η τιμή της **IntroductionController::endIntroduction()** (εκτέλεση της **endIntroduction**).

endIntroduction

Εντός της **IntroductionController::endIntroduction()** καλείται η **NAOqiPeoplePerceptionProxy:: faceDetection()** και ελέγχεται η πληρότητα της **Discussion::participantsList** (κλήση της **Discussion::getParticipantsList()**.

- Αν η πρώτη είναι αληθής (υπάρχει κάποιος χρήστης προς γνωριμία στον χώρο) **και** η λίστα δεν είναι πλήρης, η **IntroductionController::endIntroduction()** επιστρέφει τιμή "false".
- Διαφορετικά, η **IntroductionController::endIntroduction()** γίνεται αληθής, καλεί την **IntroductionController::endMessage()** και μεταβάλλει το **Discussion::state** σε "Stand by". Αν η λίστα είναι πλήρης, καλείται επιπλέον η **IntroductionController::rejectParticipant**

handshake

Για την εκτέλεση της κίνηση χειραψίας, καλούνται οι **IntroductionControler::askNameMessage()**, **NAOqiMotionProxy::handshakeMovementOn()** και **NAOqiMotionProxy:: handshakeMovementOff()**, με διαφορά 3 δευτερολέπτων. Μεταξύ των δύο τελευταίων, παρεμβάλλεται συνθήκη, που εξαρτάται από την τιμή που θα επιστρέψει **NAOqiSensorProxy:: handshakeCheck(): boolean**.

validateLanguage

Καλεί την **NAOqiAudioProxy::say(String)**, προκειμένου να αναπαράγεται μήνυμα που θα ρωτάει τη γλώσσα του χρήστη. Στη συνέχεια, καλεί τη συνάρτηση **NAOqiAudioProxy::languageIdentification()** και εξετάζει με βάση το επιστρεφόμενο String αν η γλώσσα που έδωσε ο χρήστης είναι κατανοητή (έχει αποδεκτή τιμή). Αν αυτό δεν ισχύει, καλείται η **IntroductionController:: unknownLanguageMessage()** και η **NAOqiAudioProxy::languageIdentification()** ξανά, μέχρις ότου επιστρέψει αποδεκτό String ή null.

Σημειώνεται πως σε κάθε μήνυμα απόρριψης, καλείται η συνάρτηση **NAOqiAudioProxy::say(String)**, με κατάλληλο όρισμα και η **NAOqiSensorsLEDsProxy::openLED(led: boolean)**, όπου η μεταβλητή **led** έχει ψευδή τιμή.

Παρακάτω εμφανίζεται το διάγραμμα ακολουθίας του πακέτου Γνωριμίας κθώς και αυτά των αναφορών **endIntroduction**, **Handshake**, **validateLanguage**.

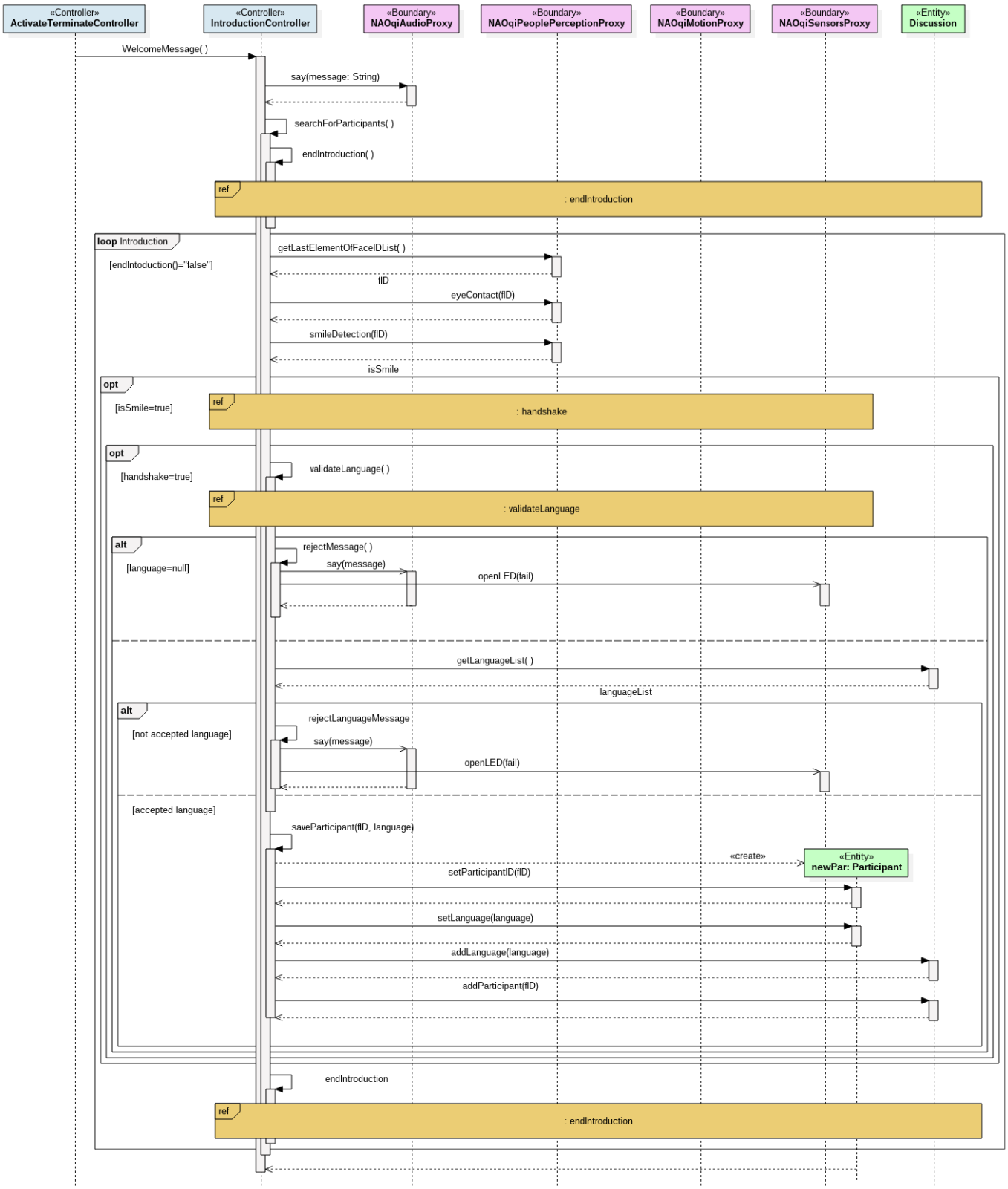


Figure 5: Introduction Sequence Diagram

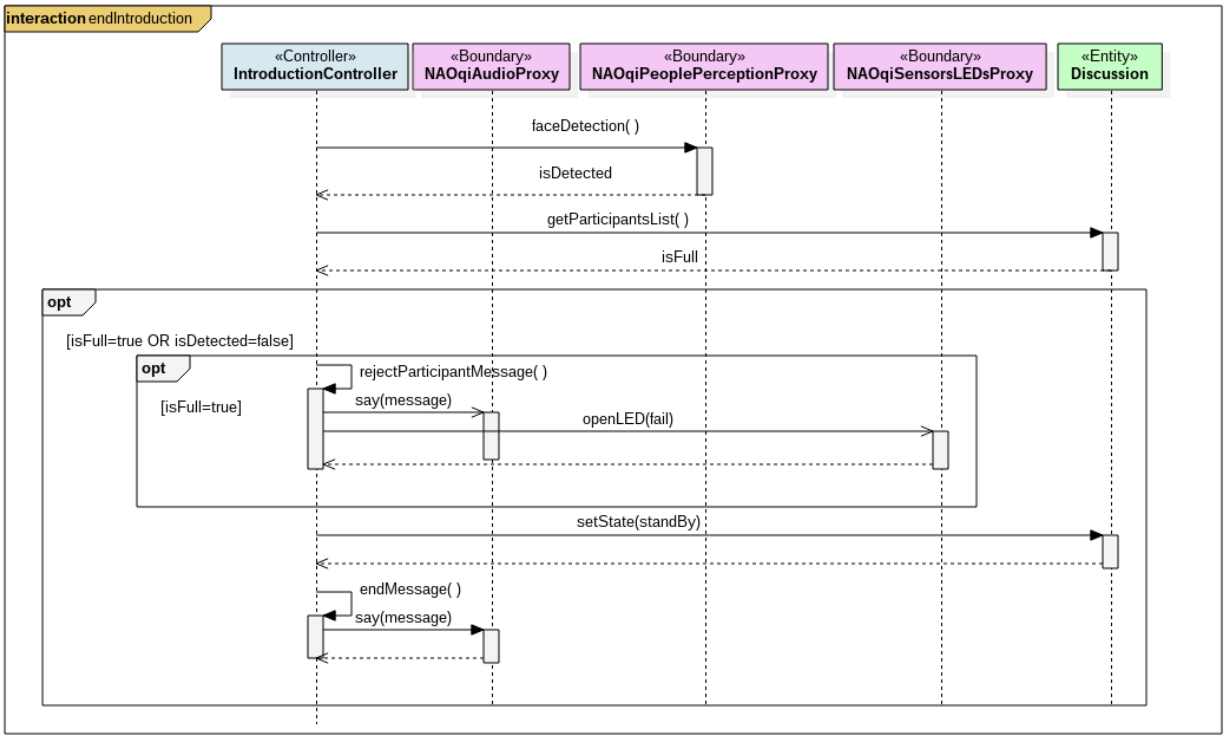


Figure 6: endIntroduction Reference

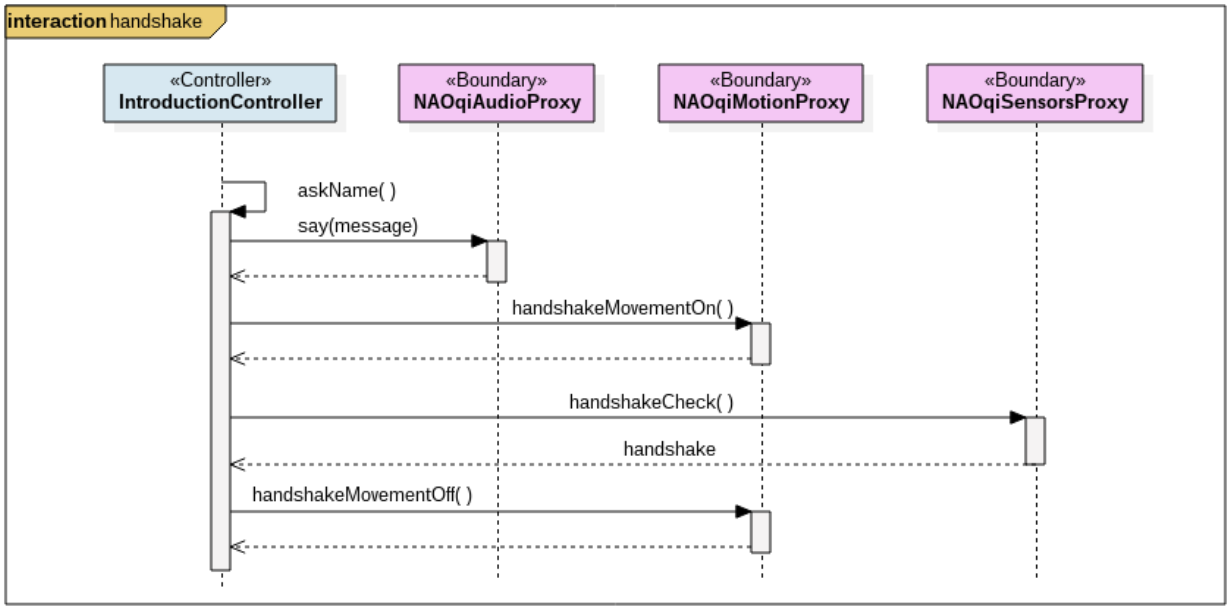


Figure 7: handshake Reference

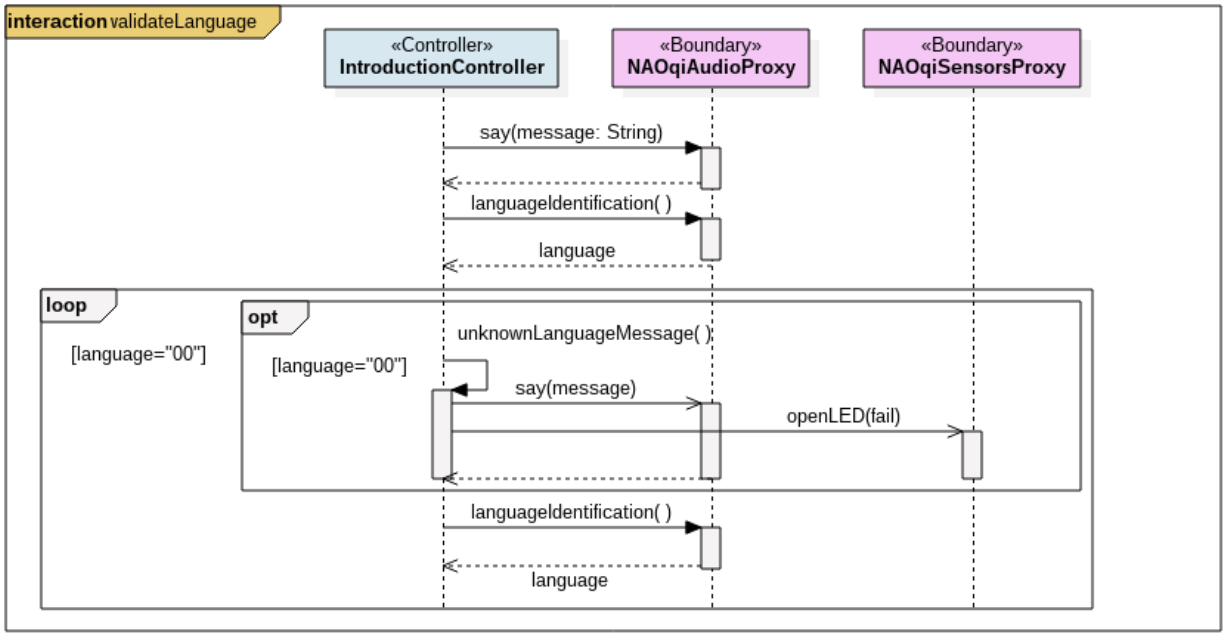


Figure 8: validateLaguage Reference

2.4 Πακέτο Διαγραφής

Το Πακέτο αυτό υλοποιεί την διαγραφή των στοιχείων ενός συμμετέχοντα από τη συζήτηση. Οι λειτουργικές απαιτήσεις που σχετίζονται με αυτό το πακέτο είναι οι:

- ΛΑ-5, ΛΑ-9.

Η διαγραφή ξεκινάει όταν αναγνωριστεί από το σύστημα ότι ο χρήστης έδωσε τη φωνητική εντολή "Delete". Έτσι, σε αυτό το διάγραμμα ως ενεργός χρήστης εμφανίζεται η κλάση VoiceOrderController.

Αφήγηση Σεναρίου:

Όταν από τον VoiceOrderController αναγνωριστεί ότι δόθηκε η εντολή "Delete", καλείται η **AddDeleteController::activity(voiceOrder)**, με όρισμα αυτήν τη φωνητική εντολή. Στη συνέχεια, αυτή καλεί τη συνάρτηση **NAOqiPeoplePerceptionProxy::detectParticipant()**, η οποία θα επιστρέψει ένα αντικείμενο τύπου Participant, που θα δοθεί ως όρισμα στην **AddDeleteController::delete(participant)**. Εντός αυτής, καλούνται οι **NAOqiPeoplePerceptionProxy::eyeContact(participant)**και **NAOqiPeoplePerceptionProxy::smileDetection(participantID)**. Ανάλογα με την τιμή της :

- Αν είναι αληθής, καλούνται οι **Discussion::deleteParticipant(Participant)** και **AddDeleteController::goodbyeMessage()**, η οποία με κλήση της **NAOqiAudioProxy::say(message)** και κατάλληλο όρισμα αποχαιρετάει τον συμμετέχοντα.
- Αν είναι ψευδής, δεν πραγματοποιείται διαγραφή και καλείται η **AddDeleteController::failureToDeleteMessage()**. Η μέθοδος αυτή καλεί την **NAOqiAudioProxy::say(message)**, προκειμένου να αναπαράγεται μήνυμα το οποίο θα δίνεται ως όρισμα, για να ενημερώσει τον χρήστη. Επίσης, ενεργοποιεί τα κόκκινα LEDS του NAO καλώντας την **NAOqiSensorsProxy::openLED(fail)**.

Η διαδικασία ολοκληρώνεται και μεταβάλλεται το **Discussion::state** σε “Stand by” (καλώντας την **Discussion::setState (state)**).

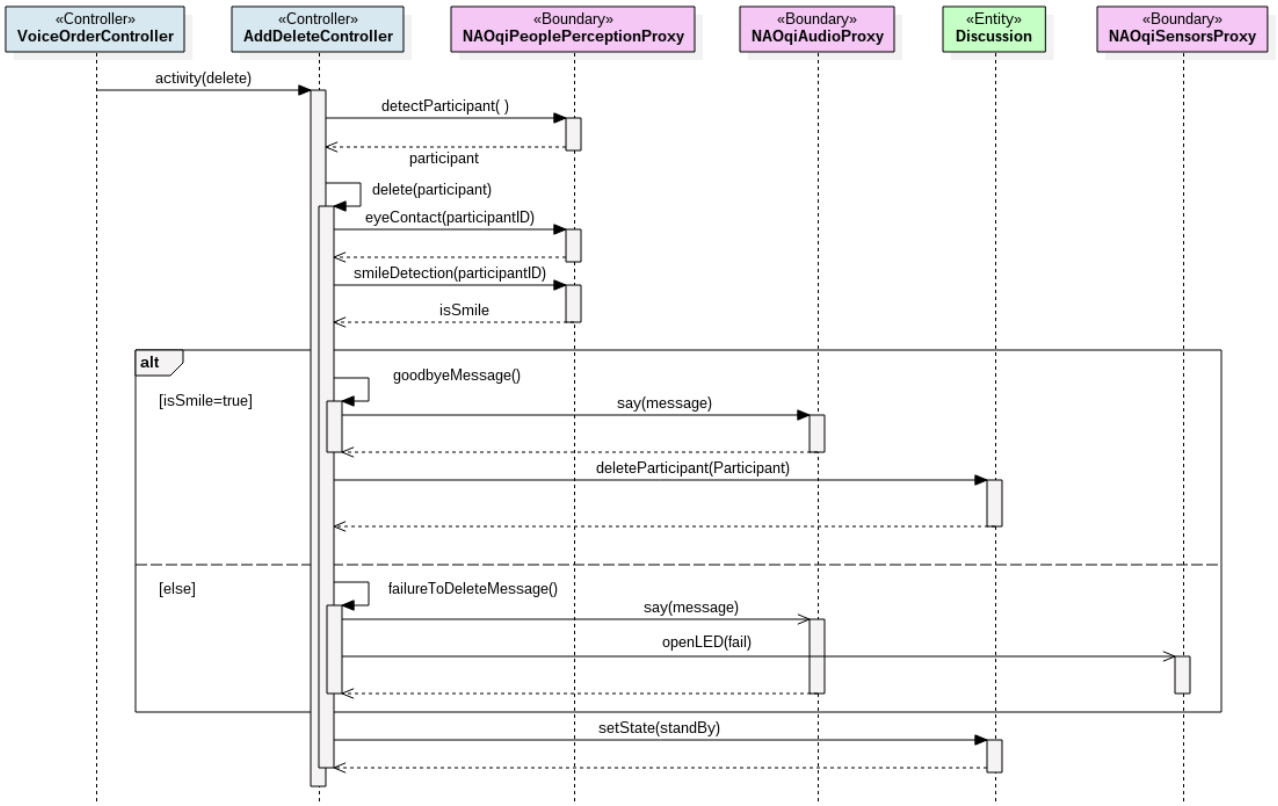


Figure 9: Delete Sequence Diagram

3. Προτεινόμενη Αρχιτεκτονική Λογισμικού

Το παρόν κεφάλαιο περιγράφει το μοντέλο σχεδίασης του συστήματος NAO Translate.

3.1.1 Η αρχιτεκτονική δύο επιπέδων (two tier)

Η αρχιτεκτονική δύο επιπέδων είναι υποκατηγορία αυτού που ονομάζεται αρχιτεκτονική πολλαπλών επιπέδων (n-tier architecture). Συνήθως τα δύο επίπεδα που απαρτίζουν αυτόν τον τύπο αρχιτεκτονικής είναι η παρουσίαση και το επίπεδο των δεδομένων (data layer). Τις περισσότερες φορές ωστόσο προτιμάται η αρχιτεκτονική τριών επιπέδων (επίπεδο παρουσίασης – επίπεδο επεξεργασίας – επίπεδο διαχείρισης δεδομένων), η οποία είναι πιο αποδοτική λόγο της διακριτοποίησης της επεξεργασίας από τη διαχείριση των δεδομένων και το επίπεδο της παρουσίασης. Γενικά, σκοπός αυτής της αρχιτεκτονικής είναι να βρίσκονται σε θέση τα επίπεδα αυτά να διατηρούνται ως ανεξάρτητες υπομονάδες και πιθανώς να υλοποιούνται ακόμη και σε διακριτές πλατφόρμες. Αυτή η ανεξαρτησία παρέχει τη δυνατότητα να προστεθεί ένα απολύτως νέο επίπεδο στο σύστημα ή να αναδιαταχθεί εξ ολοκλήρου ένα υπάρχον, χωρίς καμία επίδραση στα υπόλοιπα. Οι συνήθεις ρόλοι που καλούνται να διαδραματίσουν τα δύο αυτά επίπεδα είναι οι εξής:

- **Πρώτο επίπεδο – Επίπεδο Διαχείρισης Δεδομένων (Database Tier)**

  Συνήθως, το επίπεδο αυτό αποτελείται από τον Server στον οποίο αποθηκεύονται, ανανεώνονται και συντηρούνται τα δεδομένα που αφορούν το σύστημα. Σε αντίθεση με την αρχιτεκτονική τριών επιπέδων, σε αυτήν την αρχιτεκτονική μέρος της επεξεργασίας των δεδομένων γίνεται επίσης σε αυτό το επίπεδο, μειώνοντας όμως έτσι την απόδοση αυτού.

- **Δεύτερο επίπεδο – Επίπεδο Παρουσίασης ή Επίπεδο Πελάτη (Client Tier)**

  Συνήθως, το επίπεδο αυτό αποτελεί τη διεπαφή του χρήστη με το σύστημα (User Interface). Είναι υπεύθυνο για τη διαχείριση των Οθονών Εργασίας (User Screens), καθώς και για τη μορφοποίηση των δεδομένων που απαντώνται. Εδώ γίνεται επίσης μέρος της επεξεργασίας των δεδομένων μαζί με το Επίπεδο Διαχείρισης Δεδομένων.

3.1.2 Η αρχιτεκτονική του συστήματος NAO Translate

Η αρχιτεκτονική του συστήματος NAO Translate ανήκει στην κατηγορία των αρχιτεκτονικών δύο επιπέδων που παρουσιάστηκαν στην προηγούμενη παράγραφο, με μία ωστόσο σημαντική διαφοροποίηση. Δεδομένης της δόμησης του συστήματος γύρω από τον άξονα των φωνητικών εντολών, παραλείπεται το Επίπεδο Παρουσίασης (Client Tier), καθώς δεν υπάρχει γραφικό περιβάλλον ή οθόνες εργασίας. Αντ' αυτού, υλοποιείται ένα επίπεδο υπεύθυνο για τη διεπαφή του συστήματος με τα συστήματα με τα οποία αυτό συνδέεται, που ονομάστηκε **Επίπεδο Εξωτερικών Συστημάτων** (External Systems Tier). Παράλληλα, επειδή το σύστημα δεν απαιτεί την ύπαρξη Βάσεων Δεδομένων, το Επίπεδο Διαχείρισης Δεδομένων (Database Tier) αντικαθίσταται από αυτό που απαντάται ως δεύτερο επίπεδο μίας αρχιτεκτονικής τριών επιπέδων, δηλαδή το **Επίπεδο Επεξεργασίας** (Business Logic Tier). Πιο αναλυτικά, τα επίπεδα της αρχιτεκτονικής του συστήματος NAO Translate είναι τα ακόλουθα:

- **Πρώτο επίπεδο – Επίπεδο Επεξεργασίας (Business Logic Tier)**

  Στο τμήμα αυτό του λογισμικού πραγματοποιούνται οι περισσότερες διεργασίες των δεδομένων. Είναι υπεύθυνο για την ενεργοποίηση/απενεργοποίηση του συστήματος, για τη δια-

χείριση των φωνητικών εντολών, αλλά και για τη διαχείριση των χρηστών και των δεδομένων της μετάφρασης των λεγομένων τους. Τα υποσυστήματα που απαρτίζουν αυτό το επίπεδο είναι τα: **VoiceOrdersHandle**, **System Handle, UserHandle** και **TranslationHandle.**

- **Δεύτερο επίπεδο – Επίπεδο Εξωτερικών Συστημάτων (External Systems Tier)**

  Το επίπεδο αυτό είναι υπεύθυνο για τη διεπαφή του συστήματος με τα εξωτερικά συστήματα με τα οποία αυτό συνεργάζεται. Παρέχει τη δυνατότητα ελέγχου των συστημάτων του ρομπότ και επικοινωνίας με το εξωτερικό σύστημα μετάφρασης. Τα υποσυστήματα που απαρτίζουν αυτό το επίπεδο είναι τα: **RobotSystemHandle** και **TranslationSystemHandle**.

3.2 Αποδόμηση Συστήματος

Στην παράγραφο αυτή περιγράφεται η αποδόμηση του συστήματος, δηλαδή ο διαχωρισμός του συστήματος σε επιμέρους υποσυστήματα ανάλογα με τις λειτουργίες και τις αρμοδιότητες καθενός από αυτά. Κάθε υποσύστημα αποτελείται ένα τμήμα (component), μέσα στο οποίο έχουν ομαδοποιηθεί κλάσεις του συστήματος που είχαν άμεση συσχέτιση, όπως αυτές αναπτύχθηκαν στο δεύτερο παραδοτέο.

3.2.1 Υποσύστημα SystemHandle

Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται για την ενεργοποίηση και απενεργοποίηση του συστημάτος. Αποτελείται από τον ελεγκτή ActivateTerminateController. Επικοινωνεί με το υποσύστημα **UserHandle** μέσω της διεπαφής UserHandleAPI και με το υποσύστημα **RobotSystemHandle** μέσω της διεπαφής AudioAPI. Επιπλέον, επικοινωνεί με το υποσύστημα **VoiceOrderHandle** μέσω της διεπαφής SystemHandleAPI, παρέχοντάς του την πληροφορία που απαιτεί.

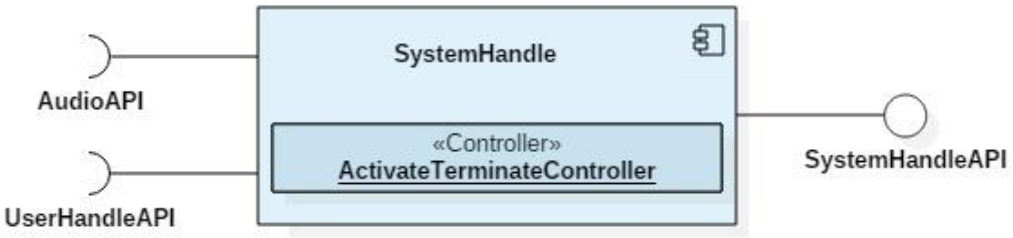


Figure 10: Υποσύστημα SystemHandle

3.2.2 Υποσύστημα VoiceOrdersHandle

Το υποσύστημα αυτό περιλαμβάνει την λειτουργικότητα που απαιτείται για την πραγματοποίηση των φωνητικών εντολών του συστήματος. Αποτελείται από τον ελεγκτή VoiceOrderController και την οντότητα VoiceOrders. Επικοινωνεί με τα υποσυστήματα: **SystemHandle** μέσω του SystemHandleAPI, **RobotSystemHandle** μέσω των AudioAPI , SensorsAPI και VoiceOrdersHandleAPI και **UserHandle** μέσω του UserHandleAPI.

Figure 11: Υποσύστημα VoiceOrdersHandle

3.2.3 Υποσύστημα UserHandle

Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται για την διαχείρηση των χρήστων, την διαδικασία γνωριμίας και αποθήκευση τους και την διαγραφή τους. Αποτελείται από τους ελεγκτές AddDeleteController και IntroductionController και τις οντότητες Discussion και Participant. Επιπλέον, επικοινωνεί με τα υποσυστήματα **RobotSystemHandle** μέσω των διεπαφών AudioAPI, MotionAPI, SensorsAPI και PeoplePerceptionAPI και με τα υποσυστήματα **SystemHandle**, **TranslationHandle** και **VoiceOrdersHandle** μέσω του UserHandleAPI.

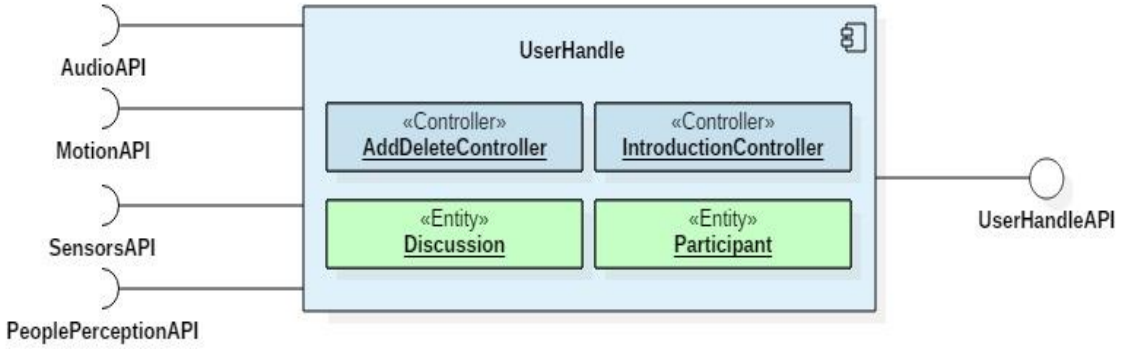


Figure 12: Υποσύστημα UserHandle

3.2.4 Υποσύστημα TranslationHandle

Το υποσύστημα αυτό περιλαμβάνει την λειτουργικότητα που απαιτείται για την διαδικασία της μετάφρασης. Αποτελείται από τον ελεγκτή Translation Controller και τις οντότητες Translation και Audio. Επιπλέον, επικοινωνεί με το υποσύστημα **RobotSystemHandle** μέσω των διεπαφών AudioAPI, MotionAPI και PeoplePerceptionAPI, με το υποσύστημα **TranslationSystemHandle** μέσω των TranslationSystemAPI και TranslationHandleAPI και με το υποσύστημα **UserHandle** μέσω της

διεπαφής UserHandleAPI.

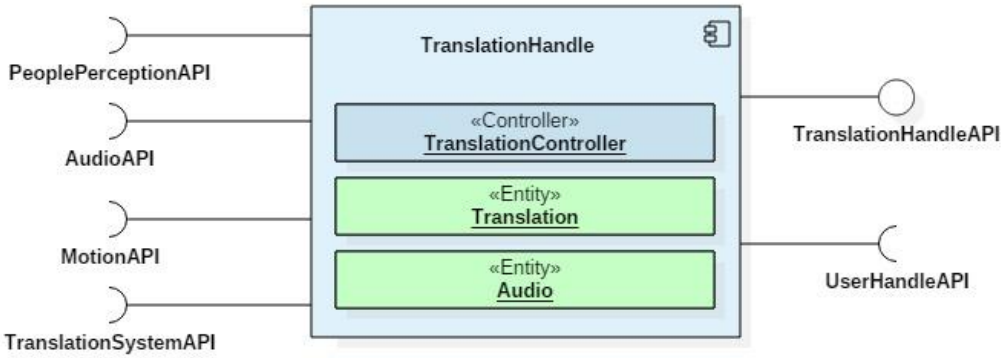


Figure 13: Υποσύστημα TranslationHandle

3.2.5 Υποσύστημα RobotSystemHandle

Το υποσύστημα αυτό περιλαμβάνει τη λειτουργικότητα που απαιτείται για τις λειτουργίες των διεπαφών του NAO με τα άλλα υποσυστήματα. Αποτελείται από τις οριακές κλάσεις NAOqiAudioProxy, NAOqiSensorProxy, NAOqiMotionProxy και NAOqiPeoplePerceptionProxy. Επιπλέον, επικοινωνεί με τα υποσυστήματα **VoiceOrdersHandle**, **SystemHandle**, **UserHandle** και **TranslationHandle** μέσω των διεπαφών AudioAPI, MotionAPI, SensorsAPI και PeoplePerceptionAPI και VoiceOrdersHandle μέσω του VoiceOrdersHandleAPI.

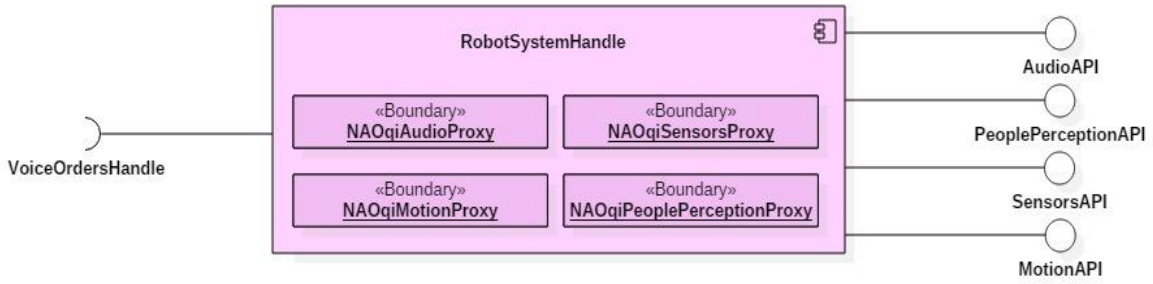


Figure 14: Υποσύστημα RobotSystemHandle

3.2.6 Υποσύστημα TranslationSystemHandle

Το υποσύστημα αυτό περιλαμβάνει τις διεπαφές οι οποίες είναι απαραίτητες για την διαδικασία της μετάφρασης. Αποτελείται από την οριακή κλάση TranslationSystemProxy και επικοινωνεί με το υποσύστημα **TranslationHandle** μέσω του TranslationHandleAPI και του TranslationSystemHandleAPI.

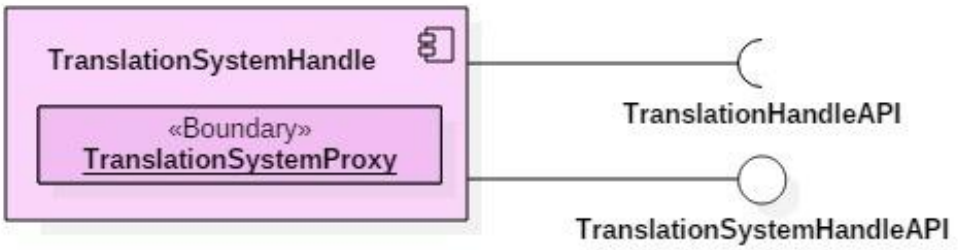


Figure 15: Υποσύστημα TranslationSystemHandle

3.2.7 Διάγραμμα Τμημάτων

Παρακάτω παρατίθεται το συνολικό διάγραμμα τμημάτων (Component Diagram) με σκοπό την επίδειξη των συσχετήσεων και αλληλεπιδράσεων μεταξύ των επιμέρους υποσυστημάτων. Τα διαγράμματα αυτά βοηθούν στον καταμερισμό των εργασιών μεταξύ των μελών μίας ομάδας ανάπτυξης, αλλά και στον έλεγχο της λειτουργικότητας του συστήματος.

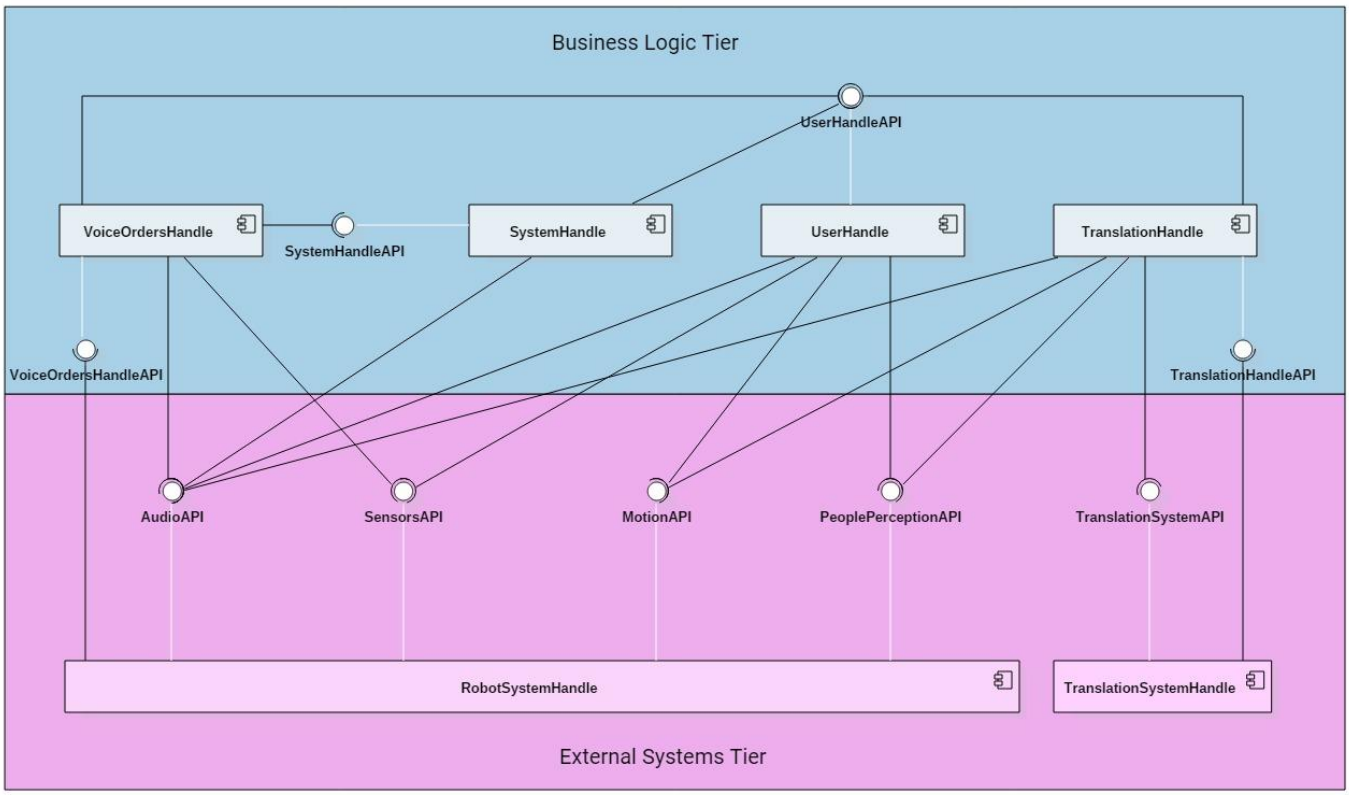


Figure 16: Component Diagram

3.3 Απεικόνιση Υλικού/Λογισμικού

Παρακάτω παρουσιάζεται διαγραμματικά, μέσω της UML, η απεικόνιση της σχέσης μεταξύ υλικού και λογισμικού του συστήματος NAO Translate. Για το σκοπό αυτό έχει σχεδιαστεί το αντίστοιχο διάγραμμα ανάπτυξης (Deployment Diagram) του συστήματος. Στόχος αυτής της παραγράφου είναι η φυσική παρουσίαση του συστήματος, ώστε να καταστούν πλήρως κατανοητές οι ανάγκες που συνοδεύουν την πραγματική ύπαρξη και λειτουργία του. Παράλληλα, η μελέτη αυτή συνδράμει και στην καλύτερη και πιο ολοκληρωμένη κατανόηση του υπό ανάπτυξη συστήματος.

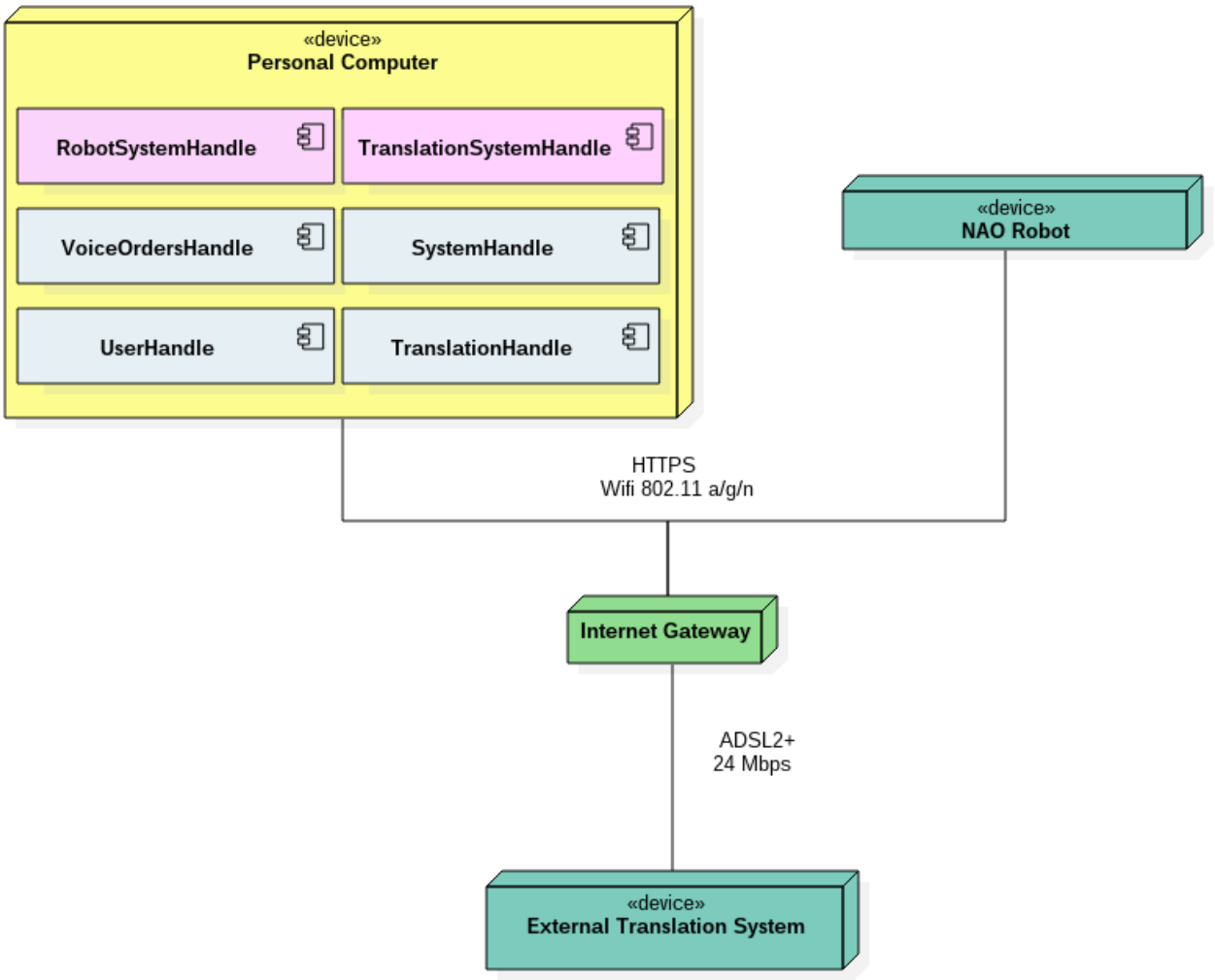


Figure 17: Deployment Diagram

3.3.1 Φυσική Υπόσταση

Το σύστημα ως προς τη φυσική του υπόσταση βρίσκεται και εκτελείται στον προσωπικό υπολογιστή του πελάτη. Η απόφαση αυτή λήφθηκε με γνώμονα την καλύτερη επίδοση και ευκολότερη ανάπτυξή της εφαρμογής σε σχέση με το αν θα εκτελούνταν πάνω στο NAO και το χαμηλότερο κόστος ανάπτυξης και συντήρησης καθώς αποφεύγεται η οργάνωση και συντήρησης ενός cloud υπολογιστικού συστήματος, που θα καθιστούσε το τρέχον λογισμικό οικονομικά λιγότερο βιώσιμο. Ως ρομποτικό σύστημα χρησιμοποιείται το NAO το οποίο έχει το δικό του λειτουργικό, το NAOqi OS.

3.3.2 Συνδέσεις

Ο προσωπικός υπολογιστής, στον οποίο εκτελείται η εφαρμογή, συνδέεται μέσω 802.11 a/g/n σε ένα Internet Gateway (router) και από εκεί μέσω γραμμής ADSL2+ στο εξωτερικό σύστημα μετάφρασης. Το NAO επίσης συνδέεται μέσω 802.11 a/g/n στο Internet Gateway (router) και επικοινωνεί με τον προσωπικό υπολογιστή μέσω του τοπικού δικτύου που δημιουργείται. Η μεγαλύτερη πρόκληση του συστήματος αφορά την ταχύτητα με την οποία μπορεί να στέλνει και να λαμβάνει τα αρχεία από το εξωτερικό σύστημα μετάφρασης και για το λόγο αυτό είναι απαραίτητο να πληρείται αυτή η προδιαγραφή. Έτσι, είναι απαραίτητη η χρήση του ADSL2+ 24Mbps το οποίο προσφέρει και ικανοποιητικές ταχύτητες upload οι οποίες είναι ζωτικής σημασίας για την εύρυθμη λειτουργία του συστήματος. Τέλος, είναι σημαντικό να αναφερθεί ότι το πρωτόκολλο επικοινωνίας που χρησιμοποιείται μεταξύ του συστήματός μας και του εξωτερικού της μετάφρασης είναι το HTTPS με κωδικοποίηση κλειδιού 256bit. Η χρήση αυτού του πρωτοκόλλου καθίσταται απαραίτητη για την προστασία τόσο των ευαίσθητων δεδομένων φωνής που αποστέλλονται για μετάφραση, όσο και για την εξασφάλιση της εχεμύθειας ως προς το περιεχόμενο της συζήτησης.

3.4 Έλεγχος Πρόσβασης και Ασφάλεια

Δεδομένου του γεγονότος ότι η εφαρμογή στηρίζεται σε φωνητικές εντολές για την εκτέλεση των λειτουργικοτήτων της, δεν υπάρχει ο συνήθης έλεγχος πρόσβασης στο σύστημα μέσω εισαγωγής username και password, όπως στις εφαρμογές δικτύου. Ουσιαστικά, χρήστης του συστήματος μπορεί να είναι οποιοσδήποτε βρίσκεται στον ίδιο χώρο με το NAO.

Ωστόσο, υπάρχει διακριτοποίηση των χρηστών ως προς το εύρος της προσβασιμότητας που μπορεί να έχουν στο σύστημα. Συγκεκριμένα, οι χρήστες διακρίνονται στους **απλούς χρήστες** και τους **συμμετέχοντες**, ενώ οι συμμετέχοντες με τη σειρά τους διακρίνονται στους παρόντες ομιλητή και ακροατές. Διαφορά στο εύρος προσβασιμότητας έχουν μόνο οι χρήστες με τους συμμετέχοντες. Ένας χρήστης που δεν είναι συμμετέχων (απλός χρήστης) μπορεί να πατήσει το κεντρικό κουμπί στο κεφάλι του NAO και να εκφωνήσει κάποια φωνητική εντολή, αλλά δεν είναι σε θέση π.χ. να εκφωνήσει έναν λόγο, ο οποίος θα μεταφραστεί από το σύστημα.

Ο έλεγχος πρόσβασης και η ασφάλεια περιγράφουν τη μοντελοποίηση των χρηστών του συστήματος με τη μορφή ενός πίνακα πρόσβασης (Access Matrix), ο οποίος παρουσιάζεται παρακάτω.

Απλός Χρήστης	NAOqiAudioProxy	NAOqiSensorsProxy	NAOqiPeoplePerceptionProxy
	onButtonPressed( )	handshakeCheck( )	faceDetection( ) eyeContact(fID: integer) smileDetection(fID: integer)

Participant	NAOqiAudioProxy	NAOqiPeoplePerceptionProxy
	onButtonPressed( ) microphoneRecord( ) languageIdentification( )	faceDetection( ) eyeContact(fID: integer) isThereEyeContact(p: Participant) smileDetection(fID: integer) smileDetection( ) searchForParticipant(langToTranslate: String[1..3]) detectParticipant( ) searchForSpeaker( ) isSpeechFinished( )

Ασφάλεια προσωπικών δεδομένων

Πέρα όμως από την ασφάλεια που παρέχεται στους συμμετέχοντες λόγω του μικρού εύρους πρόσβασης των απλών χρηστών στη συζήτηση, υπάρχει και άλλο ένα τμήμα του σχεδιασμού που ευνοεί τον τομέα της ασφάλειας της εφαρμογής. Το σύστημα βάσει του σχεδιασμού του, δεν αποθηκεύει σε κάποια βάση δεδομένων τα προσωπικά στοιχεία του χρήστη ή τα όσα ειπώθηκαν στη συζήτηση. Κατά την διαδικασία της μετάφρασης, μόλις εκφωνηθεί ο μεταφρασμένος λόγος ενός ομιλητή και ξεκινήσει η μετάφραση ενός νέου λόγου, ο τελευταίος θα αντικαταστήσει τον προηγούμενο στη μνήμη του συστήματος. Έτσι, μετά το πέρας της χρήσης της εφαρμογής, τα δεδομένα που αφορούν τους χρήστες και τους λόγους τους διαγράφονται και δεν μπορούν πλέον να ανακτηθούν με οποιονδήποτε τρόπο. Συνεπώς, επιτυγχάνεται μεγαλύτερη ασφάλεια όσων αφορά την ιδιωτικότητα και τα προσωπικά δεδομένα των χρηστών που χρησιμοποιούν την εφαρμογή NAO Translate, αφού δεν αποθηκεύονται σε κανένα σημείο δεδομένα στη μόνιμη μνήμη. Ως προς το εξωτερικό σύστημα μετάφρασης, διευκρινίζεται ότι κανένα δεδομένο, ούτε φωνής, ούτε κειμένου που προκύπτει από την επεξεργασία των ηχητικών αρχείων, μένει αποθηκευμένο στο εξωτερικό σύστημα μετάφρασης αφού ολοκληρωθεί κάθε ξεχωριστή συνεδρία μετάφρασης. Παράλληλα, οπώς προαναφέρθηκε, χρησιμοποιείται το πρωτόκολλο επικοινωνίας HTTPS με κρυπτογράφηση κλειδιού 256bit προκειμένου να διασφαλίζεται η ασφάλεια των δεδομένων σε περίπτωση διαδικτυακής υποκλοπής.

4. Πίνακας ιχνηλασιμότητας εγγράφων Σχεδίασης και Απαιτήσεων Λογισμικού

Στο παρόν Έγγραφο Σχεδίασης Συστήματος (SDD) απαιτήθηκε η μεταβολή δύο στοιχείων συγκριτικά με το πως αυτά είχαν οριστεί στο Έγγραφο Απαιτήσεων Λογισμικού (SRD). Συγκεκριμένα:

- Η μέθοδος **NAOqiAudioProxy::onButtonPressed( ): String** που είχε οριστεί στο προηγούμενο παραδοτέο αντικαθίσταται από την **NAOqiAudioProxy::onButtonPressed( ): void**, προκειμένου να μην υπάρχει λανθασμένη επιστροφή στο Διάγραμμα Ακολουθιών.
- Επιπλέον, στο Διάγραμμα Κλάσεων του Εγγράφου Απαιτήσεων Λογισμικού είχε παρουσιαστεί ότι π.χ. η κλάση **ActivateTerminateController** δημιουργούνταν από την κλάση **VoiceOrderController.** Στο παρόν Έγγραφο θεωρήθηκε ότι όλες οι κλάσεις τύπου Controller δημιουργούνται με την εκκίνηση της εφαρμογής και με βάση αυτό το σκεπτικό δομήθηκαν τα Διαγράμματα Ακολουθιών.

Παρακάτω παρουσιάζεται ο Πίνακας Ιχνηλασιμότητας των Εγγράφων Σχεδίασης Συστήματος και Απαιτήσεων Λογισμικού:

Έγγραφο Απαιτήσεων Λογισμικού	Έγγραφο Σχεδίασης Συστήματος
NAOqiAudioProxy::onButtonPressed( ): String	NAOqiAudioProxy::onButtonPressed( ): void
Δημιουργία Controller από άλλον Controller	Μαζική δημιουργία Controllers κατά την εκκίνηση

5. Παράρτημα Ι – Ανοιχτά Θέματα

- Υπάρχει το ενδεχόμενο, κατά την περαιτέρω ανάπτυξη του συστήματος, να δομηθεί και ένα γραφικό περιβάλλον διεπαφής της εφαρμογής με τον χρήστη προκειμένου αυτός να είναι σε θέση να εισάγει τα στοιχεία του (όνομα, γλώσσα, χαρακτηριστικά προσώπου), χωρίς να είναι απαραίτητο να εμπλακεί στην περιγραφείσα λειτουργία της Εισαγωγής (Introduction). Θα πρέπει συνεπώς να υλοποιηθεί μία Desktop ή/και μία Mobile εφαρμογή που να επιτελεί την συγκεκριμένη λειτουργία. Σε αυτήν την περίπτωση, η αρχιτεκτονική του συστήματος θα παραμείνει Αρχιτεκτονική n-Επιπέδων, αλλά τα επίπεδα θα αυξηθούν από δύο σε τρία. Φυσικά, η αρχιτεκτονική που περιγράφηκε στο παρόν κείμενο, παρέχει τη δυνατότητα εισαγωγής νέου επιπέδου, χωρίς τη μεταβολή κάποιου υπάρχοντος υποσυστήματος. Έτσι, μπορεί απλά να προστεθεί ένα Επίπεδο Παρουσίασης (Presentation Tier), με σκοπό τη διαχείριση των Οθονών Εργασίας, την επιτυχημένη γραφική διεπαφή του χρήστη με το σύστημα (Graphical User Interface) και τη μορφοποίηση των δεδομένων που απαντώνται.

- Όπως συζητήθηκε στο κεφάλαιο της ασφάλειας του συστήματος, η εφαρμογή δεν χρησιμοποιεί βάσεις δεδομένων και συνεπώς δεν αποθηκεύει τα δεδομένα των χρηστών, αλλά αντίθετα αυτά διαγράφονται μετά το πέρας της χρήσης του συστήματος. Ωστόσο, ως μελλοντική προσθήκη στη δομή της εφαρμογής θα μπορούσε να δοθεί η επιλογή των χρηστών για την αποθήκευση των δεδομένων τους σε κάποια βάση δεδομένων, προκειμένου να αποφευχθεί η λειτουργία της Εισαγωγής (Introduction) τους στο σύστημα κάθε φορά που χρησιμοποιούν την εφαρμογή. Αυτή η προσθήκη θα δημιουργούσε επιπλέον ζητήματα ασφαλείας και συνεπώς θα πρέπει να μελετηθεί εκ νέου σε περίπτωση υλοποίησής της. Παράλληλα, θα ήταν απαραίτητη η μεταβολή της αρχιτεκτονικής του συστήματος από 2 Επιπέδων σε 3 Επιπέδων αρχιτεκτονική, με εισαγωγή ωστόσο, όχι ενός Επιπέδου Παρουσίασης, αλλά ενός Επιπέδου Διαχείρισης Δεδομένων (Database Tier). Το επίπεδο αυτό θα αποτελούνταν από έναν Server στον οποίο αποθηκεύονται, ανανεώνονται και συντηρούνται τα δεδομένα που αφορούν το σύστημα.