Teollista Internetiä työkoneissa – palveluarkkitehtuuri suorituskyvyn optimointiin

Petri Kannisto, David Hästbacka, Seppo Kuikka
Tampereen teknillinen yliopisto, Systeemitekniikan laitos
PL 692, 33101 TAMPERE
Tel. (03) 3115 11, telefax (03) 3115 2340
{petri.kannisto|david.hastbacka|seppo.kuikka}@tut.fi, http://www.tut.fi/ase

AVAINSANAT hajautettu palveluarkkitehtuuri, sääntöpohjainen päättely, tiedonhallinta

TIIVISTELMÄ

Automaatiota on perinteisesti käytetty tuottavuuden lisäämiseen. Aina kyse ei ole pelkästään laitetason toiminnoista, vaan sopivilla tietojärjestelmäratkaisuilla tuottavuuslisäyksiä saadaan aikaan parantamalla tiedonhallintaa. Modernit liikkuvat työkoneet sisältävät monimutkaisia alijärjestelmiä, joiden toimintaa voidaan optimoida erilaisten käyttöasetusten arvoja muuttamalla. Käyttöasetuksia koskeva tietämys ei kuitenkaan ole helppoa hallita, sillä sopiviin käyttöasetusten arvoihin voivat vaikuttaa esimerkiksi olosuhteet, tehtävän työn tyyppi tai kuljettajan taitotaso. Tietämystä käyttöasetuksista saadaan lisää soveltamalla tilastollisia menetelmiä, mutta se pitäisi voida myös varastoida johonkin. Kaikkein parasta olisi, jos tietämystä voitaisiin soveltaa antamalla kuljettajalle ehdotuksia käyttöasetusten arvojen muuttamisesta – työkoneessa tulisi olla järjestelmä, joka tarkkailisi koneen suorituskykyä ja käyttöasetuksia sekä analysoisi niitä käytön aikana. Tämä artikkeli keskittyy erityisesti tietämyksen hallintaan ja koneiden käytön aikaiseen hyödyntämiseen vaadittavaan tietojärjestelmäarkkitehtuuriin. Järjestelmä on pakostikin hajautettu, sillä dataa tuottavat koneet ovat hajallaan ympäri maailmaa, mutta toisaalta dataa ja tietämystä on voitava käsitellä samalla kerralla keskitetysti. Toisaalta tietämys pitää edelleen saada käyttöön eri koneyksilöihin. Haaste on hallittavissa syvällisellä vaatimusten ymmärtämisellä sekä huolellisella suunnittelutyöllä. Toteutustekniikoiksi riittävät suurelta osin Internetissä jo käytössä olevat tekniikat. Tämä artikkeli käy ensiksi läpi vaatimukset ja suunnitteluvalinnat, minkä jälkeen esitellään tarkoitukseen tehty prototyyppi.

1 JOHDANTO

Modernien liikkuvien työkoneiden korkeasta automaatioasteesta huolimatta on edelleen mahdollista lisätä tuottavuutta jo olemassa olevia komponentteja hyödyntämällä. Koneissa on lukuisia käyttöasetuksia, joilla niiden toimintaan ja suorituskykyyn voidaan vaikuttaa. Jos tiedetään kuhunkin käyttötilanteeseen parhaiten sopivat käyttöasetukset, sillä voi olla tuottavuuteen suuri vaikutus. Käyttöasetukset voivat vaikuttaa esimerkiksi toimilaitteiden herkkyyteen tai toimintanopeuteen – liian suuri herkkyys tai nopeus hankaloittaa käsittelyä, liian alhainen arvo taas hidastaa työtä turhaan. Koneet ovat kuitenkin monimutkaisia, eikä parhaiden käyttöasetusten löytäminen ei ole helppoa. Toisaalta tuottavuuteen vaikuttaa myös koneen kuljettajan toiminta. Taitavimmat kuljettajat osaavat ajaa konetta hyvin, mutta erityisesti uusille käyttäjille on eduksi, mikäli koneen käyttämiseen on saatavilla vinkkejä – tällöin he oppivat nopeammin. Edelleen myös kulujen vähentäminen on olennainen keino parantaa tuottavuutta. Oikea-aikaisen ja oikeanlaisen huollon merkitys on suuri, kun halutaan vähentää toisaalta tarpeettomista huoltotoimenpiteistä aiheutuvia kuluja ja toisaalta huollon puutteesta johtuvaa suorituskyvyn laskua tai jopa konerikkoja. Joskus huollon tarve voidaan päätellä koneen toiminnasta, jolloin huolto voidaan kohdentaa juuri oikein.

Vaikka suorituskykytavoitteet ja diagnostiikka ovat ajatuksen tasolla suoraviivaisia asioita, niihin liittyvän tietämyksen hallinta ja soveltaminen koneellisesti eivät ole triviaaleja päämääriä. Yksittäisen koneen toimintaa on voitu mitata jo pitkään – modernissa työkoneessa on lukemattomia antureita eri tarkoituksiin. Jotta mitattujen arvojen potentiaali voidaan hyödyntää maksimaalisesti, tarvitaan kuitenkin tietämystä, joka voidaan saavuttaa vain tarkastelemalla suurta konejoukkoa. Konejoukkoa tarkastelemalla saadaan tietoa siitä, minkälaisilla käyttöasetuksilla konetyyppi on toiminut tietyissä olosuhteissa parhaiten ja millä tavoin konetta tulisi ajaa. Samoin diagnostiikkaan voidaan hyödyntää tietoa siitä, minkälaiset oireet koneessa ovat tyypillisesti ennakoineet huollon tarvetta tai vikaa.

Uusi haaste syntyy siitä, miten hallita suureen konejoukkoon liittyvää tietämystä. Mihin ja miten tieto pitäisi tallentaa? Miten tieto tulisi jäsentää? Entä kuinka analysoida ja tulkita kerättyä tietoa parhaiten? Kun lopulta on

kerätty dataa ja saatu siitä päätelmiä analyysien tuloksena, nekin on voitava tallentaa. Edelleen olisi parasta, jos tätä uutta tietämystä voitaisiin hyödyntää koneissa niiden käytön aikana siten, että koneen toiminnasta saataisiin jatkuvaa palautetta. Toisaalta tiedon kerääminen ja jalostaminen on iteratiivinen prosessi, jossa tietämys jalostuu jatkuvasti uutta dataa kerättäessä. Miten tämä tietämys voitaisiin välittää koneyksilöihin, jotta koneen kuljettaja saisi paikallisesti ja jatkuva-aikaisesti palautetta koneen toiminnasta? Entä kun tietämys paranee uusien data-analyysien myötä – miten uusi tietämys päivitetään koneyksilöihin? Teollinen Internet antaa välineet näiden haasteiden voittamiseen, mutta välineiden oikeanlainen soveltaminen on olennaista onnistumisen kannalta. Ratkaisu on soveltaa sopivia palvelukomponentteja järjestelmäarkkitehtuurin kannalta järkevästi eri tarpeisiin. Tämä artikkeli keskittyy siihen, millainen tietojärjestelmäarkkitehtuuri tarvittaisiin käyttöasetusten optimointiin sekä siihen liittyvään tiedonhallintaan.

Artikkelin sisältö on seuraava. Luvussa 2 käydään läpi järjestelmällä olevat vaatimukset ja reunaehdot. Luku 3 kertoo, miten järjestelmä suunnitellaan ja toteutetaan. Luvussa 4 käydään läpi työhön liittyvä muu tutkimus. Lopuksi, luvussa 5 on yhteenveto artikkelin koko sisällöstä.

2 VAATIMUKSET JA REUNAEHDOT

Käyttöasetuksia koskevan palautteen tuottaminen koneellisesti ei ole vaatimuksena helppo. Jotta toimiva järjestelmä saadaan aikaiseksi, on siitä johdettava edelleen useita alivaatimuksia. Kun nämä vaatimukset on saatu ristiriidattomasti kirjattua, niiden pohjalta voidaan suunnitella, miten järjestelmä toteutetaan.

Jotta tarpeet saadaan konkretisoitua, tarkasteltakoon esimerkkiä, jossa halutaan optimoida hydraulisen tehon välittämistä työkoneen puomiin. Liiallinen teho (tässä tapauksessa käytännössä tilavuusvirta) hankaloittaa puomin hallintaa, kun taas liian alhainen teho aiheuttaa turhaa odottelua. Tehoon voivat vaikuttaa esimerkiksi pumpun tuotto tai venttiilien käyttö, kun nestettä ohjataan puomiin. Oletetaan, että näitä arvoja voidaan muuttaa käyttöasetuksilla. Minkälaiset käyttöasetusten arvojen tulisi olla? Vaikuttaako siihen käyttäjän taitotaso? Entä onko olosuhteilla tai tehtävän työn tyypillä merkitystä?

Vaikka koneet ovat hajautettuina ympäri maailmaa, niiden tuottama mittaustieto on saatava yhteen tietovarastoon tai big data -alustalle hyödyntämistä varten. Toisaalta ei voida olettaa, että jokaisella koneella olisi jatkuvasti toimiva Internet-yhteys. Siten sen on voitava säilöä mittausdataa koneessa ja lähettää se sopivalla hetkellä Internet-yhteyttä käyttäen tietovarastoon data-analyysia varten. Koska yhdellä konevalmistajalla on tyypillisesti useita erilaisia konetyyppejä, tieto on voitava jäsentää niiden mukaisesti. Edelleen on tallennettava tieto siitä, minkälaisissa olosuhteissa konetta on käytetty: lämpötila, maasto, koneen käsittelemän materiaalin tyyppi ja niin edelleen; kaikki, mikä voi vaikuttaa toisaalta mittaustuloksiin sekä toisaalta siihen, minkälaiset käyttöasetukset toimivat parhaiten.

Kun koneiden mittausdataa on kerätty tarpeeksi, suoritetaan tilastollinen data-analyysi, jolla on kaksi haluttua lopputulosta. Ensiksi saadaan tietoa siitä, minkälaiset käyttöasetukset toimivat eri olosuhteissa parhaiten. Rajaava tekijä tässä on, minkälaisista käyttöolosuhteista on dataa kullekin valmistajan konetyypille. Toiseksi saadaan tietoa siitä, minkälaiset suorituskykyarvot ovat ylipäänsä mahdollisia tietyissä olosuhteissa. Periaatteessa on mahdollista, että koneen kuljettaja saavuttaa hyvän suorituskyvyn, vaikka osa käyttöasetuksista ei olisikaan tilastollisesti parhaissa arvoissaan. Syy on yksilöllisyydessä: sekä ihmiset että jossain määrin koneetkin ovat yksilöitä. Lopulta, kun tilastollinen analyysi on suoritettu, sen tulokset tallennetaan jälleen omaan tietovarastoonsa myöhempää käyttöä varten. On huomattava, että koska koneista kerättävän mittausdatan määrä kasvaa koko ajan, on tarpeen suorittaa tilastollinen analyysi välillä uudelleen, jotta viimeisin tietämys voidaan löytää.

Toimialalla, jolla konetta käytetään, on luonnollisesti asiantuntijoita, jotka voivat sanoa suuntaa antavasti, millaiset käyttöasetukset kuhunkin konetyyppiin ja olosuhteisiin tulisi asettaa. Tilastollisten analyysien lopputuloksia on tarkasteltava kriittisesti; analyysia ei ole välttämättä tehty riittävän tiedon pohjalta, jotta tulos olisi täysin luotettava. Tällöin toimialan asiantuntija osaa sanoa, mitä tuloksia ei tule uskoa. Toisaalta asiantuntijankin osaaminen kehittyy, kun hän tarkastelee analyysien lopputuloksia. Tämä johtuu siitä, että ihmisen kyky ymmärtää ja tarkastella asioita on rajallinen, ja matemaattinen menetelmä voi olla tältä osin tarkempi. Joka tapauksessa pelkillä numeerisilla tuloksilla ei tulla toimeen, vaan on voitava käyttää myös ihmisten hiljaista tietoa.

Hiljainen tieto saadaan järjestelmäarkkitehtuurissa käyttöön esimerkiksi sääntökannan avulla. Sääntöjen ajatuksena on mallintaa ihmisen tuottamaa ja ylläpitämää tietoa siitä, minkälaiset käyttöasetusarvot ovat järkeviä olosuhteet huomioon ottaen. Säännöt voivat esimerkiksi sisältää ehdottomat ylä- tai alarajat, jotta data-analyysin tulosten pohjalta ei ehdotettaisi järjettömiä käyttöasetuksia. Edelleen data-analyysi ei välttämättä paljasta ihan kaikkea sitä, mitä asiantuntija jo tietää. Tietoteknisenä menetelmänä sääntöjen johtava ajatus on, että niiden avulla ohjelmistosovelluksen toimintalogiikkaan voidaan vaikuttaa suhteellisen helposti. Sääntöjen ylläpito ei

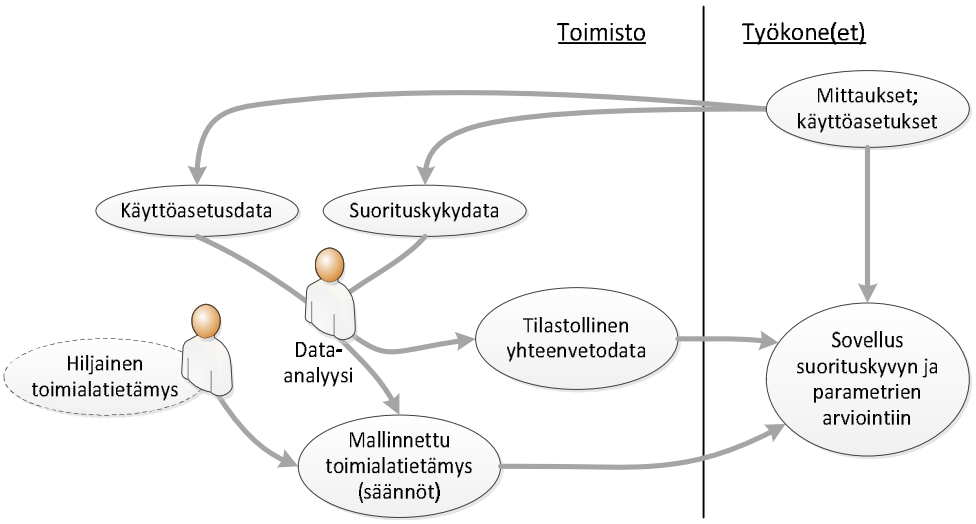
saa vaatia erityisosaamista IT-alalta (esimerkiksi ohjelmointiosaamista), eikä uusien tai muokattujen sääntöjen käyttöönotto saa olla vaikea operaatio (esimerkiksi kokonaisen ohjelmistokomponentin päivittäminen). Tähän tarkoitukseen on olemassa järjestelmiä, joilla sääntöjä voidaan ylläpitää ihmiselle intuitiivisella tavalla ja joilla säännöt saadaan tarvittaessa suoraan käyttöön.

Jotta tilastollisten analyysien tuloksista ja sääntöihin mallinnetusta toimialatietämyksestä saadaan paras hyöty, niitä on voitava hyödyntää työkoneella sen käytön aikana. Mikäli koneen mitattu suorituskyky on olosuhteisiin nähden alhainen, voi olla tarpeen tarkastella käyttöasetuksia. Koneen kuljettaja ei voi itse sellaista tehdä, joten koneelle on saatava suoritukseen sovellus, joka suorittaa tarkastelun. Tämä sovellus käyttää tilastollisten analyysien tuloksia sekä sääntöihin mallinnettua toimialatietämystä. Jotta sovellus voi päivittää niistä uusimmat versiot itselleen, ne on paketoitava sopivasti saataville, jotta sovellus voi hakea uusimmat versiot aina Internet-yhteyden ollessa käytettävissä. Sovellus siis lukee koneesta erilaisia mittausarvoja ja tarvittaessa laskee paikallisesti koneen suorituskykymuuttujia. Edelleen se hakee käytössä olevat koneen käyttöasetukset. Koneen suorituskykyä verrataan tilastollisiin arvoihin, jolloin saadaan tietoa koneen suhteellisesta suorituskyvystä. Sitten suoritetaan säännöt, joiden pohjalta tuotetaan palaute siitä, mitä koneen käyttöasetuksille pitäisi mahdollisesti tehdä. Palaute näytetään koneen kuljettajalle, joka voi niiden pohjalta päivittää koneen käyttöasetuksia. Jos koneen kuljettaja on riittävän etevä saavuttamaan hyvän suorituskyvyn, hänelle ei välttämättä voida tuottaa kovin hyviä parannusehdotuksia. Suurin haaste onkin parantaa vähemmän taitavien kuljettajien suorituksia. Toisaalta siihen liittyy myös suurin potentiaali.

3 SUUNNITTELU JA TOTEUTUS

Kun liikkuva kone mahdollistaa erilaisten sovellusten suorittamisen ja datan keräämisen antureistaan, järjestelmän tekninen toteutus riippuu ensisijaisesti soveltuvan järjestelmäarkkitehtuurin ja tiedonhallinnan kehittämisestä olemassa olevien Internet-tekniikoiden avulla. Tekniikat ovat hiljalleen yleistyneet 2000-luvulla erilaisten toimistotietojärjestelmien välisessä kommunikoinnissa ja integraatiossa. On eduksi, että palvelukomponenttien rajapinnat toteutetaan sovelluksen suoritusalustasta riippumattomilla, avoimilla tekniikoilla. Toisaalta rajapintasuunnittelussa tulee pyrkiä siihen, että järjestelmän komponenteista mikä tahansa voidaan tarvittaessa päivittää tai vaihtaa kokonaan ilman tarvetta kajota muihin komponentteihin.

Järjestelmän yleisarkkitehtuuri on esitetty alla (Kuva 1). Suuresta työkonejoukosta kerätään mittaus- ja käyttöasetusdataa keskitettyihin tietovarastoihin. Tietovarastojen sisältöä käyttäen suoritetaan data-analyyseja, jotta saadaan toisaalta tilastollista yhteenvetodataa (kuten keskiarvoja eri olosuhteissa) sekä toisaalta uutta toimialatietämystä. Sekä analyysin tuloksena syntyvä toimialatietämys että ns. hiljainen tieto mallinnetaan säännöiksi. Sekä tilastollinen yhteenvetodata että säännöt välitetään koneisiin, jossa niitä sovelletaan koneyksilön sen hetkisen suorituskyvyn ja käyttöasetusten hyvyyden arviointiin.



Kuva 1 Järjestelmän yleisarkkitehtuuri.

Esimerkkitapauksessa, jossa käsitellään koneen puomiin ohjattavaa hydraulitehoa, data-analyysi voi tuottaa yllättävääkin uutta tietämystä eri olosuhteisiin sopivista tehoon vaikuttavista asetuksista. Toisaalta toimialaosaa-

jilla voi ennestään olla tietoa esimerkiksi siitä, minkälaisia arvoja ei ainakaan kannata käyttää. Analyysin kohteena oleva data on esimerkiksi voinut olla liian suppeaa tai keskittyä vain tiettyihin käyttöolosuhteisiin. Siten toimialaosaajan laaja tietämys voi olla tarpeen, ettei kaikkia data-analyysin tuloksia laiteta sellaisenaan sääntöihin.

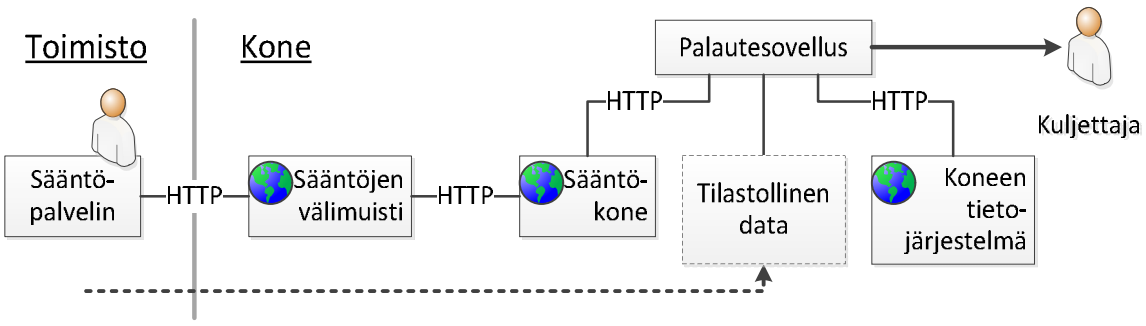
Suunnittelussa käytetään niin kutsutun *palvelukeskeisen arkkitehtuurin* (Service-Oriented Architecture eli SOA) periaatteita. Periaatteita on useita, mutta tämän työn kannalta kaikkein keskeisin on *abstraktio*. Ajatuksena on suunnitella palvelukomponentit siten, ettei käytettävä toteutusalusta näy millään tavalla palvelurajapintojen läpi. Siten rajapinnan takana olevan palvelun toteutuksella ei ole merkitystä, kunhan se täyttää palvelun toiminnalle asetetut vaatimukset. Abstraktio mahdollistaa osaltaan *löyhät kytkennät*, mikä tarkoittaa, että palvelukomponenttien riippuvuudet toisiinsa ovat mahdollisimman vähäisiä. Siten minimoidaan yhden palvelukomponentin muuttamisen tuottama työmäärä muissa komponenteissa; parhaassa tapauksessa muita komponentteja ei tarvitse muuttaa lainkaan. SOA:n periaatteita on käyty tarkemmin läpi esimerkiksi teoksessa /6/.

Palvelukomponenttien väliseen kommunikointiin valitut kaksi ydintekniikkaa ovat HTTP (HyperText Transfer Protocol) ja XML (Extensible Markup Language). Niistä kumpikin on avoin sekä alustasta riippumaton, ja kummallekin on olemassa laaja sovelluskehityksen työkalutuki. HTTP tunnetaan Internetin tietoliikenneprotokollana, jolla myös web-sivut tyypillisesti luetaan selaimeen. XML puolestaan määrittelee tekstipohjaisen syntaksin tiedon esittämiseen. Itse tiedon rakenne on kuitenkin sovelluskohtainen.

Sääntökone toteutetaan siten, että sen sisältämät säännöt mallinnetaan *sumeasti*. Se tarkoittaa tässä, ettei säännöissä käsitellä paljaita lukuarvoja, vaan lukuarvot skaalataan suullisiksi arvoiksi, jotka kuvaavat arvon suuruutta karkealla resoluutiolla. Siten mitattu työvaiheen kesto ei ole säännössä esimerkiksi 40 sekuntia, vaan tilanteesta riippuen se voi olla esimerkiksi ”heikko”, ”keskimääräinen” tai ”hyvä” – suullinen arvo siis kuvaa arvon hyvyyttä suhteessa keskimääräiseen arvoon vastaavissa olosuhteissa. Mittaus- ja suorituskykyarvoissa on aina epävarmuutta, joten sumeutus on tarpeen. Edelleen sääntöjä mallintavat toimialaosaajat eivät välttämättä ymmärrä lukuarvojen suuruusluokkien merkityksiä, ja edelleen sääntöjä voidaan ehkä käyttää uudelleen eri konetyypeille, vaikka absoluuttiset suorituskykyarvot olisivatkin niiden kesken poikkeavia.

Sääntökoneen ohjelmointirajapinnan mallinnuksessa pyritään yleiskäyttöisyyteen. Sääntökoneeksi soveltuvia tietokonesovelluksia on useita. Voidaan käyttää jotakin jo olemassa olevaa toteutusta, tai sitten voidaan toteuttaa järjestelmää varten oma sääntökoneensa. Oli toteutus mikä tahansa, sääntökone on järkevää kääriä geneerisen rajapinnan taakse, jotta se voidaan tarvittaessa korvata toisella. Toisaalta itse sääntöjen tietomalli todennäköisesti muuttuu järjestelmän elinkaaren aikana – tällainen ylläpitotarve ei saa aiheuttaa muutostarvetta rajapinnassa. Sääntökoneen käärivä komponentti suorittaa tarvittavan kuvauksen geneerisestä rajapinnastaan sääntöjen tietomalliin. Puomin hydraulitehoa käsittelevässä esimerkkitapauksessa voidaan huomata, että aiemmin toteutettu sääntöjen tietomalli on liian suppea kattamaan hydraulitehoa ohjaavan venttiilin asetukset. Mikäli tässä tapauksessa päätetään lisätä sääntöjen tietomalliin uusia piirteitä, ainoastaan sääntökanta sekä sen käärivä palvelukomponentti vaativat muutoksia.

Järjestelmäkehityksen yhteydessä on luotu prototyyppi (Kuva 2). Se kattaa sovelluksen, jota suoritetaan työkoneessa paikallisesti. Sovellus lukee koneen tietojärjestelmästä mittausdataa, jonka pohjalta lasketaan koneen suorituskyky edellisten työsyklien aikana. Samalla haetaan koneessa olevat käyttöasetukset. Suorituskykyarvoja verrataan tilastollisiin arvoihin konejoukosta mitatusta suorituskyvystä, jolloin saadaan suhteelliset suorituskykyarvot. Vastaavanlainen vertailu tehdään myös käyttöasetuksille. Suhteelliset arvot syötetään sääntökoneelle, joka päättelee, miten käyttöasetuksia tulisi mahdollisesti muuttaa, jotta suorituskykyä voitaisiin parantaa.



Kuva 2 Prototyyppijärjestelmän rakenne.

Prototyyppi tarvitsee työkoneen ulkopuolella tehtäviä toimintoja, jotta se saa tarvittavat tiedot arvioinnin suorittamiseen. Tilastolliset arvot haetaan Internetin yli toimistojärjestelmästä. Sääntöjen hakemista varten työkoneen sääntökoneella on paikallinen välimuisti. Internet-yhteyden ollessa käytettävissä välimuistissa oleva sääntöpaketti päivitetään, muussa tapauksessa käytetään aiemmin haettua versiota.

Prototyyppiä on suoritettu paikallisesti PC:ssä, jossa on ollut ajossa työkoneen tietojärjestelmän kopio, jossa on todellisen koneen käytön aikaista mittaus- ja käyttöasetusdataa. Näiden arvojen pohjalta on suoritettu kokeita, joiden perusteella prototyyppi ehdottaa käyttöasetusten arvojen muuttamista. Siten valtaosa sovelluksen peruslogiikasta on jo toteutettu prototyyppiin.

4 LIITTYVÄ TUTKIMUS

Hajautettujen *palveluarkkitehtuurien* tutkimus liittyy läheisesti tähän työhön, erityisesti teollisuus- ja laitekontekstin osalta. Palveluita on sovellettu paljon sähköisen kunnossapidon (ns. E-maintenance) toiminnoissa /2; 10/. Laitteiden tietoja tarjoavat rajapinnat voidaan toteuttaa palveluina, jolloin tietoa on helpompi hakea eri tietojärjestelmiin ja siten saada uutta liiketoimintahyötyä /4; 11/. Tietyissä teollisuuden toiminnoissa on reaaliaikavaatimuksia, joiden huomioon ottoa palveluarkkitehtuureissa on myös tutkittu /12; 3/. Käytännön palveluarkkitehtuurista on esimerkki työssä /7/, jossa sekä palvelukeskeistä tehdasmallijärjestelmää että laitteiden palvelurajapintoja sovelletaan kunnonvalvontatoimintoihin. Palveluiden hyödyntämistä tehdasmalleissa on käsitelty myös työssä /8/.

Toinen työtä sivuava aihe on *sääntöpohjaiset järjestelmät*, joille on useita erilaisia O&M-sovelluskohteita. Shrestha et al. /14/ ovat suunniteelleet järjestelmän, joka soveltaa sääntöjä vesivarantona toimivan järven operointiin, kun varannon tarvitsijoina on useita eri tahoja monenlaisine tarpeineen. Case-pohjaista päättelyä (Case-Based Reasoning eli CBR) on sovellettu teollisuuden kunnossapidon tietojärjestelmissä /15; 1/. Koneellisesti generoituja sääntöjä on käytetty ennustamaan tuotteiden läpimenoaikaa valmistuslinjalla /5/. Edelleen sääntöjä voidaan soveltaa tapahtumapohjaisissa järjestelmissä tapahtumien käsittelyn määrittämiseen /13/. Tarkastelun kohteena olevassa liikkuvien työkoneiden kontekstissa on aiemmin kehitetty sääntöpohjaista tietojärjestelmää käyttäjän suorituskyvyn arviointiin /9/.

5 YHTEENVETO

Tämä artikkeli esittää joukon vaatimuksia hajautetulle tietojärjestelmälle, jolla hallitaan liikkuvien työkoneiden suorituskyky- ja käyttöasetusdataa. Järjestelmän perimmäinen tarkoitus on opastaa koneen käyttäjiä löytämään parhaiten erilaisiin tarkoituksiin ja ympäristöihin parhaiten sopivat käyttöasetukset. Vaatimusten pohjalta esitetään arkkitehtuuri, jota noudattaen vaadittu tiedonhallintaratkaisu voidaan toteuttaa. Lopuksi esitellään prototyyppi, joka toteuttaa osan halutusta arkkitehtuurista. Suunnittelussa on pyritty luomaan palvelukomponenttiarkkitehtuuri, jonka eri osat ovat alustasta riippumattomasti toteutettuja sekä toisiinsa löyhästi kytkettyjä. Siten saavutetaan maksimaalinen joustavuus jatkokehityksessä ja ylläpidossa.

Yhteenvetona voidaan sanoa, että kyseessä ovat varsin monimutkaiset tiedonhallintaan liittyvät ongelmat. Suunnitteluun vaaditaan joukko monenlaisia palvelukomponentteja eri tarkoituksiin, eri komponentit on hajautettu voimakkaasti, ja järjestelmä sisältää sekä synkronista että taustalla tapahtuvaa asynkronista kommunikointia. Silti huolellisella ja tarkoituksenmukaisella arkkitehtuurisuunnittelulla saadaan aikaan järjestelmä, joka tuo selkeää lisäarvoa vaadittujen suorituskykykriteerien mukaisesti. Internetin kautta verkottuneet teolliset järjestelmät luovat edellytyksiä kehittää uusia toimintoja, palveluita ja liiketoimintamalleja. Nykyisessä globaalissa ja voimakkaasti kilpaillussa liiketoimintaympäristössä voittomarginaalit ovat pieniä, ja siten suhteellisen alhainenkin tuottavuusparannus voi vaikuttaa selvästi toiminnan kannattavuuteen.

Lisää tutkimusta vaaditaan, jotta prototyypistä voidaan laajentaa kokonainen järjestelmä käyttöasetusten optimointiin. Seuraavaksi samalta aihealueelta voitaisiin tutkia esimerkiksi koneiden vikadiagnostiikkaa tai palautteen antamista kuljettajalle koneen käytön aikana; kumpikin näistä aihealueista mahdollistaa tuottavuuden parantamisen edelleen.

KIRJALLISUUS

1. Aarnio, P., Seilonen, I. & Friman, M., 2014. Semantic repository for case-based reasoning in CBM services. In ETFA 2014, *Emerging Technology and Factory Automation.*

2. Bangemann, T., Rebeuf, X., Reboul, D., Schulze, A., Szymanski, J., Thomesse, J.P., Thron, M. & Zerhouni, N., 2006. PROTEUS – Creating distributed maintenance systems through an integration platform. *Computers in Industry*, 57.6 (2006): 539–551.

3. Cameron, A., Stumptner, M., Nandagopal, N., Mayer, W. & Mansell, T., 2015. Rule-based peer-to-peer framework for decentralised real-time service oriented architectures. *Science of Computer Programming 97* (2015): 202–234.

4. Cândido, G., Colombo, A.W., Barata, J. & Jammes, F., 2011. Service-oriented infrastructure to support the deployment of evolvable production systems. *Industrial Informatics, IEEE Transactions on* 7.4 (2011): 759–767.

5. Chang, P.C. & Liao, T.W., 2006. Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory. *Applied Soft Computing* 6.2 (2006): 198–206.

6. Erl, T., 2005. *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall.

7. Kannisto, P., 2011. *Service-Oriented Business Process Modeling in Operations and Maintenance*. Master of Science Thesis. Available: http://URN.fi/URN:NBN:fi:tty-2011051914675

8. Kannisto, P., Hästbacka, D. & Kuikka, S., 2013. Improving Asset Management: Plant Models Supported by Composite Services. *Automaatio XX Seminar*, Helsinki, Finland.

9. Kannisto, P., Hästbacka, D., Palmroth, L. & Kuikka, S., 2014. Distributed Knowledge Management Architecture and Rule Based Reasoning for Mobile Machine Operator Performance Assessment. In ICEIS 2014, *Proceedings of the 16th International Conference on Enterprise Information Systems*.

10. Karim, R., 2008. *A service-oriented approach to e-maintenance of complex technical systems*. Doctoral Thesis.

11. Karnouskos, S., 2012. Realising next-generation web service-driven industrial systems. *The International Journal of Advanced Manufacturing Technology* 60.1-4 (2012): 409–419.

12. Moritz, G., Prueter, S., Timmermann, D. & Golatowski, F., 2009. Deployment of embedded web services in real-time systems. *Scalable Computing: Practice and Experience* 10.3 (2009).

13. Paschke, A., Kozlenkov, A., & Boley, H., 2007. A homogeneous reaction rule language for complex event processing. In *Proc. 2nd International Workshop on Event Drive Architecture and Event Processing Systems* (EDA-PS 2007).

14. Shrestha, B.P., Duckstein, L. & Stakhiv, E.Z., 1996. Fuzzy rule-based modeling of reservoir operation. *Journal of water resources planning and management* 122.4 (1996): 262–269.

15. Yu, R., Iung, B. & Panetto, H., 2003. A multi-agents based E-maintenance system with case-based reasoning decision support. *Engineering applications of artificial intelligence* 16.4 (2003): 321–333.