Jouni Ikonen

Luonnollisen kielen käyttöliittymä tietoliikenneverkon infrastruktuurin tietomallin käsittelyyn

Tietotekniikan kandidaatintutkielma

28. huhtikuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Jouni Ikonen

Yhteystiedot: ikonen.jouni@gmail.com

Ohjaaja: Anneli Heimbürger

Työn nimi: Luonnollisen kielen käyttöliittymä tietoliikenneverkon infrastruktuurin tietomallin käsittelyyn

Title in English: Natural Language Interface for Broadcast Network Infrastructure Model

Työ: Kandidaatintutkielma

Sivumäärä: 31+0

Tiivistelmä: Tässä tutkimuksessa selvitettiin, mitä komponentteja luonnollista suomen kieltä ymmärtävän dialogikäyttöliittymän toteuttaminen tietoliikenneverkkoinfrastruktuurin tietomallin käsittelemiseen vaatii, ja tunnistettiin tärkeimmät komponenttityypit sekä suomenkieliset tietämyskannat, jotka voidaan sovittaa kielenymmärtämisprosessin käyttöön. Tutkimuksen johtopäätöksenä on, että rajoitetusti suomen kieltä ymmärtävä käyttöliittymä on toteutettavissa, mutta kaikkiin vaatimuksiin ei pystytä vastaamaan olemassaolevin komponentein.

Avainsanat: kielen ymmärtäminen, tietoverkko, infrastruktuuri

Abstract: This research studies which components are needed to create a finnish language natural language dialogue interface for the network infrastructure data model. Study has identified key natural language understanding component types and finnish language knowledge bases that can be re-purposed for the language understanding process. Study concludes that limited finnish natural language interface for the broadcast network infrastructure model is feasible, but all requirements can not be met with available components.

Keywords: NLU, broadcast network infrastructure, language understanding, dialogue

Kuviot

Taulukot

Sisältö

1 Johdanto

Tämä tutkimus liittyy ohjelmistoprojektiin, jonka tavoitteena on toteuttaa tietoliikenneverkon infrastruktuurin suunnitteluohjelmisto. Maanlaajuisen tietoliikenneverkon infrastruktuuria koskeva tieto on useissa eri tietojärjestelmissä, ja usein tietoa muokataan ja käsitellään piirustusten muodossa. Piirustusten sijasta verkkoinfrastruktuurin keskeinen tieto on muunnettavissa koneluettavaan muotoon, jolloin tiedon käsittelyprosessi muuttuu. Sen sijaan, että muutokset kirjattaisiin käsin muokkaamalla järjestelmäpiirustuksia, suoritetaan muutokset suoraan tietomalliin. Tietomallin perusteella voidaan tuottaa piirustuksia ja käyttöliittymiä automaattisesti eri tarkoituksiin. Tiedon perusteella voidaan piirtää esimerkiksi ajantasainen järjestelmäpiirrustus, tarjota valvontanäkymä verkon laitteisiin tai luoda automaattisesti käyttöliittymä laitteiden konfigurointia varten.

Kuvatun kaltaisia ongelmia lähestytään usein luomalla sovellusaluekieli. Kokemukseni perusteella käyttäjät, joilla ei ole ohjelmointitaustaa, kokevat ohjelmoinnin omaiset ratkaisut vaikeiksi, mikä rajoittaa ohjelmointikieleen perustuvan sovelluksen adoptointia organisaatioissa. Ohjelmointiympäristön käyttäminen arkipäiväisessä tiedon käsittelyssä edellyttää myös, että käyttäjällä on sovelluksen tietomallia vastaava mentaalinen malli käsiteltävästä asiasta. Usein sovelluksen tietomalli on ohjelmoijan näkökulmasta tehty, ja se ei välttämättä vastaa arkielämän kokemusta asiasta. On siis perusteltua etsiä ratkaisua, jossa järjestelmän ohjaaminen pyritään suorittamaan luonnollista kieltä ja arkielämän käsitteitä käyttäen. Tässä tutkimuksessa keskitytään selvittämään, miten nykytiedon mukaan on toteutettavissa luonnollista suomen kieltä ymmärtävä tekstipohjainen käyttöliittymä tietoliikenneinfrastruktuurin tietomallin käsittelyyn. Tutkimuksessa selvitetään, mitä komponentteja suomen kieltä ymmättävän käyttöliittymän tekemiseen vaaditaan.

Winograd (1972) osoitti, että rakenteellisen tiedon käsittely on toteutettavissa yksinkertaista englannin kieltä käyttäen. Tänä päivänä tunnetaan useampia menetelmiä tulkita informaatiota tekstistä (Ruotsalainen 2008; Jurafsky ja Martin 2009). Lisäksi suomen kielen automaattiseen jäsentämiseen on saatavilla työkaluja (Haverinen 2014). Tavoitteena on tehdä käyttötilanteesta keskustelumainen, ja älykkäitä agentteja on tutkittu tässä tarkoituksessa (Jokinen ja McTear 2010). Winograd (1971), Haverinen (2014) ja Jurafsky ja Martin (2009) tunnis-

tavat kukin lähes samanlaisen prosessin kielen jäsentämisessä, vaikka he kukin käsittelevät aihetta eri näkökulmista. Se, miten luonnollisen kielen ymmärtämisen prosessi nähdään, ei ole vuosien kuluessa juurikaan muuttunut, mutta prosessin yksittäiset osat tunnetaan paremmin tai niille on tarjolla useampia vaihtoehtoisia toteutustapoja. Toisessa luvussa pohditaan luonnollisen kielen käyttöliittymien tutkimusta, ja sitä mihin tämä ongelma tutkimuskentässä asettuu. Ohjelmistoprojektiin perustuvassa kolmannessa luvussa esitellään tietoliikenneinfrastruktuurin tietomalli ja ohjelmistoprojektin erityiset vaatimukset perusteluineen. Neljännessä luvussa tutustutaan luonnollisen kielen käyttöliittymäarkkitehtuureihin ja komponentteihin, ja viidennessä luvussa pohditaan tunnistettuja komponentteja ohjelmistoprojektin vaatimusten näkökulmasta. Tutkimus perustuu kirjallisuuteen luku kolme poislukien.

2 Luonnollisen kielen ymmärtämisestä

Tässä luvussa esitetään, mistä luonnollisen kielen ymmärtämisessä on kyse, millaisiin tutkimussuuntiin se jakaantuu, ja mitä keskeisiä ongelmia ala käsittelee. Lopuksi esitellään luonnollisen kielen ymmärtämiseen liittyviä käsitteitä.

Luonnollisen kielen käyttöä ohjelmointiin ja tietokoneen ohjaamiseen on pohdittu tietotekniikan alkuajoilta lähtien. Haverinen (2014) määrittelee luonnollisen kielen ymmärtämisen siten, että tietoteknisesti pyritään saamaan aikaiseksi jonkinlainen ymmärrys tekstistä kielen analyysin avulla. Jurafsky ja Martin (2009) summaavat luonnollisen kielen koneymmärtämisen tutkimuksen ja kehityksen näihin päiviin asti. Molemmat tunnistavat Winogradin luonnollisen kielen ymmärtämistä käsittelevän työn merkittäväksi askeleeksi koneymmärtämisen alalla. Winograd esitti, miten luonnollinen kieli ja tieto voidaan esittää ohjelmallisina proseduureina (Winograd 1971). Ruotsalainen (2008) esittelee useita menetelmiä merkityksellisen tiedon jäsentämiseksi tekstistä, ja suomen kielelle on olemassa ohjelmalliset tekstin ymmärtämisen ja kielen jäsentämisen välineet (Haverinen 2014). Jokinen ja McTear (2010) esittävät, että ihmisen ja tietokoneen välinen dialoginomainen vuorovaikutus voidaan toteuttaa älykkäiden agenttien avulla.

Jurafskyn ja Martitinin mukaan luonnollisen kielen ymmärtämisessä käytetyt menetelmät on karkeasti jaettavissa symbolisiin ja stokastisiin. Luonnollisen kielen jäsentämisprosessissa tarvitaan molempia lähestymistapoja. Puheentunnistukseen ja koneoppimiseen käytetyt menetelmät ovat usein luonteeltaan tilastotieteeseen perustuvia eli siis stokastisia. Kielen jäsentämiseen käytetään perinteisesti symbolisia menetelmiä (Jurafsky ja Martin 2009). Kielen jäsentämisen ongelmaa on lähestytty syntaktisesti, sanastopohjaisesti ja tilastollisesti. Sanojen muotoa voidaan selvittää syntaktisesti, mutta sanojen keskinäistä riippuvuutta on tarkasteltava tilastollisesti. Syntaktinen analyysi tarkoittaa, että on olemassa säännöt, joilla sanoja voi tunnistaa ja generoida (Chomsky 2002). Sanastopohjainen analyysi käyttää esimerkkinä tarkoin määrättyä joukkoa kirjoitetun kielen esimerkkitekstejä eli korpusta (Haverinen 2014). Tilastollinen analyysi jäsentää sanoja sen mukaan, miten sanoja on käytetty suuressa joukossa tekstejä (Ruotsalainen 2008). Winogradin toteuttama synktaktinen ohjelmallinen sanojen tulkkaus on nykyään yksi vaihtoehto muiden joukossa (Winograd 1971). Tuoreemmat rat-

kaisut hyödyntävät lähes poikkeuksetta asiantuntijoiden luomia tietämyskantoja, kielitieteilijöiden rakentamia korpuksia, ontologiatietokantoja tai koneoppimista sen eri muodoissa.

Luonnollisen kielen ymmärtämisprosessi vaatii logiikkaa jo tekstin jäsentämisprosessissa, mutta sen lisäksi tarvitaan komponentteja, jotka suorittavat päättelyä tietokoneen oppimien asioiden tai tietomallien perusteella. Ajan mittaan on luotu erilaisia tekoälykieliä ja ratkaisijoita, jotka osaavat määriteltyjen sääntöjen perusteella tuottaa johtopäätöksiä. Näistä tunnetuin lienee Prolog-logiikkaohjelmointikieli. Logiikka-paradigman kielet perustuvat sääntöihin ja kyselyihin. Säännöt kuvaavat tietoa, ja kyselyillä suoritetaan päättelyä.

Tietokoneen ja ihmisen välisen kommunikoinnin tutkimus on lomittain psykologian, lingvistiikan ja tietotekniikan alojen kanssa. Tietokoneen ja ihmisen välisen dialogin tutkimuksessa käyttäjästä käytetään termiä luonnollinen käyttäjä, ohjelmiston osaprosessista käytetään usein nimeä agentti, ja ohjelmiston keskustelua hoitavasta komponentista nimeä dialogin hallinta tai diskurssianalyysi. Näillä kuvataan käyttäjän ja tietokoneen välistä tiedonvaihtoa ja sen hallintaa. Dialogin tarkoituksena on yhteisen käsityksen muodostaminen jostain asiasta, ja dialogissa on kyse viestin muodostamisesta ja viestin välittämisestä. (Jokinen ja McTear 2010)

2.1 Ongelmat

Keskeisiä ja tutkituimpia luonnollisen kielen ymmärtämiseen liittyviä ongelmia ovat: kielen kääntäminen, tiedonhaku, tiedon louhinta ja dialogin kaltaiset käyttötilanteet. Luonnollisen kielen käyttöä ohjelmoinnissa ja ohjelmakoodin generoinnissa on tutkittu, mutta vähemmän. Tähän on syynä muun muassa kritiikki, jota Dijkstra (1977) on kohdistanut luonnollisen kielen soveltuvuuteen ohjelmointitarkoituksessa. Dijkstran mukaan luonnolliselta kieleltä vaaditaan ohjelmoinnissa samaa formaaliutta kuin ohjelmointikieleltä, ja luonnolliselle kielelle ominainen epätarkkuus ja moniselitteisyys toimii saavutettavia tehokkuushyötyjä vastaan.

Jurafskyn ja Martinin mukaan jo 1950-luvulla todettiin, että kielten kääntäminen on tietokoneille vaikea ongelma (Jurafsky ja Martin 2009). Kääntämisongelma on erilainen kuin tässä tutkimuksessa lähtökohtana oleva tietomallin käsittelyongelma. Tietomallin käsittelyssä lause halutaan tulkita toteutettavaksi toiminnoksi. Kielen kääntämisessä odotetaan tuloksek-

si vain käännettyä tekstiä. Kääntämisen haasteena on se, että sanojen vastaavuuksien etsiminen ei riitä, vaan on etsittävä kielistä keskenään vastaavat käsitteet ja ilmaisut (Jurafsky ja Martin 2009).

Usein ohjelmiston halutaan vastaavan johonkin kysymykseen. Näissä sovelluksissa keskitytään avainsanojen ja viitteiden perusteella palauttamaan kysyjälle vastaus yleensä jonkin tietokannan sisällöstä. Tuoreita tunnettuja esimerkkejä näistä ovat Jeopardy tietokilpailupeliä pelannut IBM:n Watson ja Sephen Wolframin Wolfram Alpha, jotka osaavat tuottaa vastauksia luonnollisella kielellä esitettyihin kysymyksiin. (Wolfram 2011)

Tiedon louhinnassa pyritään suuresta määrästä tekstimateriaalia tunnistamaan faktoja tai määrittelemään asioiden välisiä suhteita, eli ontologioita (Ruotsalainen 2008). Tästä on hyvänä esimerkkinä Carnegie Mellon Yliopistossa kehitetty Nell-ohjelmisto, joka on vuodesta 2010 alkaen oppinut yksittäisiä faktoja pelkästään lukemalla verkkosivuja (Carlson ym. 2010).

2.2 Luonnollisen kielen ymmärtämisen käsitteitä

Haverisen mukaan luonnollisen kielen sovelluksilla pyritään jonkin asteiseen kielenymmärtämiseen analyysin avulla tai kielen tuottamiseen (Haverinen 2014). Usein käytetty prosessi jakaantuu virkkeiden jakamiseen lauseiksi, saneistukseen, morfologiseen analyysiin, syntaktiseen analysiin ja semanttiseen analyysiin, jotka suoritetaan yksi toisensa perään. Ruotsalainen (2008) jakaa syntaktisen analyysin vielä lisäksi sanaluokkajäsennykseen, lauseenosien tunnistukseen ja lauseenosien riippuvuusrakenneanalyysiin. Semanttinen analyysi jakaantuu Ruotsalaisen tekstissä semanttiseen merkkaukseen ja diskurssianalyysiin. Saneistuksessa merkkijonot jaetaan yksittäisiksi sanoiksi. Morfologisessa analyysissä sanat palautetaan perusmuotoon ja niihin liitetään niiden esitysmuodon morfologinen kuvaus. Syntaktisessa analyysissä päätellään, mikä sanojen rooli lauseessa on ja semanttinen analyysi liittää sanoihin tiedon siitä mihin joukkoon tai luokkaan ne kuuluvat. Haverisen mukaan virkkeiden ja lauseiden jäsentämiseen on kaksi yleisesti käytettyä tapaa (Haverinen 2014). Ensimäinen on formaalien kielten puolelta tullut jäsennyspuu, jossa lauseet jaetaan predikaatti ja nominiryhmävälikkeiden johdoilla jäsennyspuuksi. Toinen tapa jäsentää lauseita on riippuvuusraken-

neanalyysi, jonka Haverinen (2014) määrittää siten, että siinä sanojen välinen suhde pyritään määrittämään, jotta saadaan selville, kumpi sana määrää toisen sanan merkityksen.

Ontologia on tietojenkäsittelytieteessä kuvaus sovellusalueen käsitteistä ja käsitteiden välisistä suhteista. Ruotsalaisen mukaan ontologiat voivat olla sovelluskohtaisia, aihealuekohtaisia tai korkean tason ontologioita (Ruotsalainen 2008). Ontologiahierarkia muodostuu *sanastoalkioista, synonyymeistä, luokista ja ominaisuuksista, luokkahierarkiasta, funktioista, funktiohierarkiasta, aksioomista ja säännöistä*. Hierarkian osat koostuvat *käsitteistä, ilmentymistä, relaatioista, taksonomiasta ja aksioomista.* Ontologioita luodessa käytetään edellisessä kappaleessa kuvattuja tekstin jäsennysmenetelmiä, joita soveltamalla pyritään selvittämään ovatko käsitteet *synonyymejä* eli a on b:n kaltainen, *hyperonyymejä* eli b kuuluu joukkoon a, *hyponyymejä* eli a kuuluu joukkoon b, *holonyymejä* eli a on koko joukko, *meronyymejä* eli a on b:n osa vai *antonyymejä*, jolloin a on b:n komplementti. Ontologioita luodaan asiantuntijoiden toimesta, mutta ontologian luominen on myös mahdollista automaattisesti symbolisia ja tilastollisia menetelmiä käyttäen. (Ruotsalainen 2008).

Haverinen (2014) listaa väitöskirjassaan suomen kielen online-lähteitä ja kielen analyysiin käytettävissä olevia työkaluja. Haverinen ei kuitenkaan käsittele semanttista analyysiä, ainoastaan kielen rakenteita. Luetelluista lähteistä suomen kielen perussanastoksi sopii esimerkiksi Joukahainen (Pitkänen 2015a), joka on avoimen lähdekoodin tarkoituksiin koottu suomen kielen sanasto. Joukahainen on myös oikolukuun suunnitellun Voikko-ohjelmiston ja analyysiin tarkoitetun Omorfi-ohjelmiston pohjalla (Pitkänen 2015b; Helsingin Yliopisto 2000). Synonyymien etsinnässä voi hyödyntää esimerkiksi FinnWordNet sanastoa (Helsingin Yliopisto 2015). OmorFI ja Voikko tai kaupalliset Lingsoftin FinTwol ja FinCG sisältävät morfologiseen analyysiin soveltuvat tiedot (Lingsoft Oy 2015). Semanttisen merkkaamisen tarkoituksiin voidaan tutkia erikseen ontologiatutkimuksen perusteella syntyneitä tietokantoja, kuten esimerkiksi Aalto Yliopiston SeCo-tietokannat (Aalto yliopisto 2015). Suomen kielen automaattinen generointi edellyttää, että on olemassa sanoja taivuttava ja lauseita jäsentävä komponentti. Avointa tietolähdettä, jossa olisi valmis rajapinta sanojen taivuttamiseen, ei tunnistettu. Voikko ja OmorFI sisältävät tarvittavat tiedot sanojen taivuttamista varten, mutta valmista rajapintaa sanojen taivuttamiseen morfologisen kuvauksen avulla ei niissä vielä ole. Haverinen (2014) esittelee myös riippuvuusrakenneanalyysiin soveltuvat Turku Dependency

Treebank (Haverinen ym. 2010), FinnTreeBank (Voutilainen ja Linden 2011) ja Connexor Machinese Syntax Parser (Connexor Oy 2015) ratkaisut. Kaikille esitellyille tietokannoille on yhteistä se, että ne on tarkoitettu tutkimiseen, kääntämiseen, dokumenttien prosessointiin tai faktojen etsimiseen, joten ne on erikseen sovitettava käyttöliittymäkäyttöön.

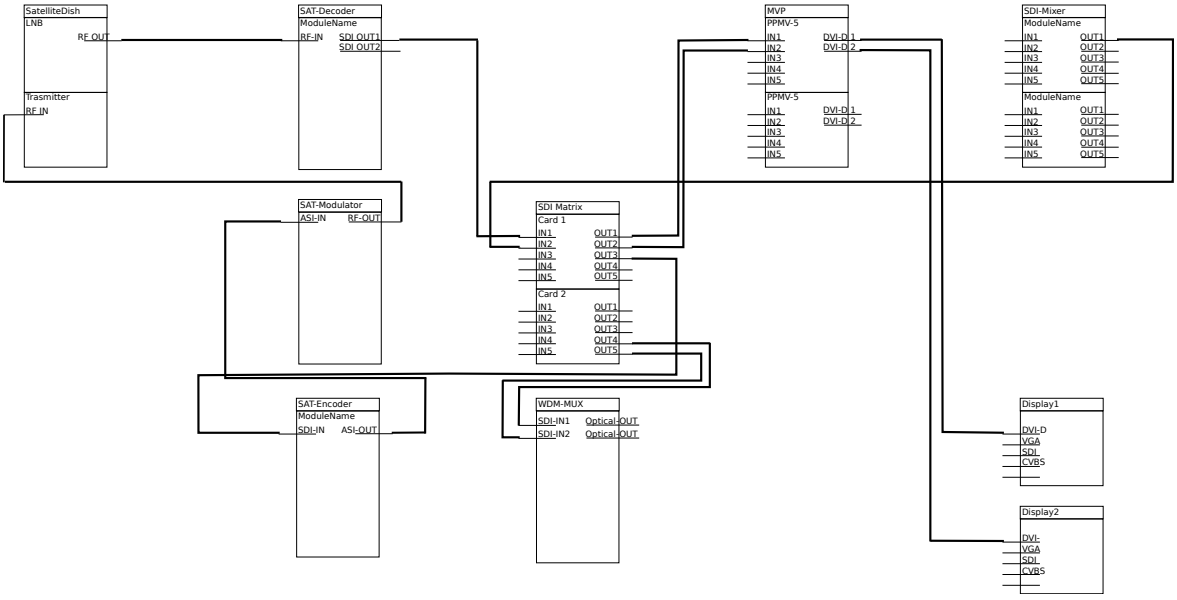
Luonnollisen kielen käyttö tietokoneen ja käyttäjän välisessä dialogissa tulee lähteissä vastaan muun muassa tekstiseikkailupelien yhteydessä. Perinteiset tekstiseikkailut perustuvat hyvin yksinkertaisiin ota, tee tai mene tyyppisiin komentoihin, joiden avulla tutkitaan huoneista koostuvaa virtuaalimaailmaa. Kehittyneempää luonnollisen kielen käyttöliittymää edustaa Facade-peli. (Mateas ja Stern 2004). Arora, Batra ja Singh (2013) jakavat dialogikäyttöliittymät graafeihin, kehyksiin ja agentteihin perustuviin. Jokinen ja McTear (2010) ja Allen ym. (1996) lisäävät listaan myös puhetoimitusmalliin perustuvat. Graafeihin tai äärellisiin automaatteihin perustuvat dialogit etenevät ennalta määrätyn kaavan mukaisesti. Esimerkkinä voidaan mainita esimerkiksi puhelinoperaattorin asiakaspalveluautomaatti. Kehyksiin perustuva malli tarkoittaa, että keskustelua varten on luotu valmiita keskustelukehyksiä ja lausemalleja, joihin diskurssianalyysi tai dialogin hallintakomponentti täydentää tietoja käyttäjältä tiedustellen tai muilta komponenteilta keräämänsä tiedon perusteella. Puhetoimitusmallissa tunnistetaan, minkä tyyppisestä lausahduksesta dialogissa on kyse. Lausahdus voi olla esimerkiksi kysymys, toteamus, komento, varmistus tai ehdotus. Älykkäät agentit ovat itsenäisiä ohjelmaprosesseja, jotka havainnoivat ympäristöä vertailevat havaintoja tavoitteisiinsa ja suorittavat toimenpiteitä (Jokinen ja McTear 2010). Agentteja on monenlaisia. Voidaan ajatella, että ne ovat kontrollimekanismejä kuten esimerkiksi säätimiä. Agenttien yhteydessä käytetään myös termiä suunnitelma tai suunnitelman tunnistaminen, jolla tarkoitetaan älykkään agentin tavoitetta. Navigaattorisovellus, joka opastaa käyttäjäänsä kohteeseen, on hyvä esimerkki älykkäästä agentista. Myös luonnollisella käyttäjällä katsotaan olevan jonkinlainen tavoite, ja älykäs dialogin hallinta pyrkii sen ymmärtämään (Jokinen ja McTear 2010). Arora, Batra ja Singh (2013) tunnistavat, että dialogikomponentin tehtäviin kuuluvat: keskusteluhistorian ylläpitäminen, dialogistrategioiden toteuttaminen, virheellisen ja viallisen tekstin käsitteleminen, tietoähdehaut, parhaan vastauksen päättäminen, aloitteet, järjestelmän vastauksen käsitteleminen, käytännöllisyys, diskurssianalyysi ja yhteisen näkemyksen ylläpitäminen. Dialogimanageri toteuttaa mainitut toimenpiteet hyödyntämällä dialogimallia, käyttäjämallia, tietämyskantaa, keskustelunhallintaa, viitteiden selvitintä ja yhteisen

näkemyksen ylläpitämiskomponenttia. Yksittäisten aliprosessien toteuttaminen on sovelluskohtaista. Tämän tutkielman osalta todetaan, että ne ovat osa keskustelukomponenttia ja voidaan toteuttaa esimerkiksi älykkäinä agentteina.

3 Tietoliikenneverkon infrastruktuurin tietomalli

Tämä luku perustuu ohjelmistoprojektin vaatimuksiin. Luvussa esitellään ohjelmistoprojektissa käytettävä tietoliikenneinfrastruktuurin tietomalli, ja sen käsittelemiseen tarvittavat funktiot. Tietomalli syntyi toimiessani järjestelmäsuunnittelijana, jolloin työnäni oli suunnitella muutoksia tietoliikenneverkkoon, jonka osat oli rakennettu usean vuosikymmenen kuluessa. Järjestelmän eri aikoina syntyneet osat olivat kukin erikseen kuvattuna kymmenissä osajärjestelmäpiirrustuksissa, tekstidokumenteissa ja niiden eri-ikäisissä versioissa. Suunnitelmien teko ja tietojen tarkistaminen oli hyvin hidasta, joten syntyi tarve saattaa oman työn kannalta olennainen tieto yhteen ajantasaiseen muotoon. Tämä tehtiin kokoamalla kaikki yksittäiset laitteiden sijainnit, liitännät ja kaapeliyhteydet yhteen mittavan listaan, josta on esimerkkinä taulukko 1, joka sisältää kuvion 1 mukaisen satelliittiläheytysyksikön laitteiden väliset yhteydet ja sijainnit.

Kuvio 1: Järjestelmäpiirustus



Järjestelmäpiirustus kuvaa järjestelmän osia ja niiden välisiä yhteyksiä. Tietoliikenneverkkoa suunnitellessa tarvitaan eri alojen asiantuntemusta. Arkkitehdit, sähköinsinöörit, LVI-ammattilaiset ja tietoverkkosuunnittelijat käyttävät kukin alansa työkaluja ja notaatioita osa-

järjestelmäsuunnitelmissaan. Käytännön suunnittelutyössä suunnitelmien ja piirustusten laatu vaihtelee ruutupaperille raapustetusta hahmotelmasta CAD-sovelluksella tehtyihin 3D-malleihin asti. Tyypillinen järjestelmätoimittajan kuvaus järjestelmästä eli tietoliikennejärjestelmäpiirustus, kuvaa järjestelmän toimiakseen vaatimat yhteydet ja laitteet, mutta ei ota kantaa niiden sijaintiin tai yhteyksien reitteihin. Tästä esimerkkinä kuvio 1, joka kuvaa yksinkertaista satelliitti-up/downlink järjestelmää.

Taulukko 1: Laitelista

paikka	rakennus	huone	laitekappi	laite	moduli	liitäntä	kaapelinippu	kaapelinumero
Pasila	NOC	Katto		Lautanen	LNB	RF-out		1
Pasila	NOC	Katto		Lautanen	Transmitter	RF-in		2
Pasila	NOC	22	LK1	SAT-DEC1		RF-IN		1
Pasila	NOC	22	LK1	SAT-MOD1		RF-out		2
Pasila	NOC	22	LK1	SAT-DEC1		SDI-OUT 1	12	2
Pasila	NOC	22	LK1	SAT-MOD1		ASI-IN	12	1
Pasila	NOC	22	LK1	SAT-ENC1		ASI-OUT	12	1
Pasila	NOC	22	LK1	SAT-ENC1		SDI-IN	10	3
Pasila	NOC	22	LK4	SDI-MTX	CARD 1	SDI-IN 1	12	2
Pasila	NOC	22	LK4	SDI-MTX	CARD 1	SDI-IN 2	12	3
Pasila	NOC	22	LK4	SDI-MTX	CARD 1	SDI-OUT 1	10	1
Pasila	NOC	22	LK4	SDI-MTX	CARD 1	SDI-OUT 2	10	2
Pasila	NOC	22	LK4	SDI-MTX	CARD 1	SDI-OUT 3	10	3
Pasila	NOC	22	LK4	SDI-MTX	CARD 2	SDI-OUT 4	11	1
Pasila	NOC	22	LK4	SDI-MTX	CARD 2	SDI-OUT 5	11	2
Pasila	NOC	18	LK1	MVP	PPMV-5	SDI-IN 1	10	1
Pasila	NOC	18	LK1	MVP	PPMV-5	SDI-IN 2	10	2
Pasila	NOC	18	LK1	MVP	PPMV-5	DVI-1		1
Pasila	NOC	18	LK1	MVP	PPMV-5	DVI-2		2
Pasila	NOC	15	Näytöt	Näyttö1		DVI-1	1	1
Pasila	NOC	15	Näytöt	Näyttö1		DVI-2	1	2
Pasila	NOC	14	Studiopöytä	Kuvamikseri		SDI-OUT1	1	2
Pasila	NOC	22	LK4	WDM-MUX	SDI-EO1	SDI-IN 1	11	1
Pasila	NOC	22	LK4	WDM-MUX	SDI-EO1	Optical Out	YLE	1
Pasila	NOC	22	LK8	WDM-MUX	SDI-EO2	SDI-in 2	11	2
Pasila	NOC	22	LK8	WDM-MUX	SDI-EO2	Optical Out	YLE	1

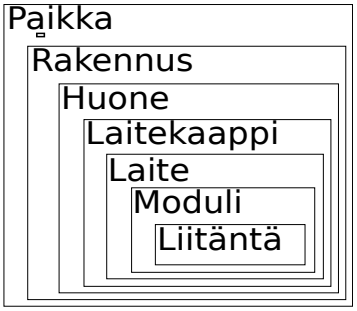
Kullakin taulukon 1rivillä kuvataan yhden laitteen fyysinen sijainti, yksi liitäntä ja numeroitu kaapeli, joka laitteeseen on kytketty. Eri tilojen väliset yhteydet on kuvattu runkokaapelilistassa, johon myös tämän taulukon umeroidut kaapelit on lueteltu.

Listan olemassolo nopeutti suunnittelutyötä merkittävästi, ja sen avulla saatiin luotua kunnollinen tilannekuva kaikista kytkennöistä. Vanhojen kytkentöjen merkinnöissä oli tyypillisesti käytetty kulloisenkin rakennusprojektin edellyttämää merkintätapaa. Listan avulla voitiin vanhat yhteydet turvallisesti uudelleen nimetä, ja kaapeleille saatiin yhtenäinen nume-

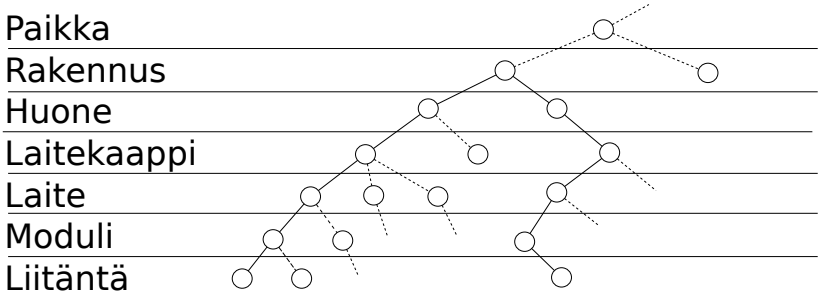
rointi. Lista auttoi myös vanhojen järjestelmän osien purkamiseen tarvittavien muutosten vaikutusten määrittämisessä. Seuraamalla kytkentöjä rekursiivisesti oli mahdollista päätellä mihin palveluihin yksittäinen muutos vaikutti, ja näin vältyttiin katkoksilta.

TV- ja radioverkon infrastruktuuri muodostuu asemista ja saiteista. Asemalla on tyypillisesti rakennuksia ja antennimasto. Saitti voi olla pelkkä antennimasto metsän keskellä tai vaikkapa rakennuksen katto, jolle on asennettu tietoliikennelaitteita. Asemien rakennuksissa on laitehuoneita, joissa on laitekaappeja, joissa on laitteita. Laitteiden välillä on kuparikaapeli-, valokaapeli- ja radiolinkkiyhteyksiä.

Kuvio 2: Säilörakenne



Kuvio 3: Puurakenne



Kuvatun kaltainen infrastruktuuri on mahdollista kuvailla sisäkkäisillä säilöillä. Ks. kuvio 2. Säilörakenne on selvästi hierarkinen eli puun kaltainen. Ks. kuvio 3. Tietotekniikassa puutietorakenne on yleisesti tunnettu. Puun oksisto ilmaistaan solmuin, joilla on viitteitä toisiin solmuihin siten, että edeltävää solmua kutsutaan vanhemmaksi ja jäljempiä solmuja lapsiksi. Puun samaan tasoon kuuluvat solmut ovat yhtä monen yhteyden päässä juurisolmusta. Esimerkin kaltaisessa säilömallissa solmun lapsi-viite tarkoittaa, että lapsielementti kuuluu tämän säilöelementin sisälle, ja puun tasot tarkoittavat erilaisia säilöelementtejä. Ks. kuviot 2 ja 3. Todellisessa maailmassa alemman tason solmujen kuvaamat elementit ovat fyysisesti ylemmän tason elementin sisällä.

Puurakenteeseen kuuluu myös solmuja, jotka eivät sisällä enää muita solmuja. Näitä kutsutaan lehdiksi. Lehdet vastaavat todellisessa infrastruktuurissa laitteiden kaapeliliitäntöjä. Laitteiden välillä voi olla kaapeliyhteyksiä, jotka on kytketty laitteiden kaapeliliitäntöihin. Kaapelin todellinen fyysinen reitti vastaa puutietomallissa kahden puun lehden välistä polkua pitkin puun oksia ja vartta. Ks. kuvio 3:n yhtenäinen viiva. Yhteyksien muodostama pol-

ku voi olla pitkä ja kulkea usean laitteen kautta, mutta se etenee kuitenkin vain yksi yhteysväli kerrallaan. Sisäkkäisyyden seurauksena puurakenteen solmujen väliset siirtymät vastaavat hyvin sitä, miten kaapelin reitti todellisuudessa kulkee. Kaapelin on ensin päädyttävä kytkennästä laitteen ja laitekaapin reunalle ja sieltä huoneen reunalle ja siitä toiseen huoneeseen, jossa edelleen siellä laitekaappiin ja jonkin laitteen liitäntään.

Laitteilla on tietomallissa myös muita attribuutteja sijainnin ja kytkentöjen lisäksi. Nämä attribuutit ovat viitteellistä tietoa jotain erityistä tarkoitusta varten. Paikka-tyyppisellä solmulla on jokin osoite, rakennuksella nimi, huoneella sijainti, laitekaapilla kapasiteetti ja mitat. Laitteella voi viitetietona olla mitat, verkkoyhteysosoite, palveluihin liittyviä tietoja ja varastotietoja ynnä muuta. Attribuutit eivät vaikuta verkon fyysiseen rakenteeseen, joten niiden käsittely jätetään tämän tutkimuksen ulkopuolelle. Infrastruktuurin tietomallin olemassaolo mahdollistaa sen, että muilla järjestelmillä on käytettävissään yksi yhtenäinen käsitys fyysisen infrastruktuurin rakenteesta, ja sen, että attribuutit on liitettävissä todelliseen fyysiseen kappaleeseen.

3.1 Tietomallin käsittelyfunktiot ja käyttöliittymän toiminnot

Edellä esitetyllä tavalla kuvattuna jokainen puun solmu kuvaa jotakin fyysistä elementtiä. Näin ollen puun muokkaustoiminnot kuvaavat käytännössä jotakin fyysistä rakennus- tai asennustoimenpidettä. Solmun lisääminen laitekaappi-solmuun tarkoittaa laitteen asentamista ja solmun poisto vastaavasti laitteen poistamista. Samaa ajattelutapaa noudattaen kaikki käyttöliittymän toimenpiteet esitetään kuin ne olisivat fyysisiä asennustoimenpiteitä joita asentaja suorittaa. Järjestelmän käyttäjä ikään kuin ohjaa kuvitteellista asentajaa toimenpide kerrallaan. Taulukossa 2 on kuvattu asennustoimien ja tietomallimuutosten vastaavuuksia.

Taulukko 2: Asennustoimenpiteet ja ohjelmalliset vastineet

Asennustoimenpide	Ohjelmallinen vastine
Rakennuksen, laitetilan, laitekaapin, laitteen, modulin tai kaapelinipun luominen	solmun luominen
Rakennuksen, laitetilan, laitekaapin, laitteen, modulin tai kaapelinipun lisääminen johonkin	solmujen välisen viitteen luominen
Rakennuksen, laitetilan, laitekaapin, laitteen, modulin tai kaapelinipun poistaminen jostain	solmujen välisen viitteen poistaminen
Kaapelin kytkeminen liitäntään	kaapeli-viitteen luominen liitäntä-tyyppiselle solmulle
Laitteiden välisen reitin hakeminen	solmujen välisen yhteisen juurisolmun etsiminen puusta.

3.2 Luonnollisen kielen toiminnot

Edellä kuvattiin, mitä tietomalli kuvaa ja millaisin toiminnoin sitä muokataan. Tässä luvussa kuvataan käyttötilanne ja tehdään havaintoja siitä, mitä suomenkielisen käyttöliittymän tulee kyetä tekemään.

Sovellus on toteutettu olio-paradigman ohjelmointikielellä, joten kaikki metodit ja parametrit ovat olioviitteitä. Puuoliolla voi olla esimerkiksi metodi ”kytke(11,22,33)”, jossa algoritmi etsii ensin oliot 11 ja 22 ja 33 lisää niihin tarvittavat viittaukset toisiinsa. Olioviitteet eivät ole selkokielisiä, minkä vuoksi sovellukseen on luotu yksinkertainen sovellusaluekieli. Sovellusaluekielessä kytke-metodi on kapseloitu siten, että olioviitteen tilalla voidaan käyttää nimiattribuuttia. Sovellusaluekielen tulkki tunnistaa avainsanan ”kytke” ja -s merkin jälkeen odottaa lähdeolion nimeä, -d merkin jälkeen odottaa kohdeolion nimeä ja -k merkin jälkeen odottaa kaapelin nimeä tai numeroa kuten esimerkiksi ”kytke -s matriisi -d dekooderi -i kaapeli 5”. Tavoitteena on, että sovellus osaa tulkita lauseesta saman tiedon, kun komento on kirjoitettu luonnollisella kiellellä.

Käyttäjä voi antaa komennon luonnollisena tekstinä muodossa ”kytke dekooderi matriisiin kaapelilla 5”, jolloin teksti sisältää komentosanan ja lähde-, kohde- ja kytkentäobjektit. Tästä voidaan johtaa, että: *”Ratkaisun on kyettävä tulkitsemaan komento ja sen parametrit tekstistä.”* Käyttäjä voi antaa komennon suomenkielisenä monessa eri muodossa esimerkiksi käyttämällä synonyymi-ilmaisua. Kytke-sanan tilalla voi käyttää esimerkiksi liitä-sanaa, jolloin

komennon merkitys on sama, mutta sanat vaihtuvat. Tästä saadaan vaatimukseksi että: *”Ratkaisun on tunnistettava synonyymit.”* Käyttäjä voi ilmaista objektien välisen suhteen eri tavoin komentamalla esimerkiksi, että ”asenna laite laitekaapin sisään” sen sijaan, että ”asenna laite laitekaappiin.” Toisin sanoen: *”Ratkaisun on tunnistettava erilaiset tavat ilmaista asioiden suhde.”* Käyttäjä voi ilmaista objektit eri järjestyksessä. Esimerkiksi ”kytke kaapeli laitteeseen” tai ”kytke laitteeseen kaapeli.” Esimerkissä sanojen järjestys on muuttunut, mutta merkitys on sama. Vaatimukseksi saadaan, että: *”Ratkaisun on tunnistettava parametrit sanajärjestyksestä riippumatta.”* Käyttäjä voi ilmaista edelliseen komentoon viitataen, että ”asenna laite siihen kaappiin.” Edellisessä komennossa on käsitelty laitekaappia, ja tässä komennossa viitataan siihen, jolloin saadaan vaatimukseksi, että: *”Ratkaisun on kyettävä käsittelemään asioita viitteellisesti aikajanalla.”* Käyttäjä voi antaa komennon, jonka suorittaminen ei ole mahdollista tai joka on moniselitteisesti tulkittavissa. Poikkeustilanteesta on annettava käyttäjälle tieto joko suomenkielisenä virheilmoituksena tai tarkentavana suomenkielisenä kysymyksenä, mistä on johdettavissa, että: *”Käyttäjän tulee saada tieto moniselitteisestä vastauksesta tai virheestä joko virheilmoituksella tai tarkentavana kysymyksenä.”*

4 Arkkitehtuurit

Tässä luvussa esitellään kirjallisuudesta tunnistettuja luonnollisen kielen käyttöliittymätoteutuksia ja mainitaan niiden arkkitehtuurin keskeiset osat. Ensin esitellään virtuaalimaailmatoteutuksia ja tiedonhakutoteutuksia, jonka jälkeen esitellään koodia generoivia toteutuksia ja viimeisenä esitellään tiedonlouhintatoteutuksia.

Terry Winograd teki vuonna 1971 ensimmäisen sovelluksen, jossa tietokonetta ohjattiin luonnollisella kielellä. SHRDLU-sovelluksessa oli uraa uurtavaa, että sovellus ymmärsi luonnollista kieltä ja myös muotoili vastaukset englanniksi. Uutta oli myös siinä käytetty englannin kielen syntaktinen malli, joka perustui ohjelmallisiin kielioppisääntöihin. Sovelluksen tekoäly pystyi päättelemään vastauksia virtuaalimaailman objektien sijaintia ja ominaisuuksia koskeviin kysymyksiin. Sen palikkamaailma, kielen malli, tulkki ja tekoäly oli toteutettu Lisp-ohjelmointikielisinä proseduureina. SHRDLU:n arkkitehtuurissa syötteelle tehtiin ensin saneistus ja sääntöihin perustuva morfologinen analyysi. Analyysin tulos toimi syötteenä joukolle kielioppirutiineita, joihin kuului muun muassa jäsennyspuu-tyyppinen jäsennin ja jotka yhdessä tulkitsivat lauseen ominaisuuksia. Erillinen semantiikkaprosessi suoritti päättelyä lauseen ominaisuuksien perusteella hyödyntämällä erillistä logiikkakäsittelijää. Ohjelman vastauksia käsitteli oma prosessinsa, jonka tehtäviin kuului keskustelun muistaminen, aikaviittausten selvittäminen ja vastausten muotoileminen englanniksi. Kielen jäsennystä suorittavien osien lisäksi ohjelmistoon kuului myös sanojen syntaktiset ja semanttiset ominaisuudet sisältävä sanastokomponentti, jossa sanat oli tallennettu perusmuodossa. Sanoja käyttävät prosessit osasivat taivuttaa tai palauttaa sanat perusmuotoon tarpeen mukaan. Erityinen semanttisia ominaisuuksia sisältävä assosioiva verkko oli esitetty ominaisuuslistana. Sanoja ja niiden merkitystä käsittelevien komponenttien lisäksi sovellus sisälsi virtuaalimaailman objektien esittämisen ja liikuttamisen toiminnot. (Winograd 1971).

Saksankielinen HAM on Winogradin SHRLDU:n kaltainen luonnollisen kielen dialogijärjestelmä, jossa käsitellään virtuaalimaailman objekteja. Kuten SHRDLU, myös HAM on toteutettu Lisp-ohjelmointikielellä. Sovelluksen arkkitehtuurissa kielen ymmärtäminen etenee kuten standardissa kielen jäsennysprosessissa, mutta jokaiseen päävaiheeseen on lisätty mukaan yhteistoiminta käyttäjän kanssa ja jo saneistuksen yhteydessä on mahdollisuus selven-

tävään dialogiin käyttäjän kanssa. Toteutuksen suurin ero SHRLDU:uun verrattuna on tietämyskantojen laajempi hyödyntäminen prosessissa. Tietämys, kieli ja kielioppi on HAM:ssa ilmaistu Fuzzy- tekoälykielellä, jossa tietämys talletetaan muuttuja-arvo pareina. Fuzzy-kieli tarjoaa mahdollisuuden loogiseen päättelyyn SHRDLU:ssa käytetyn PLANNER-komponentin kaltaisesti. Tieto kielestä, ympäristöstä ja käsitteiden välisistä suhteista on koottu joukkoon tietämyskantoja, joita monivaiheinen tulkinta- ja keskusteluprosessi käyttää. SHRDLU:uun verrattuna saksankielinen HAM hallitsee myös moniselitteisen syötteen käsittelyn. Moniselitteisyys ratkotaan päättelemällä tietämyskannan tietojen perusteella mihin käyttäjä viittaa tai selventämällä moniselitteinen asia dialogilla käyttäjän kanssa. (Hahn ym. 1980).

Japaninkielisessä K2-sovelluksessa käyttäjä voi virtuaalisen agentin välityksellä käsitellä huoneessa olevia esineitä. Virtuaalinen agentti on hahmo, jolle voi antaa komentoja puhumalla japaniksi. Sovelluksen arkkitehtuurissa puhe syötetään puheentunnistimeen, joka tunnistaa lausutun sanajonon hyödyntämällä kielimallia ja sanastoa. Sanajono toimii syötteenä syntaktiselle ja semanttiselle analyysille, jotka hyödyntävät semanttista sanastoa. Semanttisella ja syntaktisella informaatiolla annotoitu tapauskehys välitetään diskurssianalyysiin. Diskurssianalyysi hyödyntää ontologiatietoa, keskusteluhistoriaa ja suunnitelmakirjastoa. Suunnitelmakirjaston avulla pyrittään määrittämään mitä käyttäjä haluaa virtuaalisen agentin maailmassa tekevän. K2:ssa toteutettu kielen ymmärrysprosessi hyödyntää tietämyskantoja samaan tyyliin kuin aiemmin esitelty HAM. Edellisiin esimerkkeihin verrattuna K2:n uusi piirre on sen kyky käsitellä epäselvää puhetta ja moniselitteisiä, puutteellisia tai viittellisiä lauseita. Edellisistä esimerkeistä poiketen virtuaalimaailmaa käsittelevä logiikka on kolmiulotteista. (Tokugana, Funakoshi ja Tanaka 2004).

Edellä esiteltiin virtuaalimaailmasovelluksia. Toinen tyypillinen luonnollisen kielen käyttötapaus on tiedon hakeminen. Tyypillisessä hakusovelluksessa hakusanoille etsitään vastineita synonyymien, ontologian tai hakusanoihin eri lähteissä yhdistettyjen verbien avulla ja tulokset pisteytetään. Haku suoritetaan laajennetun hakusanalistan avulla. Luonnollisen kielen tietokantakäyttöliittymissä lähtökohtana on, että käyttäjä esittää kysymyksen tai joukon kohteen määritteleviä hakusanoja. Syötteestä tunnistetaan siinä esitettyjen käsitteiden suhteet ja pyritään määrittämään tuntematon tekijä. Kun on päätelty, mihin joukkoon vastaus kuuluu ja minkälaisesta yhteydestä on kyse, suoritetaan tietokantahaku ja valitaan vastauksista sopivin.

QuestIO on luonnollisen kielen kyselykielitoteutus. QuestIO:n kielenjäsennys alkaa kysymystekstin morfologisella analyysillä, jonka tulos annotoidaan ontologiatiedolla. Morfologisella analyysillä etsitään sanan perusmuoto ja perusmuodolla hakien ontologiatiedosta saadaan sanalle joukko tai luokka johon se kuuluu. Tuloksista kootaan joukko kandidaattitulkintoja, jotka pisteytetään. Parhaan tuloksen perusteella generoidaan SQL-kysely tietokantahakua varten (Tablan, Damljanovic ja Bontcheva 2008). Querix on myös kyselytoteutus kuten QuestIO. Erona on, että kun sanat on saatu eroteltua, tehdään haun pisteyttäminen synonyymien perusteella ja pisteytetään syntynet lauseyhteydet. Tuloksen perusteella luodaan kysely (Kaufmann, Bernstein ja Zumstein 2006). Precise tietokantahakujärjestelmä poikkeaa edellisistä. Siinä hakulauseesta luodaan graafi, sanoille haetaan synonyymivastineita, joille graafin kanssa luodaan verbi-nomini pareja. Verbi-nomini pareille etsitään vastaavuuksia, pisteytetään vastaavuudet ja parhaan tuloksen perusteella generoidaan haku (Popescu, Etzioni ja Kautz 2003). Stephen Wolframin Wolfram Alpha-hakukone tuottaa symbollisen esityksen kysymyslauseesta ja symbolisen esityksen avulla pyrkii tietämyskantojen avulla ratkaisemaan vastauksen. Wolfram Alphaa verrataan IBM:n Watson tekoälyyn, joka voitti ihmispelaajat Jeopardy-tietokilpailupelissä. Wolfram Alpha käyttää hyödykseen kielen ymmärtämisen tekniikoita ja symbolista laskentaa ja Watson analysoi tietolähteiden tekstiä avainsanojen avulla tilastollisesti (Wolfram 2011).

Trains on luonnollisen kielen dialogisovellus, jossa käyttäjä voi suunnitella junamatkansa USA:n itärannikon junareiteillä keskustelemalla järjestelmän kanssa. Trains on hakusovellus, mutta poikkeaa edellisistä siten, että siinä käyttäjän ja järjestelmän välillä käydään dialogia matkakohteesta ja reitistä. Virtuaalimaailmaesimerkkien dialogeissa on käytetty kehysperiaatetta, mutta Trainsissa keskustelun tulkitseminen tapahtuu puhetoimitus-tyyppisesti. Parseri vastaanottaa tulkittua puhetta tai tekstiä syöttölaitteelta ja tuottaa puhetoimitustulkintoja. Puhetoimitustulkinnat syötetään diskurssin hallintakomponentille, joka alitoimintoina käsittelee viitetiedot, tulkitsee toiminnot ja käyttää varsinaista taustajärjestelmää. Taustajärjestelmä suorittaa junareittejä ja aikatauluja koskevan ongelmanratkaisun ja sovellusaluepäättelyn. Diskurssinhallintakomponentin suunnittelema dialogi palautetaan puhetta generoivalle komponentille, joka rakentaa tiedoista virkkeen puhesyntetisaattorille ja tekstin näytölle. Erona aiempiin esimerkkeihin on, että käyttäjän syöte kuvaillaan ensin puhetoimituksena ja vasta tarvittaessa tulkitaan se syntaktisesti tai semanttisesti. Käyttäjälle vastaaminen

on toteutettu kehysperiaatteella. (Allen ym. 1996).

Luonnollisen kielen ymmärtämistä hyödynnetään myös tiedonlouhinnassa. Tiedonlouhinnalla pyritään koneellisesti tekstiä tutkimalla luomaan faktoja. Nell järjestelmä on vuodesta 2010 alkaen lukenut internetin verkkosivuja ja päättelee ja luo yksinkertaisia a kuuluu joukkoon b tyyppisiä faktoja ilmentäen jonkin asteista ontologian oppimiskykyä (Carlson ym. 2010). Knowledge Vault-tietokantaan on tallennettu jo 2.8 miljardia faktaa, jotka järjestelmä on kerännyt tilastollisin menetelmin verkkosivuja seuraamalla (Dong ym. 2014).

Luonnollisen kielen käyttöä ohjelmakoodin tuottamisessa on sovellettu Metafor-sovelluksessa, joka englannin kielisen tekstin sisällön perusteella osaa tunnistaa asioita luokiksi ja asioita koskevat tekemiset metodeiksi. Sen arkkitehtuuri koostuu luonnollisen kielen parserista, tietämyskannasta, ohjelmatulkista, sisäisestä kielestä, koodin vedostajasta, kuvailijasta, dialogista ja käyttöliittymästä (Liu ja Lieberman 2005). Vadasin ja muiden kokoama luonnollisen kielen ohjelmointiympäristö on samankaltainen kuin Metafor, mutta luokkien ja metodien lisäksi sovellus osaa muodostaa tekstin perusteella myös algoritmihahmotelmia. Sen arkkitehtuuri on kuvattu vaatimatimattomasti sisältäen yleiset syntaktisen ja semanttisen analyysiin sekä komponentin, joka tunnistaa funktiot tekstistä. (Vadas ja Curran 2005).

Luonnollisen kielen ymmärtämisprosessi on edellä mainituissa esimerkeissä karkeasti ottaen sama. Erot löytyvät siitä, miten prosessin eri osat on toteutettu. Winogradin SHRDLU:ssa tietämys- ja ohjelmalogiikka on esitetty proseduraalisesti. SHRDLU:ta tuoreemmissa sovelluksissa käytetään asteittain enemmän erillisiä tietämyskantoja ja vähemmän ohjelmoitua tietoa. Tietämyskantojen tehtävänä on tarjota esimerkiksi sanojen morfologisen muodon kuvaus, synonyymit ja semanttinen tieto määrämuotoisena datana. Hakusovelluksissa synonyymi- ja ontologiatietämyskannat tarjoavat vaihtoehtoisia tulkintoja käyttäjän hakutekstille, ja muissa kuin hakusovelluksissa ne mahdollistavat synonyymien käytön käyttäjän komennoissa. Lauserakennejäsentelyssä jäsennyspuu on yleisempi kuin riippuvuusrakenneanalyysi. Käyttäjän ja sovelluksen välinen interaktion määrä vaihtelee käyttötilanteen mukaisesti; hakusovelluksissa interaktio on minimaalista, mutta virtuaalimaailmasovelluksissa jatkuvaa.

Yhteenvetona todetaan, että kaikissa esitellyissä esimerkeissä on toteutettu jonkinlainen luon-

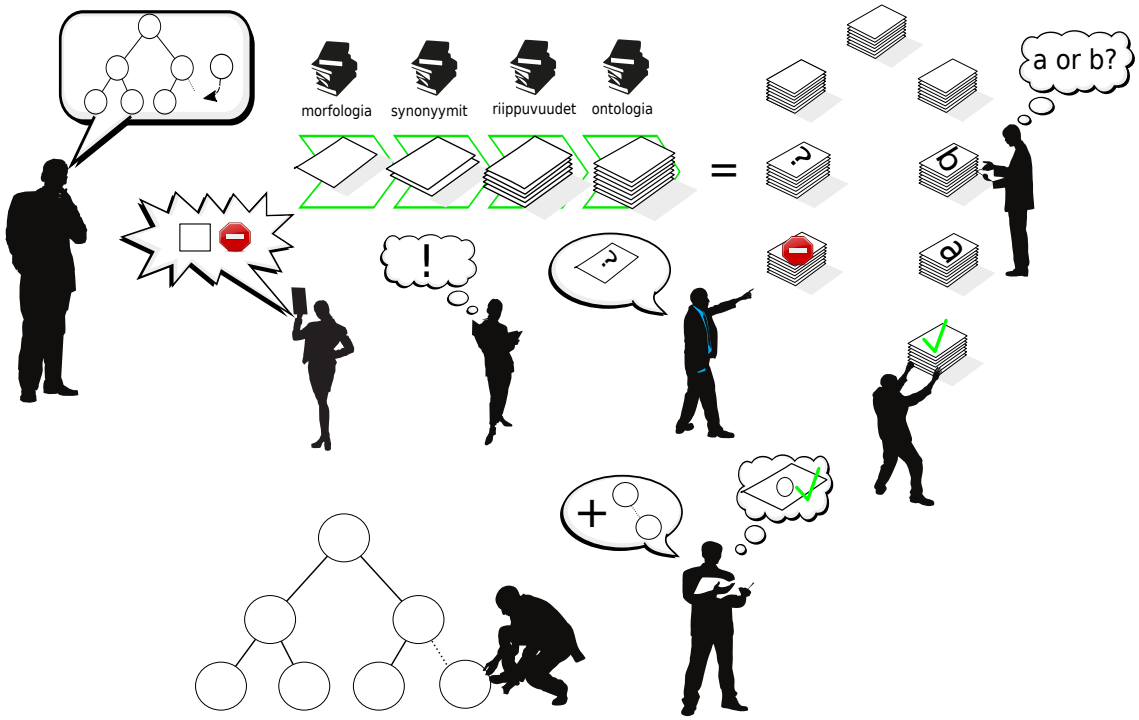
nollisen kielen ymmärtämisen komponentti, joka suorittaa kielen jäsentämisen. Sen lisäksi toteutukset sisältävät diskurssianalyysin tai dialoginhallinnan, joka hoitaa luonnollisen käyttäjän ja sovellusaluekomponentin välisen keskustelun. Lopuksi kaikissa esimerkeissä on jokin sovelluskomponentti, joka suorittaa sovellusaluekohtaiset toimenpiteet.

5 Tunnistetut komponentit

Tässä luvussa luvun 3 vaatimusmäärittelyyn yhdistetään se, mitä luvun 4 lopuksi todettiin luonnollisen kielen käyttöliittymäkomponenteista. Sovelluksessa arkikielinen komento halutaan yhdistää tietomallia muokkaavaan funktioon, ja komentolauseesta halutaan tunnistaa funktion tarvitsemat parametrit.

Luvun 4 yhteenvedossa todettiin, että luonnollisen kielen dialogikäyttöliittymä vaatii kolme pääkomponenttia: luonnollisen kielen tulkin, dialogin hallintakomponentin ja sovellusaluekohtaisen komponentin. Arkkitehtuurien perusteella luonnollisen kielen tulkin päätehtävä on analysoida käyttäjän syöte, tuottaa siitä mahdollisimman hyvä analyysi ja koota viitetietoja. Modernit tulkkitoteutukset hyödyntävät erillisiä tietämyskantoja lauseiden jäsentämiseen. Tietämyskannat palauttavat syötesanalle listan Avain – arvo-pareja, jotka kuvaavat lauseen kielellistä muotoa ja merkitystä. Avain – arvo-joukko toimii syötteenä dialogin hallintakomponentille, joka avain – arvo-joukon perusteella päättelee, mitä sen kuuluu tehdä. Sovellusaluekohtaisen komponentin funktiot voidaan esittää vastaavalla tavalla kielellisten ominaisuuksien joukkona, joihin dialogin hallintakomponentti syötteen kuvausjoukkoa vertailee. Mikäli joukot täsmäävät tai täydentävät toisiaan, voi dialogin hallinta kutsua sovelluksen funktiota. Esimerkkinä käytetty kytke-funktio edellyttää kolmen sanan syötettä, joista ensimäinen on kytkeä-sanan yksikön toisen persoonan imperatiivimuoto, ja seuraavat sanat ovat lähdettä ja kohdetta ilmaisevissa muodoissa. Mikäli nämä tiedot ilmenevät syötelauseesta, voi dialogin hallintakomponentti kutsua funktiota. Erilaiset älykkäät agentit voivat vastaavalla tavalla poimia kuvausjoukosta niitä kiinnostavia tietoja. Keskusteluhistoriaa käsittelevä agentti voi esimerkiksi seurata keskusteluhistoriassa lauseen objekteja ja päätellä viittaako lauseen pronomini edellisen komennon objektiin. Useissa luvun 4 esimerkkitoteutuksissa on erillinen logiikkaprosessi, joka suorittaa loogista päättelyä. Päättely edellyttää, että on olemassa joukko määrityksiä, jotka on mahdollista testata. Dialogin hallintakomponentti voi päättelyllä esimerkiksi täydentää epätäydellisiä syötelauseita tai määrittää, mitä sen kuuluu käyttäjältä kysyä. Kuviossa 4 on sarjakuvamaisesti kuvattu edellä käsitelty luonnollisen kielen ymmärtämisprosessi, viitetietolähteet, dialogin hallinta, älykkäät agentit ja tietomallin muokkaus.

Kuvio 4: Luonnollisen kielen dialogikäyttöliittymä



Seuraavaksi pohditaan, mitä edellä kuvatun prosessin osia on toteutettava, jotta luvun 3 vaatimusmäärittelyyn voidaan vastata. Ensimäisenä vaatimuksena oli, että: *"Ratkaisun on kyettävä tulkitsemaan komento ja parametrit tekstistä."* Lauseen ja sen sanojen muoto määritellään, ja muodosta saatua kuvausta vertaillaan sovellusaluekomponentin funktioista annettuihin kuvauksiin. Mikäli kuvaukset täsmäävät hyvin johonkin funktioon, voi dialoginhallintakomponentti kutsua funktiota. Sanojen muodon selvittäminen edellyttää tässä tapauksessa ainakin saneistusta ja morfologista analyysiä. Morfologiseen analyysiin voidaan käyttää Voikko- tai OmorFi-tietokantoja, lisäksi dialogin hallintakomponentin on kyettävä suorittamaan syötteen kuvauksen ja funktioista annetun muotokuvauksen vertailu eli avain – arvojoukkojen vertailu. Seuraavana vaatimuksena oli, että: *"Ratkaisun on kyettävä tunnistamaan synonyymit."* Lauseen sanoille voidaan hakea synonyymejä saneistuksen jälkeen tai morfologisen analyysin tuloksen perusteella, kun sanan perusmuoto on selvitetty. Tietokantahaku palauttaa joukon synonyymisanoja, joita dialogin hallintakomponentti voi testata syötesanan tilalla sovelluksen funktioiden määrityksiä vasten. Suomen kielen sanakirjat, esimer-

kiksi FinnWordNet, tarjoavat synonyymitietoja. *"Ratkaisun on tunnistettava erilaiset tavat ilmaista asioiden suhde."* Yksi sana voi vastata useamman sanan ilmaisua, joten on tutkittava lauseen sanojen välisiä suhteita ja selvitettävä vastaavatko ne jotakin ilmaisua. Teoriassa riippuvuusrakenneanalyysi sisältää työkalut tähän. FinnTreeBank- ja Turku Dependency TreeBank-tietokantojen sopivuutta tähän tarkoitukseen voidaan tutkia. Tässä tapauksessa sanat voivat olla sovellusaluekohtaisia nimisanoja, joten ulkoinen lähde ei välttämättä ole se oikea ratkaisu, vaan on etsittävä ohjelmallista keinoa selvittää sanojen välinen suhde. *"Ratkaisun on tunnistettava parametrit sanajärjestyksestä riippumatta."* Edellä mainittuja riippuvuusrakenneanalyysin työkaluja voidaan tutkia tähänkin käyttöön. Ongelmat ovat samat kuin edellä ja valmista ratkaisua ei ole. *"Ratkaisun on kyettävä aikajanalla käsittelemään asioita viitteellisesti."* Historia- ja keskustelutiedon ylläpitäminen katsotaan osaksi diskurssianalyysiä, mutta komponenttina historian ylläpitäminen katsotaan osaksi dialogin hallintaa (Jurafsky ja Martin 2009). Luvun 4 perusteella dialogimanagerin toiminta on sovelluskohtaista, eikä toteutettuja ratkaisuja voi suoraan sovittaa tähän tarkoitukseen. *"Käyttäjän tulee saada tieto moniselitteisestä vastauksesta tai virheestä, joko virheilmoituksella tai tarkentavana kysymyksenä."* Hyvää ohjelmointitapaa noudattaen kunkin ohjelmistokomponentin ja aliprosessin tehtäviin kuuluu sitä koskevien poikkeustilanteiden käsittely. Tilanteessa, jossa sovelluskomponentti tai agentti odottaa jotain parametria, ja sitä ei tekstissä tule, voidaan asia joko ilmaista käyttäjälle tai pyrkiä etenemään ilman puuttuvaa parametria. Tutkimuksen perusteella dialoginhallintakomponentin rooliin kuuluu tällaisten viestien käsitteleminen ja muotoilu käyttäjälle. Vastauksen muotoilu suomen kielellä vaatii dialogimallista riippumatta lauseen sanojen muotoiluun kykenevän komponentin. Valmista komponenttia tähän ei ole, mutta Haverisen mukaan tekstin generointia tutkitaan nyt, kun suomenkielen analyysiin on käytettävissä avoimen lähdekoodin sovelluksia ja tietokantoja (Haverinen 2014).

6 Yhteenveto

Tutkimuksessa selvitettiin kirjallisuuden avulla, mitä komponentteja tarvitaan suomenkielisen luonnollisen kielen käyttöliittymän toteuttamiseen tietoliikenneinfrastrukuurin tietomallin käsittelyä varten. Työssä tunnistettiin yleiset kielen jäsentämiseen ja dialogiin tarvittavat ohjelmistokomponentit, ja lisäksi selvitettiin, mitä suomen kielen viitetietokantoja kielen jäsentämisprosessin tueksi on saatavilla. Tämän jälkeen pohdittiin, mitä prosessin osia on toteutettava, jotta ohjelmistoprojektin vaatimuksiin voidaan vastata. Lopuksi todettiin, että ohjelmiston vaatimuksiin voidaan vastata toteuttamalla osia luonnollisen kielen jäsentämisprosessista ja rajoittamalla kieli komentoihin. Kaikkiin prosessin osiin ei ole saatavilla valmiita käyttöliittymäkäyttöön soveltuvia tietolähteitä. Tietolähteiden käyttöönotto edellyttää niiden sovittamista tähän nimenomaiseen tarkoitukseen. Lisätutkimusta tarvitaan erityisesti automaattisesta suomen kielen generoinnista, sanojen taivuttamisesta ja riippuvuusrakenneanalyysistä.

Lähteet

Aalto yliopisto. 2015. “SeCo”. Viitattu 23. maaliskuuta 2015. http://seco.tkk.fi.

Allen, James F., Bradford W. Miller, Eric K. Ringger ja Teresa Sikorski. 1996. “A Robust System for Natural Spoken Dialogue”. Teoksessa *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics,* 62–70. ACL ’96. Santa Cruz, California: Association for Computational Linguistics. doi:10.3115/981863.981872. http://dx.doi.org/10.3115/981863.981872.

Arora, Suket, Kamaljeet Batra ja Barabjit Singh. 2013. “Dialogue System: A Brief Review”. *CoRR* abs/1306.4134. http://arxiv.org/abs/1306.4134.

Carlson, Andrew, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr. ja Tom M. Mitchell. 2010. “Toward an Architecture for Never-Ending Language Learning”. Teoksessa *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010).*

Chomsky, N. 2002. *Syntactic Structures.* Mouton classic. Bod Third Party Titles. ISBN: 9783110172799. http://www.google.fi/books?id=a6a%5C_b-CXYAkC.

Connexor Oy. 2015. Viitattu 23. maaliskuuta 2015. http://www.connexor.fi.

Dijkstra, E. 1977. “On the foolishness of "natural language programming"”. Viitattu 23. maaliskuuta 2015. http://www.cs.utexas.edu/users/EWD/transcriptions/EWD06xx/EWD667.html.

Dong, Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun ja Wei Zhang. 2014. “Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion”. Teoksessa *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 601–610. KDD ’14. New York, New York, USA: ACM. ISBN: 978-1-4503-2956-9. doi:10.1145/2623330.2623623. http://doi.acm.org/10.1145/2623330.2623623.

Hahn, W., W. Hoeppner, A. Jameson ja W. Wahlster. 1980. “The Anatomy of the Natural Language Dialogue System HAM-RPM”. Teoksessa *Natural Language Based Computer Systems,* toimittanut L. Bolc. Munchen: Hanser/Macmillan.

Haverinen, Katri. 2014. “Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains”. Tohtorinväitöskirja.

Haverinen, Katri, Timo Viljanen, Laippala Veronika, Samuel Kohonen, Filip Ginter ja Tapio Salakoski. 2010. “Treebanking Finnish”. Teoksessa *In Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories,* 79–90. Turku Centre for Computer Science.

Helsingin Yliopisto. 2000. “Omorfi”. Viitattu 23. maaliskuuta 2015. `http://www.ling.helsinki.fi/kieliteknologia/tutkimus/omor/`.

———. 2015. “FinnWordNet”. Viitattu 23. maaliskuuta 2015. `http://www.ling.helsinki.fi/kieliteknologia/tutkimus/finnwordnet/`.

Jokinen, K., ja M. McTear. 2010. *Spoken Dialogue Systems.* Synthesis lectures on human language technologies. Morgan & Claypool Publishers. ISBN: 9781598295993. `http://books.google.fi/books?id=ualwulnD020C`.

Jurafsky, D., ja J.H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall series in artificial intelligence. Pearson Prentice Hall. ISBN: 9780131873216. `https://books.google.fi/books?id=fZmj5UNK8AQC`.

Kaufmann, Esther, Abraham Bernstein ja Renato Zumstein. 2006. “Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs”. Teoksessa *In: 5th ISWC,* 980–981. Springer.

Lingsoft Oy. 2015. “Lingsoft Oy”. Viitattu 23. maaliskuuta 2015. `http://www.lingsoft.fi`.

Liu, Hugo, ja Henry Lieberman. 2005. “Metafor: Visualizing Stories As Code”. Teoksessa *Proceedings of the 10th International Conference on Intelligent User Interfaces,* 305–307. IUI ’05. San Diego, California, USA: ACM. ISBN: 1-58113-894-6. doi:10.1145/1040830.1040908. http://doi.acm.org/10.1145/1040830.1040908.

Mateas, Michael, ja Andrew Stern. 2004. “Natural Language Understanding in Façade: Surface-Text Processing” [kielellä English]. Teoksessa *Technologies for Interactive Digital Storytelling and Entertainment,* toimittanut Stefan Göbel, Ulrike Spierling, Anja Hoffmann, Ido Iurgel, Oliver Schneider, Johanna Dechau ja Axel Feix, 3105:3–13. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-22283-5. doi:10.1007/978-3-540-27797-2_2. http://dx.doi.org/10.1007/978-3-540-27797-2_2.

Pitkänen, H. 2015a. “Omorfi”. Viitattu 23. maaliskuuta 2015. http://joukahainen.puimula.org.

———. 2015b. “Voikko”. Viitattu 23. maaliskuuta 2015. http://voikko.puimula.org.

Popescu, Ana-Maria, Oren Etzioni ja Henry Kautz. 2003. “Towards a Theory of Natural Language Interfaces to Databases”. Teoksessa *Proceedings of the 8th International Conference on Intelligent User Interfaces,* 149–157. IUI ’03. Miami, Florida, USA: ACM. ISBN: 1-58113-586-6. doi:10.1145/604045.604070. http://doi.acm.org/10.1145/604045.604070.

Ruotsalainen, N. 2008. “Ontologoioiden oppiminen tekstistä”. Tutkielma.

Tablan, Valentin, Danica Damljanovic ja Kalina Bontcheva. 2008. “A Natural Language Query Interface to Structured Information”. Teoksessa *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications,* 361–375. ESWC’08. Tenerife, Canary Islands, Spain: Springer-Verlag. ISBN: 3-540-68233-3, 978-3-540-68233-2. http://dl.acm.org/citation.cfm?id=1789394.1789430.

Tokugana, Takenobu, Kotaro Funakoshi ja Hozumi Tanaka. 2004. “K2: Animated Agents that Understand Speech Commands and Perform Actions” [kielellä English]. Teoksessa *PRICAI 2004: Trends in Artificial Intelligence,* toimittanut Chengqi Zhang, Hans W. Guesgen ja Wai-Kiang Yeap, 3157:635–643. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-22817-2. doi:10.1007/978-3-540-28633-2_67. http://dx.doi.org/10.1007/978-3-540-28633-2_67.

Vadas, David, ja R. James Curran. 2005. “Programming With Unrestricted Natural Language”. Teoksessa *Proceedings of the Australasian Language Technology Workshop 2005,* 191–199. Sydney, Australia. http://aclweb.org/anthology/U05-1027.

Winograd, T. 1971. *Procedures As A Presentation For Data In A Computer Program For Understanding Natural Language.* Tekninen raportti. Stanford.

———. 1972. *Understanding Natural Language.* 1. painos. New York: Academic Press.

Wolfram, S. 2011. “jeopardy-ibm-wolframalpha”. Viitattu 23. maaliskuuta 2015. http://blog.stephenwolfram.com/2011/01/jeopardy-ibm-and-wolframalpha/.

Voutilainen, A., ja K. Linden. 2011. “Specifying a linguistic representation with a grammar definition corpus”. Teoksessa *In Proceedings of Corpus Linguistics.*