Henri Niemeläinen

Mikropalveluiden ja konttien soveltuvuus toiminnanohjausjärjestelmien toteutuksiin

Tietotekniikan kandidaatintutkielma

7. kesäkuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Henri Niemeläinen

Yhteystiedot: henri.k.niemelainen@student.jyu.fi

Työn nimi: Mikropalveluiden ja konttien soveltuvuus toiminnanohjausjärjestelmien toteutuksiin

Title in English: Suitability of microservices and containers for ERP system implementations

Työ: Kandidaatintutkielma

Sivumäärä: 23+0

Tiivistelmä: Toiminnanohjausjärjestelmät ovat keskeisessä osassa suurten yritysten toimintaa, mutta yhä pienemmät yritykset ovat ottaneet niitä käyttöönsä. Toiminnanohjausjärjestelmät ovat tyypillisesti olleet suuria järjestelmiä joiden käyttöönotto ei ole ollut riskitöntä. Pienempien yritysten osalta myös toiminnanohjausjärjestelmille asetetut vaatimukset ovat poikkeavia suurista yrityksistä, joten tämä tutkielma tuokin esiin näkökulman toteutuksiin uusilla teknologioilla. Toiminnanohjausjärjestelmien toteutusta mikropalveluiden ja konttien avulla ei ole tutkittu, mutta tämä tutkielma pyrkii tuomaan esille mahdollisia toteutustapoja sekä niiden haittoja ja hyötyjä.

Avainsanat: toiminnanohjausjärjestelmä, mikropalvelut, kontit

Abstract: ERP systems are crucial part of large enterprises, but smaller enterprises have been adopting them in recent years. Typically ERP systems have been large monolithic systems, thus requiring careful planning to mitigate possible risks during the implementation. Smaller enterprises have different criteria for ERP systems than large enterprises, and therefore this paper will give some new views about implementing ERP systems using modern technologies. Although the suitability of implementing ERP systems using microservices and containers has not yet been explored in literary, this paper will try to give ideas how to implement ERP systems using containers and microservices, and to think about the possible benefits and challenges of doing so.

Keywords: ERP, microservices, containers

Kuviot

Sisältö

1 Johdanto

Toiminnanohjausjärjestelmät (ERP, ERP-järjestelmä) ovat yritysten toimintaa ja taloutta ohjaavia kokonaisuuksia, joiden tarkoituksena on tarjota yritykselle sen tarvitsemia ratkaisuja logistiikan, varastonhallinnan, kirjanpidon ja muiden yrityksen kannalta oleellisten osien hallintaan (Klaus, Rosemann ja Gable 2000). Johtuen yritysten toiminnan kannalta tärkeistä tarpeista, toiminnanohjausjärjestelmät ovat usein hyvin laajoja ohjelmistoja jotka tarjoavat kaikki ominaisuudet yhtenä isona kokonaisuutena (Davenport 2000). Varsinaisesta standardista ei voida puhua, vaan toiminnanohjausjärjestelmät kattavat käyttöönottavan yrityksen tarpeet ja ne vaihtelevat yrityksen toimialasta ja markkinasta riippuen (Klaus, Rosemann ja Gable 2000). Toiminnanohjausjärjestelmien käyttöönottoon liittyy paljon riskejä, mutta sen hyödyt ovat yritykselle merkittäviä. Onnistunutta toiminnanohjausjärjestelmän käyttöönottoa voidaankin mitata eri tavoin ja käyttöönottoon liittyvät vaiheet voidaan jakaa pienempiin osa-alueisiin lähtien yrityksen tarpeiden kartoituksesta, aina järjestelmän jatkokehitykseen saakka (Loh ja Koh* 2004). Onnistuneet Toiminnanohjausjärjestelmien käyttöönotot ovat tarvinneet hyvää kommunikointia, halukkuutta prosessimuutoksiin, muutoksenallintaa ja läheistä yhteistyötä järjestelmän kehittäjien ja sen loppukäyttäjien välillä (Klaus, Rosemann ja Gable 2000; Loh ja Koh* 2004; Ahmad ja Cuenca 2013). Johtuen toiminnanohjausjärjestelmien mittavasta koosta, niiden käyttöönotto on ollut tyypillistä suurille yrityksille, mutta myös PK-yritykset ovat ottaneet käyttöönsä toiminnanohjausjärjestelmiä. Pienten ja keskisuurten yritysten tarpeisiin perinteiset toiminnanohjausjärjestelmät ovat hyvin laajoja ja PK-yritysten osalta kaikkia järjestelmän tarjoamista toiminnallisuuksista ei hyödynnetä (Buonanno ym. 2005). Toisaalta erilaisten yritysten tarpeiden perusteella olemassaolevat toiminnanohjausjärjestelmät eivät myöskään välttämättä tarjoa yrityksille juuri niiden tarvitsemiaan ominaisuuksia joka aiheuttaa mukautustarpeita järjestelmän käyttöönotossa (Klaus, Rosemann ja Gable 2000; Loh ja Koh* 2004).

Viime vuosina erilaiset yleistyneet mikropalvelut ovat tapa toteuttaa ohjelmistoja siten, että ne keskittyvät vain yhden asian toteuttamiseen. Mikropalvelut tarjoavat hyvin suunniteltuja rajapintoja ja pystyvät toimivaan sellaisenaan, ilman riippuvuuksia muihin järjestelmiin. Mikropalveluille oleellista onkin tarkasti rajattu toimialue, joka mahdollistaa tarkem-

man vaatimusmäärittelyn ja siten tiukemman testauksen (Fowler 2014; Dragoni ym. 2017). Myös viimeaikoina esiin nousseet kontit, ovat oleellisessa osassa ohjelmistokehitystä. Kontit mahdollistavat ohjelmistojen toteuttamisen itsenäisinä kontteina, mitkä sisältävät niille tärkeät riippuvuudet ja tarjoavat eristetyn ympäristön. Kontteja on kehittänyt muun muassa Docker, joihin perustuvat useat uudet teknologiat, ja sen konttiteknologia voidaan nähdä yhtenä suurimpana vaikuttajana konttiteknologian kehityksessä. Itsestään kontit eivät tarjoa juurikaan etuja olemassaoleviin virtualiteknologioihin, mutta niiden hallintaan tarkoitetut orkestrointityökalut tuovat merkittäviä etuja konttien käyttöön.

Tämän tutkielman tarkoituksena on vastata kysymykseen, kuinka mikropalvelut ja kontit soveltuvat toiminnanohjausjärjestelmien toteutuksiin. Lisäksi tutkielman päämääränä on:

- kertoa toiminnanohjausjärjestelmistä
- löytää yhteys mikropalveluiden ja toiminnanohjausjärjestelmien välille
- antaa esimerkkejä mahdollisista toteutuksista
- pohtia hyödyistä ja haitoista mahdollisissa toteutuksissa

Alkuun selitetään tarkemmin toiminnanohjausjärjestelmistä, sekä niiden osista ja toteutuksiin liittyvistä haasteista. Sen jälkeen on kerrotaan, mitä ovat mikropalvelut ja kontit, sekä mikä niiden merkitys on ohjelmistotuotannossa. Lopuksi luodaan johtopäätöksiä mikropalveluiden ja konttien soveltuvuudesta toiminnanohjausjärjestelmien toteutuksiin.

2 Toiminnanohjausjärjestelmät

Yritysten kasvaessa ja teknologian kehittyessä yritysten toiminnanohjauksen tarpeet ovat kasvaneet siinä määrin, että yrityksen toiminnan tueksi on tarvittu järjestelmä (Muscatello, Small ja Chen 2003). toiminnanohjausjärjestelmät ovat yritysten toimintaa ja taloutta ohjaavia järjestelmiä. ERP (*enterprise resource planning*) termi ei ole hirveän kuvaava, sillä järjestelmien toiminnallisuus ei liity pelkästään yritysten resurssien hallintaan, vaan kaikkeen yrityksen toimintaan liittyvään, mistä syystä ERP termille onkin ehdotettu muita paremmin kuvaavia termejä (Klaus, Rosemann ja Gable 2000; Davenport 2000). Perinteisesti toiminnanohjausjärjestelmiä ovat hyödyntäneet suuret yritykset, mutta markkinoiden kasvaessa ja kehityksen jatkuessa yhä pienemmät yritykset ovat ottaneet käyttöönsä toiminnanohjausjärjestelmiä. Vaikka toiminnanohjausjärjestelmät tarjoavat tyypillisesti kaikki yrityksen vaatimat toiminnot, on järjestelmissä eroja johtuen yritysten ja toimialojen tarpeista. Perusominaisuuksiin voidaan katsoa kuuluvan kuitenkin taloudenhallintaan, laskutukseen, varastonhallintaan jne. liittyviä toiminnallisuuksia (Klaus, Rosemann ja Gable 2000).

Toiminnanohjausjärjestelmät ovat kehittyneet alkujaan teollisuuden tarpeiisin, mutta jo 1990-luvulta alkaen muidenkin toimialojen yrityksille on ollut hyötyä ottaa toiminnanohjausjärjestelmiä käyttöönsä (Klaus, Rosemann ja Gable 2000). Toiminnanohjausjärjestelmien kehitys ei ole suinkaan hiljentynyt viime vuosina, vaan kasvavat markkinat ja yritysten erilaiset toimintamallit ovat vaatineet uusia järjestelmiä toimiakseen. Myös toiminnanohjausjärjestelmien käyttöönotto on helpottunut tietojärjestelmien ja ohjelmistojen kehittyessä ja yhä pienemmät yritykset ovat pystyneet hyödyntämään yrityksen toimintaa ohjaavia järjestelmiä (Ahmad ja Cuenca 2013). Toiminnanohjausjärjestelmät ovat yleensä yhteiseen tietokantaan perustuvia monoliittisiä järjestelmiä, joiden etu hajautettuihin järjestelmiin tulee etenkin tiedon hallinnan vaivattomuudesta ja tiedonhallinnasta. Yrityksen eri komponentit pystyvät hyödyntämään järjestelmän tietoja vaivattomasti ja eri komponentteihin vaikuttavat toiminnot voivat luottaa, että käsiteltävät tieto on samaa koko järjestelmässä (Klaus, Rosemann ja Gable 2000).

Kirjallisuuden (Loh ja Koh* 2004; Aladwani 2001; Ehie ja Madsen 2005) perusteella toiminnanohjausjärjestelmien käyttöönotto voidaan jakaa useaan eri vaiheeseen, jotka ovat pääpiir-

teittäin adoptio-, hankinta-, toteutus-, ja jatkokehitys-vaiheet. Adoptiovaiheessa, selviää tekijät, joiden takia yritys päätyy ottamaan käyttöön toiminnanohjausjärjestelmän. Hankintavaiheessa yritys määrittelee tavoitetilan reunaehdot ja etsii nämä parhaiten täyttävän järjestelmän. Toteutusvaiheessa tapahtuu itse käyttöönotto. Tässä tutkielmassa keskitytään adoptio-, hankinta- ja toteutus-vaiheisiin.

2.1 Toiminnanohjausjärjestelmän adoptioon johtavat syyt

Suuret yritykset ovat päätyneet toiminnanohjausjärjestelmiin erilaisista syistä. Davenport (2000) mainitsee, että yrityksen hallinta ei olisi luontevaa ilman järjestelmää, joka sisältää keskitetysti kaiken yrityksen tarvitseman toiminnallisuuden. Pienten ja keskisuurten yritysten tarpeet eivät ole niin mittavia kuin suurilla yrityksillä, pienemmille yritykselle riittää usein toiminnoiltaan rajatumpia ja vähemmän ominaisuuksia sisältävä toiminnanohjausjärjestelmä (Doom ym. 2010), ja palveluntarjoajat ovatkin tuoneet markkinoille kevennettyjä palvelukokonaisuuksia, sekä PK-yrityksille suunnattuja järjestelmiä (Loh ja Koh* 2004).

Toiminnanohjausjärjestelmien käyttöönottoon päädytään usein yrityksen sisäisistä muutospaineista johtuen (Buonanno ym. 2005). Käyttöönottava yritys näkee että sille on hyötyä ottaa käyttöön järjestelmä, tai että olemassaolevat järjestelmät eivät toteuta yrityksen niille asettamia tavoitteita. Suuret yritykset päätyvät useimmiten toiminnanohjausjärjestelmän käyttöönottoon suurien organisaatiomuutosten ja prosessimuutoksien yhteydessä, mutta PK-yritykset ovat usein tyytyväisiä olemassaolevaan organisaatioon ja prosesseihin ja päätyvät käyttöönottoon muista syistä (Buonanno ym. 2005). PK-yritykset hakevat lisäarvoa toiminnanohjausjärjestelmistä, kun taas suuret yritykset pääpiirteittäin pyrkivät kehittämään ja ohjaamaan liiketoiminnan strategiaa (Buonanno ym. 2005).

2.2 Toiminnanohjausjärjestelmän hankinta ja toteutus

Toiminnanohjausjärjestelmän hankintavaihe alkaa saatavilla olevien järjestelmien rajauksesta, missä saatavilla olevista järjestelmistä valitaan parhaiten kriteerit täyttävät järjestelmät. Toiminnanohjausjärjestelmiä on tarjolla sekä kaupallisia että avoimeen lähdekoodiin perustuvia ratkaisuja ja suurista kansainvälisistä toimijoista esimerkiksi SAP ja Oracle tarjoavat

omia ratkaisujaan, mitkä ovat suunnattuja suurten yritysten tarpeisiin. Avoimeen lähdekoodiin perustuvia ratkaisuja ovat esimerkiksi Odoo (ennen OpenERP) ja ERPNext. Olemassaolevien järjestelmien laajuudet vaihtelevat suurien yritysten tarpeista keskisuurille yrityksille, mutta pienempien yrityksien tarpeet eivät usein ole linjassa suurempien yritysten toimintamallien ja toimintatapojen kanssa (Buonanno ym. 2005). Järjestelmien rajaukseen ja valinnan tekemiseen on tehty erilaisia malleja, mutta näyttöä niiden soveltamisesta käyttöönotossa PK-yrityksissä ei ole. Rajausvaiheessa tyypillistä on, että rajauksen valintaa osallistuu, hyvä otos järjestelmää käyttävistä henkilöistä, sekä projektimanageri sekä mahdollisia teknisiä konsultteja, jotta järjestelmä pystytään rajaamaan tarpeita vastaavaksi. Mikäli hankintavaiheessa löydetyissä järjestelmissä on paljon mukautustarpeita, ovat yritykset tarkastelleet myös mahdollisuutta toteuttaa täysin uuden järjestelmän, mutta useimmiten ovat päätyneet valmiiden järjestelmien mukauttamiseen (Olsen ja Sætre 2007; Zach ja Erik Munkvold 2012).

Tyyppillisesti valmiit toiminnanohjausjärjestelmät tarjoavat yrityksen toiminnanohjauksen perustoiminnot ja toimialakohtaiset toiminnallisuudet toteutetaan joko osana toiminnanohjausjärjestelmää tai erikseen. Suurten toimijoiden tuottamat toiminnanohjausjärjestelmät eivät sovellu kokonaisvaltaisesti yrityksille, vaan käyttöönottavan yrityksen tarpeet ovat tärkeässä asemassa. toiminnanohjausjärjestelmät ovat suunniteltu laajennettaviksi ja eri toimialoihin mukautuvuksi (Lehrer 2006; Klaus, Rosemann ja Gable 2000). Löytyneestä kirjallisuudesta selviää, että toiminnanohjausjärjestelmien käyttöönotto vaatii usein yrityksen sisäisiä muutospaineita, ja onnistuneet toiminnanohjausjärjestelmien käyttöönotot vaativat usein yritysten prosessien uudelleensuunnittelua (Loh ja Koh* 2004). Tästä johtuen olemassaolevat järjestelmät eivät sellaisenaan kelpaa yritykselle, vaan on tyypillistä, että yritys ottaa käyttöön järjestelmän, jota on mukautettu sen tarpeita vastaavaksi (Zach ja Erik Munkvold 2012). Hankitavaiheessa selvinneet mukautustarpeet vaikuttavat järjestelmän käyttöönottoon ja onkin ehdotettu, että yrityksille olisi hyödyllistä ottaa toiminnanohjausjärjestelmä käyttöön juuri organisaatio- ja prosessi-muutosten yhteydessä, jolloin mahdollisia järjestelmän mukautustarpeita voidaan lieventää, muuttamalla yrityksen organisaatiota ja prosesseja (Loh ja Koh* 2004). Toisaalta on myös mainittu, että yrityksen liiallinen mukautuminen käyttöönotettavaa järjestelmää varten ei ole kannattavaa (Buonanno ym. 2005).

Toteutusvaiheessa erittäin tärkeässä asemassa on projektijohto ja projektin hallinta. Useat lähteet (Loh ja Koh* 2004; Muscatello, Small ja Chen 2003; Laukkanen, Sarpola ja Hallikainen 2007) kertovat, että toiminnanohjausjärjestelmä vaatii vahvaa projektijohtamista, sekä tahtotilaa toteuttaa käyttöönotto. Käyttöönoton onnistumisen yksi merkittävimmistä tekijöistä onkin käyttöönoton projektivaiheen hallinta ja muutoshalukkuus (Ahmad ja Cuenca 2013). Käyttöönottaessa tulee olla yrityksen johdon vahva tuki (Doom ym. 2010) ja lopuksi tärkeää on myös käyttäjien kouluttaminen järjestelmän käyttöön, etenkin järjestelmän käyttöön liittyvien ongelmien ehkäisyn kannalta. Teknisen toteutuksen ja järjestelmän käyttäjien käsityksen järjestelmästä tulisi tulla esille toteutusvaiheessa, ja mahdolliset epäkohdat tulisi käsitellä toteutusvaiheessa koulutusten ja järjestelmän manuaalien kautta (Loh ja Koh* 2004).

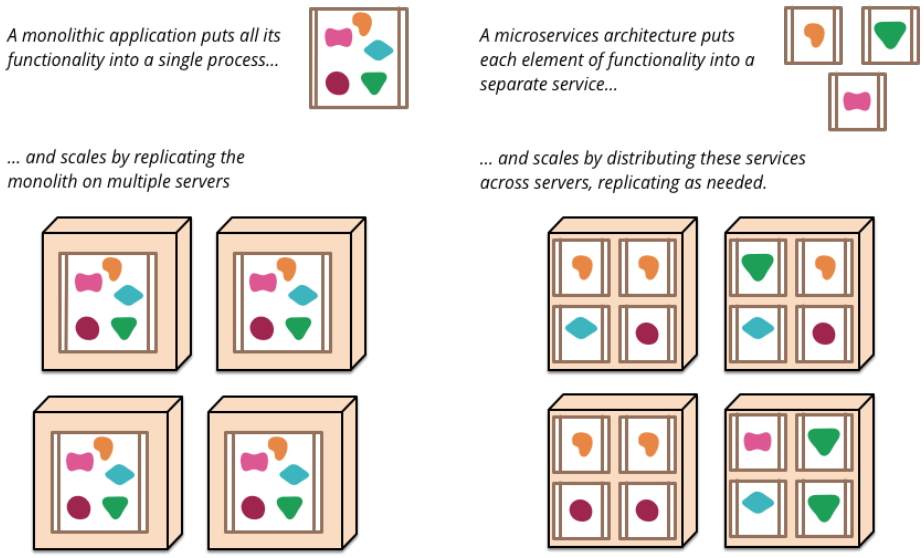
3 Mikropalvelut ja kontit

Viime vuosina ohjelmistotuotannossa esiintyneet mikropalvelut ja kontit ovat tarjonneet uusia tapoja suunnitella järjestelmien toteutuksia. Mikropalvelut ovat pyrkineet eristämään yksittäisiä osia isommista järjestelmistä ja kontit ovat pyrkineet tarjoamaan tehokkaampia virtualisointitekniikoita sekä tapoja riippuvuuksien hallintaan. Mikropalvelut eivät ole varsinaisesti uusi idea, vaan niiden määrittely on viime vuosina tarkentunut ja ohjelmistotuotannon kehitystapojen muuttuessa ne ovat nousseet esille hyvänä tapana kehittää uusia järjestelmiä. Kontit sen sijaan ovat uusi tapa, joka yhdistää vanhempia periaatteita virtualisoinnista uusien kehitysmetodien kanssa.

3.1 Mikropalveluiden piirteitä

Mikropalveluista on erittäin kattavasti kertonut muun muuassa Fowler (Fowler 2014) ja Dragoni et al. (Dragoni ym. 2017). Mikropalvelu käsite ei itsessään kerro kuinka kompleksi se on ominaisuuksiltaan, vaan kertoo mikropalvelun luonteesta. Mikropalvelut ovat tarkasti määriteltyjä ohjelmistoja jotka toteuttavat vain niiden toimialueeseen liittyvän toiminnon ja tarjoavat sen rajapinnan kautta käytettäväksi. Mikropalveluille olennainen rajapinta tekee niistä helposti uudelleenkäytettäviä ja mahdollistaa mikropalvelun käytön osana muita järjestelmiä ilman muutoksia itse mikropalveluun. Mikropalveluiden rajoitteita taas ovat niiden tarkka määrittely, mikä vaikeuttaa niiden kehitystä, sillä olemassaolevaa mikropalvelua tulee kehittää sen määritelmien sallimissa puitteissa, jotta mikropalvelun kehityksestä aiheutuvat muutokset eivät aiheuttaisi muutostarpeita sitä hyödyntävissä järjestelmissä.

Mikropalveluille tyypillistä on niiden käyttäminen järjestelmissä, joka koostetaan useista mikropalveluista. Koostaminen on hyvin pragmaattinen lähestymistapa ohjelmistokehityksessä, joissa pyritään tuottamaan haluttu palvelu mahdollisimman vähällä vaivailla. Mikropalveluiden osalta voidaan luottaa siihen, että niiden rajapinnat eivät muutu merkittävästi, ja vaativat vähän resurssia niiden käyttöönotossa ja ylläpidossa. Tämä on etenkin hyödyllistä, kun toteutetaan järjestelmiä pienin toimiva tuote (MVP, minimum viable product) periaatteella, ja tarkoituksena on vain saavuttaa tavoitetila. Mikropalveluista koostetun järjestelmän



Kuvio 1. Mikropalveluarkkitektuuri Fowler 2014

suorituskyky ei useinkaan ole merkittävä mittari, vaan tarvittaessa järjestelmää voidaan optimoida myöhemmin. Koostaessa järjestelmän eri mikropalveluita voidaan myös vaihtaa vähäisellä vaivalla. Mikäli esim. järjestelmään valittu tietokanta ei ole tarpeeksi nopea, voidaan sen tilalle valita toinen tietokanta joka toteuttaa saman rajapinnan.

3.2 Mitä ovat kontit

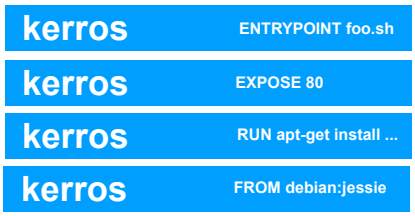
Kontit (*containers*) ovat virtualisointiteknologia missä kullekin kontille jaetaan käyttöjärjestelmän ydin sekä resurssit, mutta muilta osin ne toimivat omissa erillään muista käyttöjärjestelmässä toimivista konteista. Kullekin kontille näkyy, kuin se olisin käyttöjärjestelmän ainoa käyttäjä, vaikka todellisuudessa kontit toimivatkin vain omassa ympäristössään. Perinteisesti virtualisointia varten on tarvittu raskas virtualisointialusta, joita ei konttien tapauksessa tarvita, sillä järjestelmän resurssit jaetaan muilla tavoin. Käyttöjärjestelmän virtualisointi perinteisessä virtualisointialustassa onkin raskasta ja näkyy virtualisointialustan vieraskäyttöjärjestelmän käynnistysajoissa. Usein virtualisointialustalle pystytetään samoja vieraskäyttöjärjestelmiä, kun taas kontit jakavat yhden käyttöjärjestelmän, mikä vähentää huomattavasti resurssien kulutusta.

Konttien virtualisointi järjestelmätasolla ei ole ainoa asia minkä takia ne ovat nousseet esille,

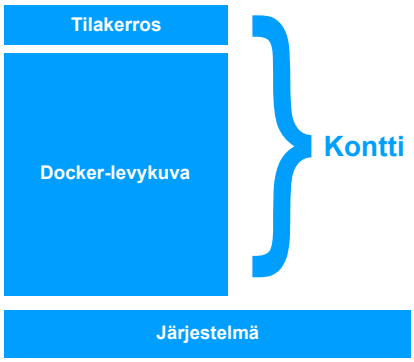
vaan niiden tarjoama kokonaisuus helppona tapana hallita riippuvuuksia ja tarjota ympäristö järjestelmälle. Konteissa on tosin myös varjopuolia. Verrattuna perinteiseen virtualisointiin, missä ei jaeta käyttöjärjestelmän ydintä, konteissa haavoittunut käyttöjärjestelmän ydin voisi mahdollisesti vaarantaa käyttöjärjestelmän sekä muiden konttien tietoturvan. Tässä tutkielmassa keskitytään Docker-konttiteknologiaan, mutta on hyvä tiedostaa että muut vastaavat konttiteknologiat, kuten CoreOS rkt, tarjoavat samantyyppisiä ominaisuuksia vaikka niitä ei tässä tutkielmassa mainita. Kontit ovat jo oleellinen osa pilvipalveluja, ja niillä on suuri kehittäjäyhteisö takana.

3.2.1 Docker-konttiteknologia

Konttiteknologioista tunnetuin lienee Dockerin kehittämä Docker-konttiteknologia (Docker 2018). Docker tarjoaa helpon tavan luoda ympäristöriippumattomia ohjelmistopaketteja, ja se hyödyntää Linuxin LXC (LinuX Container) teknologiaa konttien eristämiseen sekä tarjoaa ratkaisun riippuvuuksien hallintaan. Dockerin konttiteknologia koostuu palveluprosessista, jota komennetaan erillissellä prosessilla. Palveluprosessin avulla luodaan, hallitaan ja suoritetaan kontteja.



Kuvio 2. Docker-levykuva.



Kuvio 3. Docker kontti.

Dockerissa kontti koostuu palveluprosessisa suoritettavasta levykuvasta (Docker-image), ja sen tilan tallentavasta kerroksesta. Levykuva puolestaan koostuu erilaisista kerroksista, jotka suorittavat jonkin komennon, esimerkiksi jonkin ohjelmiston riippuvuuden suorittava asennuskomento. Levykuva luodaan näitä komentoja suorittamalla, täten levykuva on aina toistettavissa toisella palveluprosessilla, joka suorittaa kyseiset komennot, muodostaen saman-

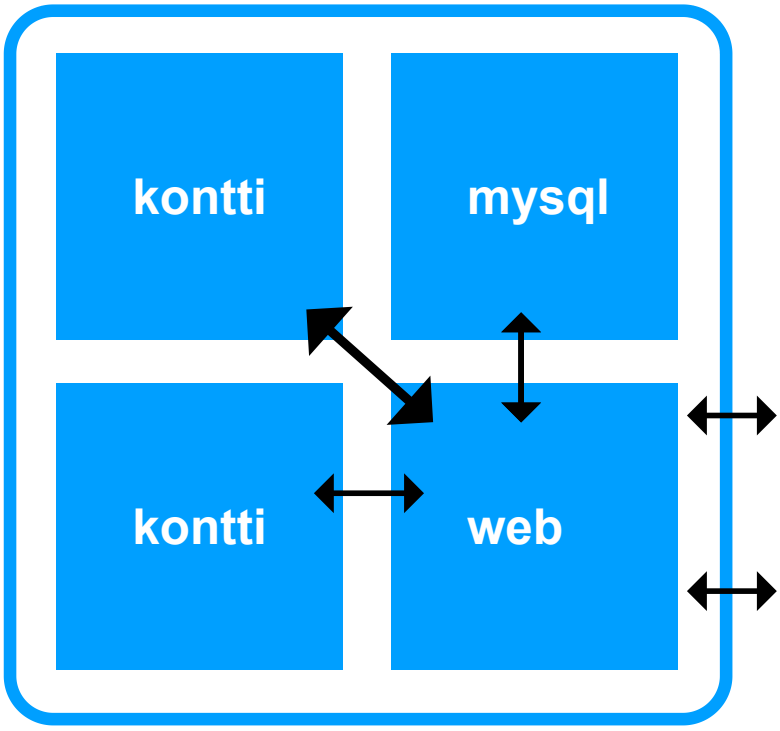
laiset kerrokset ja siten samanlaisen levykuvan. Saadut levykuvat voidaan lisäksi jakaa julkisesti internetissä (Docker-hub), ja siten niitä voi hyödyntää suuri määrä muita kehittäjiä. Replikoitavat levykuvat mahdollistavat täsmälleen samanlaisten levykuvien tekemisen, sillä kontit ovat replikoitavissa missä tahansa muussa ympäristössä missä palveluprosessi toimii, joka tekee konteista ideaaleja kehittämiseen ja tuotantokäyttöön, joissa riippuvuuksien hallinta on ollut perinteisesti haastavaa. Konttia suorittaessa käyttävät kaikki samaan levykuvaan perustuvat kontit samoja resursseja Dockerissa käytetyn CoW (Copy on Write) teknologian ansiosta, joka mahdollistaa vähäisen resurssien käytön verrattuna perinteiseen virtualisointiin, missä resurssit eivät ole jaettuja. Docker jakaa samat resurssit, mutta vasta kun kontti käsitellee itse resursseja, luodaan kontille tarvittavat itsenäiset resurssit, jotka tallentuvat kontin omaan tilan tallentavaan kerrokseen.

3.2.2 Konttien hallinnointi

Kontit itsessään ovat verrattain yksinkertainen teknologia, mikä ei itsessään tarjoa merkittävää parannusta perinteisiin virtualisointiteknologioihin. Konttien edut tulevat esiin kun otetaan käyttöön orkestrointi eli hallintatyökalut. Esimerkiksi Docker Swarm on Dockerin oma tapa hallita useita Docker kontteja ja niiden instansseja. Kubernetes puolestaan on alunperin Googlen kehittämä nykyään avoimeen lähdekoodiin perusutuva teknologia, joka tarjoaa työkaluja konttien automatisointiin ja niiden hallintaan. Sekä Google että Red Hat ovat Kubernetes-projektin takana, ja pyrkivät aktiivisesti toimimaan kehittäjäyhteisön kanssa sen kehittämiseksi. Kubernetes tarjoaa ominaisuuksia, joita Dockerin konttiteknologiasta itsestään ei löydy, ja se hyödyntää yritysten mittavaa kokemusta suurten järjestelmien toteutuksista. Kubernetes tarjoaa esimerkiksi klusteroinnin, hallinta- ja valvonta-työkaluja, joiden avulla sen päälle toteutetut järjestelmät ovat paremmin skaalattavissa. Kubernetes projektin päälle on puolestaan rakentunut Red Hat:n Openshift projekti, joka mahdollistaa konttien suorittamisen pilviympäristössä. Konttiteknologia ja niiden hallinnointiteknologiat ovat luoneet uudenlaisen markkinan. Markkinoille on tullut useita PaaS (Platform as a Service) toimijoita, jotka tarjoavat konttiympäristöjä pilvipalveluina ja konttien hallinointiteknologioita.

3.3 Mikropalveluiden ja konttien yhteydestä

Konttiympäristöihin toteutetut ohjelmistot ovat helpommin testattavia ja niiden riippuvuudet on paremmin tiedossa (Merkel 2014). Ohjelmistojen riippuvuuksien hallinta onkin yksi ohjelmistotuotannon haasteista, mikä vaikeuttaa ohjelmistojen kehitystä, sillä riippuvuudet tuovat mukanaan uusia riippuvuuksia ja rajaavat käytettävissä olevia valmiita ohjelmakirjastoja. Konttien tarjoama toiminnallisuus mahdollistaa riippuvuuksien tarkemman määrittelyn ja mikropalvelut puolestaan tarjoavat tarkoin määriteltyjä palveluita. Mikäli mikropalveluita toteutetaan konttiympäristössä, on niiden riippuvuudet tarkoin selvillä, ja niiden tarjoamat rajapinnat ovat myös helposti saatavilla.



Kuvio 4. Mikropalveluista koostettu järjestelmä

Konttiympäristön päälle voidaan siis rakentaa vaivatta mikropalveluita ja niiden tarjoaminen muille konteille tarjoaakin erittäin selkeän arkkitehtuurin rajoittaen kuitenkin ohjelmiston kompleksisuutta (Killalea 2016). Esimerkkinä tietokanta voidaan toteuttaa mikropalveluna konttiympäristöön, jolloin konttiympäristö hoitaa tietokannan riippuvuudet ja tietokantaa

voidaan huoletta käyttää muissa ohjelmistoissa vaikuttamatta oman ohjelmiston riippuvuuksiin. Yhtälailla useampi ohjelmisto voi vaatia samaa tietokantaa joka tarkoittaa vain saman kontin uudelleenkäyttöä. Kun ohjelmistoja rakennetaan konttiympäristöön pieninä paketteina, pysyy niiden laajuus hallittavana ja riippuvuudet selkeinä (Killalea 2016). Koostaminen onkin yksi konttiympäristön tarjoamista eduista mikropalveluista, ja hallittu konttiympäristö onkin luonnollinen alusta mikropalveluille (Jaramillo, Nguyen ja Smart 2016). Kun mikropalvelut toteutetaan standardimuotoisina kontteina, jää järjestelmän kehittäjälle ainoastaan tehtäväksi mikropalvelujen yhteenliittäminen rajapintojen avulla.

4 Mikropalvelujen soveltuvuudesta toiminnanohjausjärjestelmiin

Mikropalveluiden soveltuvuudesta toiminnanohjausjärjestelmiin ei juurikaan löydy kirjallisuutta. Joten tämä kappalle keskittyy kirjoittajan pohdintaan, kuinka mikropalvelut ja kontit soveltuisivat toiminnanohjausjärjestelmiin ja miten niitä voitaisiin toteuttaa mikropalveluina.

Toiminnanohjausjärjestelmät ovat alunperin kehittyneet erillään olevista järjestelmistä, joten mikropalveluilla toteutettava järjestelmä olisi eräänlainen askel taaksepäin. Mikropalveluilla tehtävälle toteutukselle olisi tyypillistä, että järjestelmän eri osat rajattaisiin tarkasti. Nykyisissä toiminnanohjausjärjestelmissä, esimerkiksi varastonhallinta voitaisiin toteuttaa itsenäisenä palveluna, joka toteuttaa varastonhallinnan perusominaisuudet riippumatta muista komponenteista, kuten laskutuksesta ja kirjanpidosta. Täten varastonhallintajärjestelmiä voitaisiin tarvittaessa pystyttää uusille varastoille ja virtuaalivarastoille tarvittaessa. Varastonhallintajärjestelmää ohjattaisiin siten rajapintojen avulla, esimerkiksi tilausjärjestelmän kuuluisi ymmärtää varastojärjestelmän rajapinta ja osata luoda tarvittavia varastosiirtoja sille. Eri järjestelmistä koostettaisiin tarvittaessa laajempia näkymiä, joiden avulla yrityksen toimintaa voitaisiin tarkastella ja hallita. Tämä rajoittaisi puolestaan näkymien laajuutta ja rajaisi näkymien toteutukseen vaadittavia resursseja. Näkymiä voidaan kehittää riippumatta muista järjestelmistä, ja mikäli näkymät toteutettaisiin mikropalveluperiaatteella, olisi näkymistä mahdollista koostaa yhä laajempia näkymiä joiden avulla saataisiin lopulta luotua kokonaiskuva kaikista yrityksen omista järjestelmistä.

Koostetun järjestelmän ei tarvitse alkujaan olla suuri, kuten suurilla yrityksillä, vaan PK-yritysten tapauksessa voidaan toteuttaa vain yrityksen liiketoiminnan tarpeiden kannalta merkittävät toiminnallisuudet. Tämä rajaa myös käyttöönoton riskejä, sillä käyttäjien koulutukseen ja prosessimuutoksiin vaadittavat resurssit voisivat pienentyä, mutta silti tarjota yritykselle mahdollisuuden laajentaa järjestelmän toiminnallisuuksia pienemissä iteraatioissa. Voitaisiin myös nähdä, että pienemmät järjestelmät pystyttäisiin alkuun ottaa käyttöön nopeasti, ja mahdollisen käyttöönoton yhteydessä muutostarpeiden ilmetessä, pystyttäisiin toteuttamaan muutoksia järjestelmiin suuria järjestelmiä nopeammin. Myös mahdollisten virhei-

den kannalta lyhyet iteraatiot olisivat PK-yrityksille kannattavia. Mikäli huomattaisiin järjestelmän soveltumattomuus, olisi siitä poissiirtyminen nopeaa. Konttiympäristön voisi taas puolestaan nähdä rajoittavan järjestelmän ylläpidosta koituvia kustannuksia, sillä erilaisten riippuvuuksien hallinta olisi toteutettu konteilla. Orkestrointityökalut puolestaan tarjoaisivat mahdollisuuksia pystyttää mikropalveluita nopeasti ja erillään kaikista muista järjestelmistä.

4.1 Toteutuksen etuja

Alunperin yritysten eri tietojärjestelmät toimivat erillään, ja toiminnanohjausjärjestelmien saavuttua muuttuivat toiminnanohjausjärjestelmät monoliittisiksi kokonaisuuksiksi. Ideana toiminnanohjausjärjestelmän toteuttaminen erillisinä mikropalveluina olisi eräänlainen askel taaksepäin, mutta toiminnanohjausjärjestelmien käyttöönottoon kuuluu oleellisesti järjestelmän mukauttaminen yrityksen tarpeisiin, joten mikropalvelut voisivat toisaalta tarjota ratkaisuja toiminnanohjausjärjestelmien toteutuksiin etenkin pienemmille yrityksille. Pienten yritysten kannalta iteratiiviset pienemmän vaiheet, eivät loisi suuren järjestelmän käyttöönoton riskejä, vaan rajoittaisivat niitä.

Mikäli toiminnanohjausjärjestelmä toteutettaisiin mikropalveluina, olisi kunkin mikropalvelun uusiokäyttö helppoa. Tämä tarjoaisi myös kustannustehokkaamman vaihtoehdon pienemmille yrityksille joiden ei tarvitsisi kuin kehittää oma tarvittava mikropalvelu mahdollisille yrityksen omille tarpeille. Kehitys tapahtuisi näin pienemmissä yksiköissä ja pienemmin askelein. PK-yrityksille olisi kannattavampaa suorittaa järjestelmän toteutusta pienemmin askelin. Esimerkiksi, pienelle yrityksylle varastonhallintajärjestelmän ottaminen käyttöön mikropalveluna ei sitoisi yritysten muiden järjestelmien kehitystä, eikä rajoittaisi niiden toimintaa. Myöhemmin mikropalveluna toteutettu osajärjestelmä voitaisiin laajentaa tai vaihtaa kokonaan. Mikropalveluilla toteuttaessa yrityksen ei tarvitsisi ottaa käyttöön kaikkia suuren toiminnanohjausjärjestelmän ominaisuuksista, mikä lieventää yrityskelle aiheutuvia prosessimuutostarpeita. Tämä puolestaan olisi etu pienemmille yrityksille, jotka eivät ole yhtä halukkaita muuttamaan yrityksen prosesseja verrattuna suuriin yrityksiin.

4.2 Toteuksiin liittyviä haasteita

Toiminnanohjausjärjestelmien perinteiset edut ovat tulleet tiedon keskittämisestä ja sen helposta hallitsemista, joten mikropalveluna toteutettava järjestelmä vaatisi tarkkaa suunnittelua tiedon rakenteiden ja formaattien kanssa. Tietojen siirto palvelusta toiseen vaatisi kultakin palvelulta mahdollisuuden viedä ja tuoda tietoa tietyissä formaateissa, tai vaihtoehtoisesti palveluiden välille tulisi toteuttaa adaptereita, jotka lieventäisivät tietojen formaattien määrittelyn tarkkuutta.

Keskeisessä osassa toiminnanohjausjärjestelmiä on myös tiedon kulku ja tapahtumien hallinta. Mikropalveluina toteuttavassa järjestelmässä tapahtumista ei olisi tietoa kaikissa palveluissa, vaan tapahtumien hallinta olisi myös toteutettu palveluna. Tämä olisi etenkin haastellista verrattuna monoliittisiin järjestelmiin, missä tiedon kulku ja tapahtumien käsittely olisi keskeisessä sijainnissa, ja kaikkien muiden järjestelmien käytössä. Mikropalveluilla toteuttaessa kukin mikropalvelu, joutuisi joko itse huolehtimaan tapahtumista tai kommunikoimaan jatkuvasti tapahtumia hallitsevan järjestelmän kanssa. Mikäli esimerkiksi laskutusjärjestelmä ja tilausjärjestelmä suorittaisivat toimintoja yhtäaikaisesti varastonhallintajärjestelmän kanssa, olisi mahdollisuus, että huonosti määritelty tapahtumien hallinta aiheuttaisi ongelmia tiedon eheyden kanssa. Yksi keskeisimmistä ongelmatekijöistä hankintavaiheessa olisi luultavasti toiminnanohjausjärjestelmän perustoiminnallisuudet, jotka ovat huomattava määrä eri komponentteja. Mikäli markkinoilta ei löytyisi valmiita mikropalveluita esimerkiksi laskutus ja kirjanpitojärjestelmiin, joutuisi yritys tuottamaan nämä järjestelmät itsenäisesti, mikä vähentäisi mikropalvelujen tuomia hyötyjä.

5 Yhteenveto

Konttien ja mikropalveluiden soveltuvuudesta toiminnanohjausjärjestelmiin ei löytynyt suoraan kirjallisuutta, mutta löytyneet materiaalin perusteella voisi päätellä, että kontit ja mikropalvelut toimisivat järjestelmän toteutukseen. Etenkin pienempien yritysten tapauksessa, pienemmät järjestelmät rajoittavat käyttöönoton riskien laajuutta ja mahdollistavat nopeamman käyttöönoton, mutta toisaalta eri palveluiden toteuttaminen vaatii tarkempaa määrittelyä, jotta eri palvelut pystyisivät toimimaan luotettavasti muiden palveluiden kanssa. On selvää, että mikropalvelut ja kontit tulevat säilymään tärkeänä ja kasvavana osana kehitystä ja ohjelmistotuotantoa. Jäänee nähtäväksi, kuinka nopeasti suuret yritykset alkavat tarjoamaan palveluita myös konttimuodoissa, ja olisikin hyvä tutkia kuinka esimerkiksi "private cloud"ja konttiympäristöt tulevat muodostamaan ekosysteemejä, joissa järjestelmiä voidaan tilata ns. konttimuodossa omassa ympäristössä suoritettaviksi.

Lähteet

Ahmad, M. Munir, ja Ruben Pinedo Cuenca. 2013. “Critical success factors for ERP implementation in SMEs”. Extended Papers Selected from FAIM 2011, *Robotics and Computer-Integrated Manufacturing* 29 (3): 104–111. ISSN: 0736-5845. doi:https://doi.org/10.1016/j.rcim.2012.04.019.

Aladwani, Adel M. 2001. “Change management strategies for successful ERP implementation”. *Business Process management journal* 7 (3): 266–275.

Buonanno, Giacomo, Paolo Faverio, Federico Pigni, Aurelio Ravarini, Donatella Sciuto ja Marco Tagliavini. 2005. “Factors affecting ERP system adoption: A comparative analysis between SMEs and large companies”. *Journal of Enterprise Information Management* 18 (4): 384–426.

Davenport, Thomas H. 2000. *Mission critical: realizing the promise of enterprise systems.* Harvard Business Press.

Docker. 2018. *Docker.* As WWW page, https://www.docker.com/, referenced 25.2.2018.

Doom, Claude, Koen Milis, Stephan Poelmans ja Eric Bloemen. 2010. “Critical success factors for ERP implementations in Belgian SMEs”. *Journal of Enterprise Information Management* 23 (3): 378–406. doi:10.1108/17410391011036120.

Dragoni, Nicola, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin ja Larisa Safina. 2017. “Microservices: yesterday, today, and tomorrow”. Teoksessa *Present and Ulterior Software Engineering,* 195–216. Springer.

Ehie, Ike C., ja Mogens Madsen. 2005. “Identifying critical issues in enterprise resource planning (ERP) implementation”. Current trends in ERP implementations and utilisation, *Computers in Industry* 56 (6): 545–557. ISSN: 0166-3615. doi:https://doi.org/10.1016/j.compind.2005.02.006.

Fowler, Martin. 2014. *Microservices.* As WWW page, https://martinfowler.com/articles/microservices.html, referenced 25.2.2018.

Jaramillo, D., D. V. Nguyen ja R. Smart. 2016. “Leveraging microservices architecture by using Docker technology”. Teoksessa *SoutheastCon 2016,* 1–5. Maaliskuu. doi:10.1109/SECON.2016.7506647.

Killalea, Tom. 2016. “The Hidden Dividends of Microservices”. *Queue* (New York, NY, USA) 14, numero 3 (toukokuu): 10:25–10:34. ISSN: 1542-7730. doi:10.1145/2956641.2956643.

Klaus, Helmut, Michael Rosemann ja Guy G. Gable. 2000. “What is ERP?” *Information Systems Frontiers* 2, numero 2 (elokuu): 141–162. ISSN: 1572-9419. doi:10.1023/A:1026543906354.

Laukkanen, Sanna, Sami Sarpola ja Petri Hallikainen. 2007. “Enterprise size matters: objectives and constraints of ERP adoption”. *Journal of Enterprise Information Management* 20 (3): 319–334. doi:10.1108/17410390710740763. https://doi.org/10.1108/17410390710740763.

Lehrer, Mark. 2006. “Two types of organizational modularity: SAP, ERP product architecture and the German tipping point in the make/buy decision for IT services”. *Knowledge Intensive Business Services: Organizational Forms and National Institutions:* 187–204.

Loh, T. C., ja S. C. L. Koh*. 2004. “Critical elements for a successful enterprise resource planning implementation in small-and medium-sized enterprises”. *International Journal of Production Research* 42 (17): 3433–3455. doi:10.1080/00207540410001671679.

Merkel, Dirk. 2014. “Docker: Lightweight Linux Containers for Consistent Development and Deployment”. *Linux J.* (Houston, TX) 2014, numero 239 (maaliskuu). ISSN: 1075-3583. http://dl.acm.org/citation.cfm?id=2600239.2600241.

Muscatello, Joseph R., Michael H. Small ja Injazz J. Chen. 2003. “Implementing enterprise resource planning (ERP) systems in small and midsize manufacturing firms”. *International Journal of Operations & Production Management* 23 (8): 850–871. doi:10.1108/01443570310486329.

Olsen, Kai A., ja Per Sætre. 2007. “ERP for SMEs is proprietary software an alternative?” *Business Process Management Journal* 13 (3): 379–389. doi:10.1108/14637150710752290.

Zach, Ondrej, ja Bjorn Erik Munkvold. 2012. “Identifying reasons for ERP system customization in SMEs: a multiple case study”. *Journal of Enterprise Information Management* 25 (5): 462–478.