Mikropalveluarkkitehtuurin liiketoiminnalliset hyödyt ja haasteet eri organisaatiokonteksteissa

Turun yliopisto
Tietotekniikan laitos
TkK-tutkielma
Tietotekniikka
Toukokuu 2026
Eero Antila

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -järjestelmällä.

TURUN YLIOPISTO
Tietotekniikan laitos

Eero Antila: Mikropalveluarkkitehtuurin liiketoiminnalliset hyödyt ja haasteet eri organisaatiokonteksteissa

TkK-tutkielma, 24 s.
Tietotekniikka
Toukokuu 2026

Jokaisen ohjelmiston taustalla on joukko teknisiä päätöksiä, jotka määrittävät, miten ohjelmiston eri osat on jaettu, miten ne kommunikoivat keskenään ja miten järjestelmä vastaa sille asetettuihin vaatimuksiin. Tätä taustalla olevaa rakennetta kutsutaan ohjelmistoarkkitehtuuriksi. Mikropalveluarkkitehtuuri on moderni ohjelmistoarkkitehtuurimalli, jossa sovellus jaetaan pieniin, itsenäisesti toimiviin palveluihin. Tässä tutkielmassa tarkastellaan mikropalveluarkkitehtuurin liiketoiminnallisia hyötyjä ja haasteita sekä arkkitehtuurin soveltuvuutta erilaisissa organisaatiokonteksteissa.

Kirjallisuuskatsauksen perusteella mikropalveluarkkitehtuurin keskeisimmiksi liiketoiminnallisiksi hyödyiksi tunnistettiin kehityksen ketteryys, skaalautuvuus ja teknologinen joustavuus. Keskeisimmiksi haasteiksi nousivat käyttöönoton alkuinvestoinnit, osaamisvaatimusten laajentuminen ja kokonaiskompleksisuuden kasvu. Liiketoiminnalliset vaikutukset on johdettu kirjallisuudessa esitetyistä teknisistä ominaisuuksista.

Analyysi osoitti, että mikropalveluarkkitehtuurin soveltuvuus ei riipu pelkästään organisaation koosta, vaan siihen vaikuttavat organisaation resurssit, osaaminen, järjestelmän kehitysvaihe ja kyky toteuttaa tarvittava kulttuurimuutos. Mikropalveluarkkitehtuuri soveltuu parhaiten suuryrityksille, joilla on resurssit ja rakenteet sen tukemiseen. Pk-yrityksessä se on harkittava vaihtoehto järjestelmän kasvaessa, kun taas startupeille monoliittinen arkkitehtuuri on tyypillisesti perustellumpi valinta.

Asiasanat: mikropalveluarkkitehtuuri, ohjelmistoarkkitehtuuri, liiketoiminnalliset hyödyt, organisaatiomallit

Sisällys

1 Johdanto

Ohjelmistot ovat nykyään keskeinen osa lähes jokaisen henkilön digitaalista toimintaa. Monet päivittäiset asiat, kuten pankkipalvelut, verkkokaupat ja sosiaalisen median alustat, toimivat käyttäjälle yhtenä sujuvana ohjelmana. Näiden ohjelmistojen taustalla on joukko teknisiä päätöksiä siitä, miten ohjelmisto on rakennettu ja organisoitu. Ohjelmistojen taustalla olevaa rakennetta kutsutaan ohjelmistoarkkitehtuuriksi. Se määrittää, miten ohjelmiston eri osat on jaettu, miten ne kommunikoivat keskenään ja miten järjestelmä vastaa sille asetettuihin vaatimuksiin. Carriere ym. (2010) [1] esittävät, että arkkitehtuuripäätökset tulisi nähdä liiketoiminnallisina päätöksinä. Eri valinnat johtavat erilaisiin muutos- ja integraatiokustannuksiin tulevaisuudessa, joiden vaikutusta tulisi arvioida osana päätöksentekoa. [1]

Ohjelmistokehittäminen on viime vuosikymmeninä muuttunut teknologian kehittymisen myötä. Baškarada ym. (2020) [2] toteavat, että dynaaminen teknologia- ja liiketoimintaympäristö sekä digitaalisesti valveutuneet asiakkaat pakottavat organisaatioita jatkuvasti innovoimaan ja päivittämään ohjelmistopalveluitaan. [2] Nopeasti muuttuvan ympäristön myötä monet organisaatiot ovat ottaneet käyttöön DevOps-käytänteitä, joissa ohjelmiston kehitys (engl. Development) ja ylläpito (engl. Operation) yhdistyvät yhteisen vastuun alle. Pilvipalveluiden yleistyminen on puolestaan mahdollistanut ohjelmistojen ajamisen ja skaalaamisen ilman omia fyysisiä laitteistoinfrastruktuureja.

Tässä kandidaatintutkielmassa tarkastellaan mikropalveluarkkitehtuuria organisaatioiden näkökulmasta, minkä liiketoimintaan kuuluu ohjelmistotuotteiden tai digitaalisten palveluiden kehittäminen. Tutkielma on toteutettu kirjallisuuskatsauksena. Tarkasteltavat hyödyt ja haasteet perustuvat kirjallisuudessa esitettyihin ominaisuuksiin, joista liiketoiminnalliset vaikutukset on johdettu. Tutkielmassa vastataan seuraaviin tutkimuskysymyksiin:

TK1: Millaisia liiketoiminnallisia hyötyjä mikropalveluarkkitehtuurille on esitetty kirjallisuudessa?

TK2: Millaisia liiketoiminnallisia haasteita mikropalveluarkkitehtuurille on esitetty kirjallisuudessa?

TK3: Miten organisaatiomalli vaikuttaa näiden hyötyjen ja haasteiden realisoitumiseen?

Tutkimuskysymyksiin TK1 ja TK2 vastataan luvussa 3 kirjallisuuden perusteella. Tutkimuskysymykseen TK3 vastataan luvussa 4 analysoimalla aiempien lukujen havaintoja. Tutkimuskysymyksiin liittyvä aineisto on haettu valtaosin IEEE Xplore, ACM Digital Library ja Taylor & Francis -tietokannoista. Hakulausekkeena on toiminut *"microservice architecture"AND business AND (benefits OR challenges OR "economic impact"OR agility OR scalability)* Lisäksi osa lähteistä on löytynyt alkuperäisten lähteiden lähdeluetteloiden kautta.

2 Ohjelmistoarkkitehtuuri ja liiketoimintamallit

2.1 Ohjelmistoarkkitehtuurin merkitys

Ohjelmistoarkkitehtuuri toimii keskeisenä yhdistäjänä organisaation teknisten ratkaisujen ja liiketoimintastrategian välillä. Clements ja Bass (2010) [3] esittävät käsitteen arkkitehtuurisesti merkittävät vaatimukset (engl. Architecturally Significant Requirements, ASR), jolla viitataan niihin teknisiin vaatimuksiin, joilla on tärkeä asema ohjelmiston arkkitehtuurin määrittämisessä. Merkittävä lähde vaatimuksille tulee suoraan organisaation liiketoimintatavoitteista, jotka ovat johtaneet järjestelmän kehitykseen. [3] Suunnitteluvaiheessa oikeiden vaatimusten tunnistamisella sekä priorisoinnilla voidaan varmistaa että ohjelmistoarkkitehtuuri tukee strategisia tavoitteita eikä rajoita niiden saavuttamista. Hyvin suunniteltu arkkitehtuuri mahdollistaa nopean reagoinnin asiakasvaatimusten ja teknologisten trendien muutoksiin, mikä tukee innovointia sekä mahdollistaa uusien tuotteiden ja ominaisuuksien nopeaa kehittämistä. [4] Näin ohjelmistoarkkitehtuuri toimii teknisen rakenteen lisäksi myös strategisena välineenä, jonka tarkalla suunnittelulla ja toteutuksella ylläpidetään kilpailukykyä.

Ohjelmistoarkkitehtuurin vaikutus ei rajoitu pelkästään strategiseen ja teknologiseen päätöksentekoon, vaan siinä tehdyt päätökset näkyvät suoraan organisaation

kyvyssä kehittää ja ylläpitää ohjelmistojaan. Carriere ym. (2010) [1] esittävät, että arkkitehtuuripäätöksiä voidaan arvioida samalla tavalla kuin muita liiketoiminnallisia investointeja. Heidän viitekehyksensä avulla arkkitehtuuriratkaisut voidaan liittää osaksi organisaation taloudellista päätöksentekoa. [1] Arkkitehtuurisuunnittelu on siten prosessi, jossa pyritään löytämään tasapaino kehitys- ja ylläpitokustannusten, sekä saavutettavien liiketoimintahyötyjen välillä. Hyvin suunniteltu arkkitehtuuri mahdollistaa resurssien tehokkaan käytön ja tukee ohjelmiston pitkäikäisyyttä.

Epäjohdonmukaisuus arkkitehtuuriratkaisuissa voi johtaa teknisen velan (engl. technical debt) kasvuun, jolloin lyhyen aikavälin päätökset heijastuvat tulevaisuudessa merkittävinä ylläpito- ja muutoskustannuksina. Nord ym. (2012) [5] määrittelee arkkitehtuurisen teknisen velan tilanteeksi, jossa valitaan nopeat ratkaisut laatuvaatimusten kustannuksella. [5] Teknistä velkaa syntyy esimerkiksi, kun toteutetaan muutoksia ilman kokonaisarkkitehtuurin huomioon ottamista. Huonojen päätösten vaikutukset voivat näkyä vasta tulevaisuudessa, kun uusia toiminnallisuuksia kehitetään. Teknisen velan ottaminen kasvattaa järjestelmän monimutkaisuutta ja vaikeuttaa sen kehittämistä tulevaisuudessa. Pitkäjänteinen lähestymistapa auttaa välttämään kalliita uudelleenrakennusprojekteja ja varmistamaan, että arkkitehtuuri tukee tavoitteita myös pitkässä juoksussa.

2.2 Mikropalveluarkkitehtuuri

Ohjelmistoarkkitehtuurien kehitys on viime vuosikymmeninä kulkenut kohti yhä hajautetumpia ja modulaarisempia ratkaisuja. Perinteisessä ohjelmistokehityksessä on hyödynnetty monoliittista arkkitehtuuria, jossa sovellus muodostuu yhdestä kokonaisuudesta. Käyttöliittymä, toimintalogiikka ja tietokanta sisältyvät samaan koodipohjaan ja ne julkaistaan yhtenä kokonaisuutena. [6] Rakenteen etuna on yksinkertainen arkkitehtuuri, joka on nopea ottaa käyttöön. Tämä tekee siitä houkuttelevan

vaihtoehdon pienissä projekteissa, joissa kehitystä ja ylläpitoa voi hoitaa jopa yksittäinen henkilö. [6, 7] Heikkoudet tulevat kuitenkin esiin sovellusten kasvaessa ja liiketoimintaympäristön muuttuessa. Kaikki osat järjestelmästä ovat sidoksissa toisiinsa, minkä takia pienillä muutoksilla voi olla ennalta-arvaamattomia vaikutuksia muihin toimintoihin. Sovelluksen yksittäisiä osia ei voida päivittää tai julkaista erikseen, vaan koodia muokatessa koko sovellus on käännettävä ja julkaistava uudelleen. Tämä tekee kehitysprosessista raskaan ja lisää virheriskiä. [6] Järjestelmän laajentuessa yksittäisten muutosten vaikutusalue suurenee entisestään. Vaikutusalueiden laajentuminen heikentää ylläpidettävyyttä ja hidastaa kehityssyklejä, mikä vaikeuttaa reagointia nopeasti muuttuviin liiketoimintatarpeisiin. [7] Digitalisaation, pilvipalveluiden ja jatkuvan kehittämisen vaatimusten myötä monoliittinen lähestymistapa on väistymässä modernimpien arkkitehtuurimallien, kuten mikropalveluarkkitehtuurin, tieltä.

Mikropalveluarkkitehtuuri (engl. Microservice Architecture) on moderni ohjelmistoarkkitehtuurimalli, joka pyrkii ratkaisemaan monoliittisen arkkitehtuurin rajoitteita. Keskeisenä ajatuksena on jakaa sovellus pieniin, itsenäisesti toimiviin palveluihin, jotka vastaavat kukin selkeästi rajatusta yhdestä toiminnosta. [2] Blinowski ym. (2022) esittää kolme keskeistä periaatetta, jotka kuvaavat mikropalveluiden luonnetta. Ensimmäisenä on yksittäisen vastuun periaate (engl. single responsibility per service), jonka mukaan jokaisella yksittäisellä palvelulla on vain yksi vastuualue. Tämä tekee palveluista helpommin ylläpidettäviä ja vähentää monimutkaisuutta. Toisena on autonomisuuden periaate, joka korostaa, että kukin mikropalvelu on itsenäinen ja vastuussa omasta toiminnastaan. Yksittäinen palvelu sisältää kaikki tarvitsemansa riippuvuudet, kuten kirjastot, suoritysympäristöt ja mahdolliset tietokannat. Tämä mahdollistaa yksittäisen palvelun päivittämisen vaikuttamatta muihin palveluihin. Kolmas periaate on palveluiden pitäminen itsenäisinä ja ensisijaisina toimijoina (engl. first-class citizens). Tämä tarkoittaa, että palvelut tarjoavat

rajapinnan muiden palveluiden käyttöön, mutta niiden sisäinen logiikka, käytetty ohjelmointikieli tai arkkitehtuuri pysyy piilotettuna. Tällöin palveluiden keskenäinen vuorovaikutus tapahtuu vain määriteltyjen rajapintojen kautta, mikä edistää löyhää kytkentää ja selkeyttää arkkitehtuuria. [6]

Yksittäinen sovellus voi koostua jopa sadoista mikropalveluista, jotka muodostovat yhdessä kokonaisuuden. Palvelut kommunikoivat keskenään kevyiden rajapintojen kautta, kuten HTTP tai REST. [8] Yllä mainitun autonomisuuden takia kehittäminen, testaaminen ja käyttöön ottaminen voidaan tehdä toisista palveluista riippumatta. Tämä nopeuttaa kehitystä ja mahdollistaa tiheämmät julkaisut. [2] Yhteiset rajapintastandardit mahdollistavat eri ohjelmointikielten tai ohjelmistokehysten (engl. Software Framework) käytön. [9] Mikropalveluarkkitehtuurin tekniset periaatteet luovat pohjan sille, miten sovelluksia kehitetään ja ylläpidetään hajautetussa ympäristössä. Arkkitehtuuri vaikuttaa myös siihen, miten kehitystiimit organisoidaan ja miten järjestelmän kapasiteettia laajennetaan.

2.3 Organisaatiomallit

Yrityksen koko, rakenne ja johtamismalli heijastuvat suoraan tapaan, miten työtä organisoidaan ja kehityshankkeita toteutetaan. Henkilömäärä, roolien erikoistuminen, päätöksenteon nopeus ja resurssien saatavuus vaihtelevat keskenään suuresti erilaisten organisaatiomallien kesken. [10] Nämä erot heijastuvat myös siihen, miten ohjelmistokehitystä toteutetaan ja millaisia arkkitehtuurisia ratkaisuja pystytään tukemaan.

Startup-yritykset

Startup-yritykset ovat uusia, dynaamisia ja kasvuun tähtääviä yrityksiä, jotka pyrkivät kehittämään uusia tuotteita tai palveluita ratkaistakseen ongelmia. Muodoltaan ketterät startupit yrittävät innovatiivisilla ideoillaan kilpailla isompien yri-

tysten kanssa tai luoda täysin uusia markkinoita. Abdelghani (2024) [10] kuvailee tyypillisiksi piirteiksi pienen henkilöstökoon, joustavat ja päällekkäiset roolit, sekä korkean riskin liiketoimintamallit. [10] Pieni tiimikoko mahdollistaa matalan hierarkian, joustavan työjaon ja nopean päätöksenteon. Tämä mahdollistaa ympäristön, jossa kokeilukulttuuri ja nopea iterointi ovat olennaisessa roolissa.

Startupien toimintaa määrittää kuitenkin vahvasti rahoituksen saatavuus. Varhaisessa vaiheessa resurssit voivat olla hyvin rajoittuneita, joka vaikeuttaa yrityksen kasvattamista ja ohjaa toimintaa lyhyen aikavälin tavoitteisiin, kuten ideoiden validoimiseen. Tätä tukee väite, että 50 % startupeista epäonnistuu ensimmäisen viiden vuoden aikana. [10] Varhaisen vaiheen yrityksille asetetaan korkea paine todistaa oma ideansa jatkaakseen toimintaa, minkä takia voidaan joutua tasapainottelemaan nopean toteutuksen ja teknisen pitkäjänteisyyden välillä. [10] Tämä voi tarkoittaa kompromisseja esimerkiksi teknisen velan suhteen. Ratkaisuja rakennetaan niin, että ne mahdollistavat nopean etenemisen, mutta ne eivät välttämättä ole pitkällä aikavälillä optimaalisia.

Pk-Yritykset

Pienet ja keskisuuret yritykset, eli pk-yritykset sijoittuvat startupien ja suuryritysten väliin. Pk-yrityksillä on tyypillisesti vakiintuneemmat resurssit käytössä kuin startupeissa, minkä myötä yrityksen toiminta ja investoinnit voivat olla suunnitelmallisempaa. Keskittyminen on usein liiketoiminnan kasvattamisessa, joka tapahtuu jo validoidun idean ympärillä.

Henkilöstömäärissä mitattuna työntekijöitä voi olla jopa 250, minkä vuoksi organisaatiorakenteiden täytyy olla muodollisempia. [10] Vaikka eri roolit voivat olla selkeämmin jäsennelty, hierarkia pysyy vielä suhteellisen matalana ja päätöksenteko ketteränä. [11] Vastuita ja työtehtäviä voidaan jakaa eri rooleille ja mahdollisille esihenkilötasoille, vaikka organisaatio säilyy edelleen suhteellisen matalana ja epämuo-

dollisella vuorovaikutuksella on tärkeä rooli yrityksen sisällä. Myös ohjelmistokehityksen näkökulmasta voidaan nähdä eriytyneempiä tiimejä, jotka kukin vastaavat omista vastuualueistaan. [12]

Organisaatioiden kasvun myötä joudutaan vähitellen selkeyttämään johtamista ja kehittämään systemaattisempia prosesseja. Erilaisten johtamisen ja tiedonhallinnan työkalujen käyttö lisääntyy yrityskoon suurentuessa. [13] Työkalujen avulla ennustettavuus parantuu ja toiminnasta tulee paremmin hallittavaa, mutta samalla se vähentää ketteryyttä, joka voidaan nähdä etuna pienemmillä toimijoilla. [14] Tämä näkyy myös siirtymänä yksittäisistä avainhenkilöistä tiimeihin, joissa suunnittelun ja koordinoinnin tärkeys korostuu aikaisempaa enemmän.

Suuryritykset

Suuryrityksille on tyypillistä huomattava henkilöstömäärä ja monitasoinen organisaatiorakenne. Dobre (2016) [11] toteaa, että suuryrityksissä muodolliset säännöt, ohjeistukset ja kontrollointi korostuvat, mikä erottaa ne pienemmistä, usein joustavammista organisaatioista. [11] Tämä tarkoittaa siis useita johtoporrastasoja ja erillisiä osastoja, kuten talous, markkinointi sekä tuotekehitys. Nämä muodostavat suuremman kokonaisuuden, jossa vastuut ovat rajattuja kullekin osastolle.

Suurissa teknologiayrityksissä, kuten Applella, rakenne pohjautuu usein eriytyneisiin osastoihin, kuten ohjelmistokehitykseen, laitteistosuunnitteluun ja operaatioihin. [15] Näillä kokonaisuuksilla on omat johtajansa, jotka raportoivat eteenpäin seuraaville johtajille. Tämä mahdollistaa syvällisen osaamisen kehittämistä, mutta lisää koordinoinnin tarvetta eri yksiköiden välillä. [15] Myös ohjelmistokehityksen näkökulmasta suuryritykset organisoivat kehitystä useisiin tiimeihin, joilla on omat tarkasti rajatut vastuualueet. Näitä vastuualueita voi olla esimerkiksi alijärjestelmät tai kokonaan omat teknologiat. Šāblis ym. (2021) [16] osoittavat, että vaikka tiimeil-

lä on omat osa-alueet, työ on jatkuvasti riippuvaista muiden tiimien toteuttamista komponenteista ja rajapinnoista. [16]

Taulukko 2.1: Organisaatiomallien vertailu, tekijän oma kooste, perustuu lähteisiin [10, 11, 13, 16]

Ominaisuus	**Startup**	**Pk-yritys**	**Suuryritys**
Hierarkia	Matala, epämuodollinen	Kohtalainen, jäsentyvä	Selkeä ja hierarkinen
Päätöksenteko	Nopea ja joustava	Ketterä, mutta koordinoitu	Hidas ja formaali
Resurssit	Vähäiset	Kohtuulliset	Laajat
Prosessit	Kevyet ja muuttuvat	Osittain vakiintuneet	Raskaasti standardoidut
Roolit	Moniosaajuus	Selkeytyvät roolit	Vahva erikoistuminen
Teknologinen kehitys	Kokeileva ja nopea	Suunnitelmallinen mutta ketterä	Vakaa ja riskienhallintaan perustuva

Yhteenvetona taulukko 2.1 esittää, miten startup-ympäristön epämuodollinen rakenne ja ketterä kehitys muuttuvat pk-yrityksissä kohti standardoituja prosesseja ja selkeitä rooleja. Suuryrityksissä korostuvat hierarkkinen ohjaus, standardoidut prosessit ja erikoistuminen omille osa-alueilleen. Ohjelmistokehityksen näkökulmasta eri organisaatiomallit mahdollistavat erilaiset lähtökohdat tiimien autonomialle, teknologisten ratkaisujen valinnalle ja arkkitehtuurin hallittavuudelle. On kuitenkin tärkeää huomioida, että yksittäisen yritykset voivat poiketa merkittävästi kyseisestä mallista. Taulukossa esitetyt ominaisuudet kuvaavat tutkimuskirjallisuudessa esitettyjä yleistyksiä startup-, pk- ja suuryrityksistä.

3 Mikropalveluarkkitehtuurin hyödyt ja haasteet

3.1 Liiketoiminnalliset hyödyt

Ketteryys ja nopeampi tuotekehitys

Teknisenä lähtökohtana mikropalveluarkkitehtuurissa on järjestelmän rajaaminen omiin itsenäisiin palveluihin. Toisistaan riippumattomuuden ansiosta palveluita voidaan kehittää, testata ja julkaista erillään. [8] Tämän modulaarisen rakenteen takia muutokset rajautuvat myös selkeämmin pienempiin osa-alueisiin. Löyhän kytkennän takia palveluiden keskinäisten vaikutusten tulisi pysyä vähäisinä. [8]

Arkkitehtuurin merkitys korostuu ohjelmiston ketteryyden ja kehitysnopeuden näkökulmasta. Teknisen velan ottaminen arkkitehtuurisissa ratkaisuissa heikentää ohjelmiston ketteryyttä ja hidastaa tuotekehitystä. [17] Huonot päätökset lisäävät kompleksisuutta ja ylläpitokustannuksia, mikä johtaa kasvaviin vaikeuksiin ohjelmiston kehittämisessä. Velka voi kasvaa itseään vahvistavassa kierteessä, jolloin heikot päätökset kasaantuvat ja heikentävät ylläpidettävyyttä edelleen. [17] Arkkitehtuuriratkaisut vaikuttavat siis suoraan organisaation kehitysnopeuteen ja kykyyn muokata tuotteita ja palveluita. Mikäli ohjelmisto ei ole hallittava, muutosten läpivienti voi hidastua ja strategisten uudistusten toteuttaminen vaikeutuu.

Mikropalveluarkkitehtuurin keskeiseksi hyödyksi mainitaan kehittämisen ketteryyden parantuminen. [2] Itsenäisesti julkaistavat palvelut mahdollistavat tiheämmät julkaisut ja nopeammat muutosten käyttöönotot. [8] Organisaation näkökulmasta tämä tarkoittaa lyhyempia kehitys- ja julkaisusyklejä, nopeampaa reagointia asiakaspalautteeseen sekä parempaa kykyä mukautua muutoksiin. Ketteryys tukee myös suoraan organisaation kilpailuetua, sillä nopeampi reagointikyky auttaa suoriutumista muuttuvilla markkinoilla. Nopean tuotekehityksen avulla pystytään seuraamaan trendejä ja hyödyntämään uusia teknologioita.

On kuitenkin tärkeää tunnistaa, ettei arkkitehtuurivalinta itsessään takaa kehityksen nopeutta tai ketteryyttä. [17] Tekninen joustavuus muuntuu liiketoiminnalliseksi ketteryydeksi vain silloin, kun arkkitehtuuria johdetaan systemaattisesti, kokonaisuutta tarkastellaan pitkäjänteisesti ja velan kertymistä hallitaan tietoisesti.

Skaalautuvuus ja kustannustehokkuus pitkällä aikavälillä

Järjestelmien skaalautuvuus on tärkeää ottaa huomioon jo hyvin varhaisessa kehityksen vaiheessa. Tietojärjestelmissä skaalautuvuutta voidaan tarkastella kahdesta eri ulottuvuudesta: pystysuora skaalaus (engl. vertical scaling, scaling up) ja vaakasuora skaalaus (engl. horizontal scaling, scaling out). Pystysuorassa skaalauksessa tehoa kasvatetaan päivittämällä yhtä palvelinta tehokkaammaksi. Vaakasuorassa skaalauksessa järjestelmän tehoa kasvatetaan lisäämällä uusia palvelimia rinnakkain. [6]

Mikropalveluiden itsenäisyyden takia järjestelmä voidaan skaalata vaakasuorassa ulottuvuudessa. [2] Tämä tarkoittaa sitä, että eniten kuormittuneille palveluille voidaan antaa lisää laskentatehoa monistamalla niiden instansseja useammalle palvelimelle. Tällöin pullonkauloja voidaan ratkaista ilman, että koko järjestelmän suorituskykyä täytyy nostaa. Tämä mahdollistaa myös keskittymisen palveluihin, jotka tuottavat eniten liiketoiminta-arvoa. Monoliittisessa ohjelmassa skaalaaminen

tapahtuu pystysuoraan. Tällöin vähemmän kuormitusta aiheuttavat osat yliskaalautuvat turhaan. [6] Monoliittisia sovelluksia pystytään skaalaamaan myös vaakasuorasti monistamalla sovellus usealle palvelimelle. Tällöin liikenne jaettaisiin palvelinten kesken kuormituksen mukaan. Tämä ei kuitenkaan poista ongelmaa vähemmän kuormitusta aiheuttavien osien yliskaalautuvuudesta, koska koko ohjelmainstanssi ajettaisiin usealla palvelimella.

Useissa tutkimuksissa käsitellään edellä mainitun skaalautuvuuden vaikutuksia laskentatehon ja kustannuksien kannalta. Okrój ja Jatkiewicz (2023) [7] havaitsivat, että mikropalveluarkkitehtuurin mahdollistama kapasiteetin lisääminen pienissä askeleissa voi vähentää kustannuksia verrattuna monoliittisen sovelluksen vertikaaliseen skaalaamiseen. [7] Kustannushyödyt eivät kuitenkaan ole itsestäänselviä. Saman havainnon teki Blinowski ym. (2022) [6] tutkimuksessaan, jossa todettiin vertikaalisen skaalauksen olevan kustannustehokkaampaa tiettyyn pisteeseen asti. [6] Vertikaalinen skaalaus on kuitenkin luonteeltaan rajallinen, koska yksittäistä palvelinta ei voi kasvattaa rajattomasti. Okrój ja Jatkiewicz (2023) [7] totesivat myös, että heidän käyttämässään Azure-ympäristössä vertikaalinen skaalautuminen tapahtui aina suurina hyppäyksinä seuraavaan palvelinluokkaan. Tutkijat esittivät esimerkin tilanteesta, jossa monoliittisen sovelluksen kapasiteetin kasvattaminen vaati siirtymistä seuraavaan palvelinluokkaan, joka miltein kaksinkertaisti kuukausikustannukset. Vastaava lisäys olisi voitu toteuttaa mikropalveluarkkitehtuurissa lisäämällä pienempiä palveluinstansseja, jolloin kokonaiskustannukset olisivat olleet noin 25 % pienemmät. [7]

Vikasietoisuus ja teknologinen joustavuus

Merkittävänä etuna mikropalveluissa on vikasietoisuus. Yhdessä palvelussa oleva vika ei välttämättä kaada koko järjestelmää. [2] Tämän seurauksena voidaan usein tarjota heikentynyttä, mutta edelleen käyttökelpoista palvelua, vaikka osa ominai-

suuksista ei toimisi. Monoliitissa häiriöt johtavat helpommin koko järjestelmän kaatumiseen. Vikasietoisuutta ei kuitenkaan voi pitää oletuksena, vaan se vaatii huolellista suunnittelua ja toteutusta arkkitehtuurissa. [18] Mikropalveluarkkitehtuuri mahdollistaa myös tarkempien palvelutasosopimusten (engl. Service Level Agreement, SLA) laatimisen. [18] Palvelua tarjoava yritys pystyy erittelemään yksittäisten ominaisuuksien tai palveluiden saatavuutta ja käytettävyyttä ilman, että samaa lupausta täytyy tehdä koko järjestelmälle.

Baškarada ym. (2020) [2] nostavat esiin yhdeksi merkittäväksi eduksi heterogeenisen teknologiapinon käytön. [2] Kuten luvussa 2.2 todettiin, mikropalveluarkkitehtuuri mahdollistaa eri ohjelmointikielten käytön eri palveluissa. Tämä mahdollistaa uusien teknologioiden ja innovaatioiden kokeilua ilman, että koko järjestelmää täytyy muuttaa. Monoliittisessa ohjelmassa mahdollisuudet uusien teknologioiden hyödyntämiseen ovat huomattavasti rajoitetummat. Tämä näkyy siinä, että monet tahot ovat alkaneet lisätä sovelluksiinsa mikropalveluita uusien toiminnallisuuksien toteuttamiseksi. Vitharana ja Daya (2024) [18] antavat esimerkkinä Yhdysvaltalaisen pankin ja lentoyhtiön, jotka uudistavat palveluitaan siirtymällä asteittain kohti mikropalveluarkkitehtuuria. [18]

3.2 Liiketoiminnalliset haasteet

Alkuinvestoinnit ja käyttöönoton haasteet

Mikropalveluarkkitehtuurin käyttöönotto edellyttää usein merkittävämpiä alkuinvestointeja verrattuna monoliittiseen arkkitehtuuriin. Hajautettujen mikropalveluiden tehokas kehittäminen vaatii tyypillisesti DevOps-käytäntöjä, kuten jatkuvaa integraatiota ja toimitusta (engl. continuous integration and continuous deployment, CI/CD). [2] Palveluiden käyttöönotto ja monitorointi vaativat sen mahdollistavaa infrastruktuuria, jossa tyypillisesti tukeudutaan pilvipohjaisiin ympäristöihin.

Teknisen infrastruktuurin lisäksi mikropalveluarkkitehtuuri edellyttää merkittäviä panostuksia suunnitteluun. Palveluiden rajojen määrittely, rajapintojen suunnittelu sekä palveluiden välisten riippuvuuksien hallinta ovat keskeisessä roolissa. [2, 7] Huolellinen suunnittelu ja toteutus ovat keskeisessä asemassa mahdollisten hyötyjen toteutumisessa. Jos palveluiden rajat määritellään huonosti tai kehitysprosessit eivät tue hajautetun järjestelmän hallintaa, arkkitehtuurin monimutkaisuus kasvaa ilman oletettujen hyötyjen saamista. [18] Tämä voi käytännössä näkyä niin, että ohjelmistosta alkaa muodostua useista pienistä monoliiteista koostuva kokonaisuus.

Taloudellisesta näkökulmasta mikropalveluarkkitehtuurin vaatimukset näkyvät erityisesti infrastruktuuri-, työkalu- ja suunnittelukustannuksina. Infrastruktuurin rakentaminen sekä palveluiden monitorointiin ja hallintaan tarvittavat työkalut lisäävät alkuvaiheen työmäärää. Tämän vuoksi yksinkertaisempi monoliittinen arkkitehtuuri voi tarjota taloudellisesti kevyemmän ratkaisun erityisesti silloin, kun kehitystiimien ja infrastruktuurin mittakaava on rajallinen. [7]

Kompleksisuus ja ylläpitohaasteet

Vaikka yksittäinen mikropalvelu voi olla rakenteeltaan yksinkertainen, koko järjestelmä muodostuu useista palveluista sekä niiden välisistä rajapinnoista ja kommunikointisuhteista. [2] Kokonaiskompleksisuudeltaan järjestelmä voi siis olla hyvinkin monimutkainen. Hajautettu rakenne aiheuttaa teknisiä haasteita, kuten palveluiden välisen kommunikoinnin hallintaa, integraatiotestauksen vaikeutumista sekä hajautettujen transaktioiden toteuttamista. [7] Tutkimuksissa todetaan, että nämä lisäävät teknistä monimutkaisuutta verrattuna monoliittiseen arkkitehtuuriin. [8] Kokonaisuuden ymmärtäminen ja hallinta vaikeutuvat palveluiden määrän kasvaessa, mikä lisää koordinointitarvetta kehitystiimien välillä.

Mikropalveluarkkitehtuuri vaikuttaa myös merkittävästi operointi- ja ylläpitotöihin. Kuten luvussa 2.2 todettiin, jokainen palvelu on oma yksikkönsä, jolla on omat

riippuvuutensa ja joka vaatii oman käyttöönottonsa. Tämän seurauksena jokainen palvelu vaatii myös omaa ylläpitoaan, kuten lokienhallintaa ja monitorointia. [2] Palveluiden välinen kommunikointi tapahtuu verkon yli, mikä tuo mukanaan viiveisiin, verkon luotettavuuteen ja virheiden käsittelyyn liittyviä haasteita. Mikropalveluista koostuvien järjestelmien ylläpito edellyttääkin laajempia valvonta- ja lokienhallintaratkaisuja, jotta palveluiden toimintaa ja keskinäisiä riippuvuuksia voidaan seurata. [8]

Jos suunnittelu- tai toteutusvaiheessa asetettuja rajoja tai vastuita ei noudateta, voi riippuvuudet palveluiden välillä johtaa tilanteeseen, jossa ylläpito vaikeutuu ja muutosten tekeminen hidastuu. Tällöin alkaa kerääntymään teknistä velkaa, joka hidastaa tulevaa kehitystyötä ja nostaa kustannuksia. [18] Vaikka mikropalveluarkkitehtuuri pyrkii parantamaan modulaarisuutta ja joustavuutta, ilman selkeitä käytäntöjä se voi johtaa samantyyppisiin ylläpitohaasteisiin kuin suurissa monoliittisissä järjestelmissä.

Osaamisvaatimukset ja kulttuurimuutos

Toisin kuin monoliittisissa ympäristöissä, joissa kehitys ja operointi ovat tyypillisesti erillisiä vastuualueita, mikropalveluarkkitehtuuri mahdollistaa tiimien organisoinnin liiketoimintakyvykkyyksien mukaan. Palvelun kehittäneet tiimit vastaavat myös sen ylläpidosta koko elinkaaren ajan. [18] Tiimien autonomiaa pidetäänkin mikropalveluarkkitehtuurin yksinä keskeisimmistä organisatorisista hyödyistä, mutta se asettaa samalla organisaatiolle merkittäviä vaatimuksia.

Tiimien itsenäisyyden takia osaamisvaatimukset kasvavat merkittävästi. Mikropalveluarkkitehtuuri edellyttää tiimeiltä osaamista hajautetuista järjestelmistä, pilvi-infrastruktuurista, automaatiosta, tietoturvasta ja monitoroinnista. [2] Näitä taitoja ei keskitetä erilliseen tukirooliin, vaan niiden osaajia tarvitaan tiimien sisällä. Tiimien määrästä riippuen organisaatio tarvitsee henkilöstöönsä siis enemmän

tiettyjen osa-alueiden osaajia tai laaja-alaisempia osaajia. Laaja-alaisia osaajia on vaikea löytää, ja osaamisen kehittäminen vie aikaa. [18] Tämä vaikuttaa suoraan henkilöstön rekrytointi- ja koulutuskustannuksiin.

Malli, jossa vastuu operoinnista ja ylläpidosta on sen rakentaneella tiimillä, vaatii ohjelmistokehittäjiltä laajempaa vastuunottoa kuin perinteisessä mallissa. Autonomisten tiimien toiminta edellyttää valmiutta poikkifunktionaaliseen työhön ja laajennettuihin rooleihin, mikä voi vaatia muutoksia organisaation kannustinrakenteisiin. [18] Baškarada ym. (2020) [2] nostavat esiin, että tiimimalli edellyttää organisaatiolta merkittävää tiimien välistä luottamusta. [2] Tiimimallin rakentaminen ja ylläpitäminen voi olla siis haastavaa, sillä se edellyttää organisaatiolta kulttuuria, jossa laajennettu vastuu ja tiimien välinen luottamus voivat toteutua. Tässä epäonnistuminen voi nostaa organisaation sisäistä monimutkaisuutta ilman vastaavia liiketoiminnallisia hyötyjä.

Toisaalta mikropalveluarkkitehtuuri luo rakenteellisesti edellytykset DevOpskäytänteiden tehokkaalle hyödyntämiselle, sillä arkkitehtuuri ja DevOps ovat tiiviisti kytkeytyneitä. [2] Tiimien sisäinen itsenäinen kehitys, testaus ja käyttöönotto sopivat luontevasti jatkuvaan integraatioon ja toimitukseen. Koska jokainen tiimi vastaa oman palvelunsa koko elinkaaresta, se voi toimia itsenäisesti ilman riippuvuutta muista tiimeistä. Tämä rakenne on käytännössä se, mikä tekee DevOps-periaatteiden soveltamisen mahdolliseksi.

4 Mikropalveluarkkitehtuuri eri organisaatiomalleissa

4.1 Startupit

Startup-yritysten toimintaympäristössä esiintyy nopea iteraatio, resurssien rajallisuus ja liiketoiminnan korkea epävarmuus. Luvussa 2.3 todettiin, että startupit toimivat usein pienillä tiimeillä, joissa roolit ovat päällekkäisiä ja päätöksenteko on ketterää. Nämä piirteet vaikuttavat ensisilmäyksellä suotuisilta lähtökohdilta mikropalveluarkkitehtuurin käyttöönotolle. Käytännössä kuitenkin arkkitehtuurin edut, kuten itsenäisesti julkaistavat palvelut, autonomiset tiimit ja skaalautuva infrastruktuuri, vaativat edellytyksiä, joita varhaisen vaiheen yrityksillä harvoin on.

Luvussa 3.2 kuvattiin mikropalveluarkkitehtuurin käyttöönottoon ja hyödyntämiseen liittyviä alkuinvestointeja, kuten DevOps-käytänteiden omaksuminen, CI/CD-infrastruktuurin rakentaminen ja palveluiden rajojen suunnittelu. [2] Varhaisen vaiheen yrityksen ensisijaisena tavoitteena on idean validoiminen ja liiketoiminnan potentiaalin osoittaminen. Tässä vaiheessa resurssien sitominen arkkitehtuurin kehittämiseen ja ylläpitoon liiketoiminnallisen kasvun sijaan voi muodostua kannattamattomaksi. Lisäksi luvussa 3.2 esitetty osaamisvaatimuksiin liittyvä haaste korostuu startup-yrityksissä pienen henkilöstömäärän takia. Esimerkiksi hajautettujen järjestelmien, pilvi-infrastruktuurin ja tietoturvan hallinta ei rajoitu yksit-

täiseen rooliin, vaan on jokaisen tiimin vastuulla. [2, 18] Vaikka samoja osaamisalueita esiintyy monoliittisissa järjestelmissä, mikropalveluympäristössa niiden hallintaa vaikeutuu palveluiden määrän ja keskinäisten riippuvuuksien kasvaessa.

Toinen keskeinen riski liittyy teknisen velan hallitsemiseen. Luvussa 2.1 todettiin teknisen velan syntyvän tilanteissa, joissa lyhyen aikavälin paineet johtavat ratkaisuihin, jotka eivät kestä pitkällä aikajänteellä. [5] Mikropalveluarkkitehtuurin edellyttämä huolellinen suunnittelu ja toteutus voivat olla ristiriidassa startup-yritysten alkuvaiheen tavoitteiden kanssa. Painopisteen ollessa idean validoimisessa tai pienimmän julkaisukelpoisen tuotteen kehityksessä, voi yritys joutua tekemään äkillisiä ratkaisuja arkkitehtuurin kustannuksella. Esimerkiksi huolettomasti kehitetyt palvelut voivat johtaa tilanteeseen, jossa ohjelmisto muodostuu toisiinsa tiukasti kytkeytyneistä palveluista. [18] Tällöin arkkitehtuuri alkaa muistuttamaan enemmän monoliittista kokonaisuutta, jossa yhdistyvät molempien arkkitehtuurien huonot puolet.

Näistä syistä monoliittinen arkkitehtuuri voidaan nähdä perustellumpana valintana varhaisen vaiheen startup-yritykselle. Nopean käyttöönoton, yksinkertaisen kehitysympäristön ja pienen tiimin hallittavuuden merkitys korostuu ympäristössä, jossa tulevaisuuden tarpeet ja liiketoiminnan suunta ovat vielä epävarmoja. Mikropalveluihin siirtymistä voidaan harkita myöhemmin, kun liiketoimintamalli on vakiintunut, tiimi on kasvanut ja järjestelmän vaatimukset ovat selkeytyneet. Tällöin olemassa oleva monoliittinen järjestelmä voidaan hajoittaa asteittain mikropalveluiksi, kun siirtymä nähdään liiketoiminnallisesti kannattavana.

4.2 Pk-yritykset

Pienet ja keskisuuret yritykset sijoittuvat organisaatiomallien vertailussa tilanteeseen, jossa sekä hyödyt että haasteet voivat olla suhteellisen tasapainossa. Luvussa 2.3 kuvattiin, että pk-yritykset ovat vakiinnuttaneet toimintansa ja resurssit voivat

olla kohtuulliset. Ominaista voi olla ketterä reagointi muutoksiin, eikä organisaation sisällä ole useita hierarkiaportaita. On kuitenkin tärkeää tunnistaa, että pk-yritykset muodostavat erittäin heterogeenisen joukon. Mikropalveluarkkitehtuurin soveltuvuus ei riipu pelkästään yrityksen koosta vaan myös ohjelmiston luonteesta, kasvunopeudesta ja teknisestä nykytilasta. Kun kehitysprosessit alkavat olla standardoituja ja tiimit eriytyvät omille vastuualueilleen, organisatoriset edellytykset alkavat vastata paremmin mikropalveluarkkitehtuurin vaatimuksia.

Luvussa 2.2 kuvattiin, miten monoliittisen arkkitehtuurin heikkoudet korostuvat järjestelmän kasvaessa. [6] Yksittäisten osien muuttaminen edellyttää koko sovelluksen kääntämistä ja julkaisemista uudelleen, jolloin muutosten vaikutusalue laajenee ja kehityssyklit hidastuvat. Mikropalveluarkkitehtuurin modulaarinen rakenne tarjoaa tähän konkreettisen ratkaisun: muutokset rajautuvat selkeämmin yksittäisiin palveluihin ja julkaisusyklit nopeutuvat. [8] Mikropalveluarkkitehtuuria voisi pitää perusteltuna ratkaisuna silloin, kun liiketoiminta on kasvanut pisteeseen, jossa monoliittisen järjestelmän kehityssyklit hidastuvat ja ylläpito aiheuttaa kasvavia kustannuksia.

Pk-yritysten näkökulmasta hyvin oleelliseksi nousevat järjestelmän tulevaisuuden kehityssuunnat. Jos ohjelmiston odotetaan pysyvän rajattuna eikä käyttäjämäärien tai toiminnallisuuden kasvu ole suurta, monoliittinen arkkitehtuuri voi täyttää tarvittavat vaatimukset. Mikropalveluarkkitehtuuri puolestaan mahdollistaa hallitumman järjestelmän teknisen skaalaamisen. [7] Kapasiteettia voidaan lisätä pieninä askelina ja kohdistaa niihin palveluihin, joissa kuormitus kasvaa, mikä voi tuoda kustannusetuja pitkällä aikavälillä. Kapasiteetin lisääminen pienemmissä askeleissa voidaan nähdä myös etuna tilanteissa, joissa resurssit ovat rajalliset, eikä suuria kertaluonteisia infrastruktuuri-investointeja haluta tehdä. Arkkitehtuuripäätökset eivät siis liity vain nykytilaan, vaan myös tulevaisuuden tavoitteisiin. Toisaalta jos järjestelmä on edelleen hyvin hallittavissa monoliittisena kokonaisuutena, eikä eriytyville

tiimeille tai riippumattomalle skaalaamiselle ole tarvetta, mikropalveluarkkitehtuurin hyödyt voivat jäädä rajallisiksi.

Liiketoiminnallisesta näkökulmasta mikropalveluarkkitehtuuri voi tuoda etuja pk-yrityksille kehitystyön organisoinnissa. Arkkitehtuurin tuoma modulaarisuus voi lyhentää muutosten läpimenoaikaa, helpottaa eri toiminnallisuuksien rinnakkaista kehittämistä ja parantaa kykyä reagoida asiakaspalautteeseen sekä markkinamuutoksiin. Skaalautuvuus tuo etuja etenkin kun tavoitteena on kasvattaa ohjelmiston käyttäjämääriä. Hyödyt kuitenkin realisoituvat vain, jos organisaatiossa on riittävästi osaamista hajautettujen järjestelmien hallinnasta ja jos tiimit kykenevät toimimaan riittävän itsenäisesti. Jos arkkitehtuurin monimutkaisuus kasvaa nopeammin kuin organisaation kyky hallita sitä, voi tämä näkyä siinä, että mikropalveluarkkitehtuurilta odotetut hyödyt jäävät saavuttamatta.

4.3 Suuryritykset

Suuryritysten kontekstissa mikropalveluarkkitehtuurin hyödyt voivat olla parhaiten saavutettavissa, mutta toisaalta haasteiden vaikutukset ovat myös laajimmat. Luvussa 2.3 todettiin, että suuryrityksissä rakenne on usein hierarkinen, prosessit ovat raskaasti standardoituja ja henkilöstö voi olla erikoistunut omille osa-alueilleen. Mikropalveluarkkitehtuurin tuoma tiimimalli tukee rakennetta, koska siinä vastuu palveluista hajautetaan omille tiimeille. Tiimimalli voi siis selkeyttää ylipäätään organisaation rakennetta ja ohjelmistokehityksen toteuttamista. Toisaalta suuryrityksissä monoliitista siirtyminen mikropalveluarkkitehtuuriin voi luoda haasteita muutosvastarinnan ja olemassa olevan rakenteen kanssa. Tiimimalli, jossa kehittäjät kantavat vastuun myös operoinnista, vaatii kulttuurimuutosta sekä merkittävää tiimien välistä luottamusta. [2] Hierarkkisissa organisaatioissa tämä voi olla haastavaa toteuttaa nopeasti.

Suuryrityksissä tyypillisesti resurssit ja henkilöstö mahdollistavat arkkitehtuurin edellyttämän infrastruktuurin rakentamisen ja ylläpidon. DevOps-käytänteiden, CI/CD-putkistojen ja monitorointiratkaisujen käyttöönotto ja ylläpito ei välttämättä ole yhtä suuri kynnys kuin startup-yrityksille tai pienemmille pk-yrityksille. Tutkimukset myös osoittavat, että skaalan kasvaessa mikropalveluarkkitehtuurin kustannusedut alkavat realisoitua. [6, 7] Näiden hyötyjen merkitys nousee erityisesti, kun ohjelmisto kasvaa käyttäjämääriltään ja toiminnallisuuksiltaan jatkuvasti isommaksi.

Luvussa 3.2 todettiin mikropalveluarkkitehtuurin teknisen monimutkaisuuden kasvavan palveluiden määrän noustessa. [8] Tämän myötä järjestelmän kompleksisuus kasvaa, joka nostaa koordinaation tarvetta organisaation sisällä. On kuitenkin tärkeää huomata, ettei koordinaatiotarve ole mikropalveluarkkitehtuurin aiheuttama ongelma, vaan se on yhtä lailla esillä monoliittisessakin arkkitehtuurissa. Monoliittisessa arkkitehtuurissa haasteita aiheuttaa yhteinen koodikanta, jossa muutokset vaikuttavat helposti toisiinsa. Mikropalveluarkkitehtuurissa koordinaation luonne muuttuu kohti rajapintojen hallinnoimista. Tämä voi suuryrityksissä olla hallittavampi muoto, sillä selkeästi määritellyt rajapinnat tekevät tiimien välisistä riippuvuuksista yksiselitteisempiä ja helpommin hallittavia.

4.4 Yhteenveto organisaatiokohtaisista eroista

Luvussa 3 esitetty kirjallisuus ja luvussa 4 analysointi esittävät, ettei mikropalveluarkkitehtuurin liiketoiminnalliset hyödyt realisoidu pelkästään arkkitehtuurivalinnalla tai tietyn kokoisella organisaatiolla. Hyötyjen toteuttaminen vaatii organisaatiolta rakenteen, prosessien ja osaamisen yhteensovittamisen arkkitehtuurin vaatimuksiin. Tämän myötä soveltuvuuden arviointi edellyttää organisaation lähtökohtien ja tavoitteiden tarkastelua ennen arkkitehtuuripäätöksen tekemistä. Taulukko

4.1 kokoaa keskeisimmät kriteerit, joiden perusteella voidaan arvioida mikropalveluarkkitehtuurin soveltuvuutta omaan kontekstiin.

Taulukko 4.1: Mikropalveluarkkitehtuurin soveltuvuutta tukevat ja sitä haastavat tekijät, tekijän oma kooste, perustuu lähteisiin [2, 4, 6, 7, 8, 18]

Tekijä	**Tukee mikropalveluarkkitehtuuria**	**Viittaa haasteisiin**
Järjestelmän koko	Järjestelmä on laaja tai sen odotetaan kasvavan merkittävästi	Järjestelmä on pieni tai rajattu, eikä kasvu ole todennäköistä
Skaalautuvuus-tarpeet	Eri toiminnot kuormittuvat eri tavoin ja vaativat itsenäistä skaalaamista	Kuormitus jakautuu tasaisesti eikä kohdennettu skaalaus ole tarpeen
Järjestelmän monimutkaisuus	Järjestelmä koostuu selkeästi eroteltavista toiminnallisuuksista, joita voidaan hallita itsenäisinä kokonaisuuksina	Järjestelmä on yksinkertainen tai toiminnallisuudet ovat tiiviisti kytköksissä toisiinsa
Tiimien rakenne ja autonomia	Tiimit kykenevät toimimaan itsenäisesti ja ottamaan vastuun palvelun koko elinkaaresta	Tiimit ovat pieniä tai vastuut ovat keskitetty, eikä autonomiselle toiminnalle ole edellytyksiä
DevOps-kypsyys	Organisaatiolla on käytössä automatisoidut julkaisu- ja monitorointikäytännöt	DevOps-käytänteet puuttuvat tai ovat alkuvaiheessa
Osaaminen	Tiimeissä on riittävästi osaamista hajautetuista järjestelmistä, pilvi-infrastruktuurista ja tietoturvasta	Osaaminen on keskitetty erillisiin tukitiimeihin tai puuttuu kokonaan
Käyttöönoton resurssit	Organisaatiolla on resursseja infrastruktuurin rakentamiseen ja palvelurajojen huolelliseen suunnitteluun	Resurssit ovat rajalliset ja ne kohdistetaan ensisijaisesti liiketoiminnan kehittämiseen
Aikajänne	Organisaatio tähtää pitkäjänteiseen kehitykseen, jossa arkkitehtuurin kustannusedut ehtivät realisoitua	Painopiste on lyhyen aikavälin tavoitteissa tai liiketoimintamalli ei ole vielä vakiintunut

Taulukossa esitetyt kriteerit kuvaavat tekijän omaa synteesiä tutkielmassa esitetyn kirjallisuuden pohjalta. Taulukkoa voi käyttää työkaluna mikropalveluarkkitehtuurin soveltuvuuden arvioimiseen. Keskeisenä havaintona on, että arkkitehtuurimallin soveltuvuutta pitäisi arvioida organisaation kokonaistilanteen pohjalta. Toisaalta yksittäinen organisaatio voi täyttää osan kriteereistä, mutta olla täyttämättä toisia, jolloin päätös edellyttää tapauskohtaista harkintaa.

5 Yhteenveto

Tutkielmassa tarkasteltiin mikropalveluarkkitehtuurin liiketoiminnallisia hyötyjä ja haasteita, jotka yhdistettiin kolmeen kirjallisuudessa esiintyvään organisaatiomalliin. Liiketoiminnalliset hyödyt ja haasteet perustuvat kirjallisuudessa esitettyihin teknisiin ominaisuuksiin, joista liiketoiminnalliset vaikutukset on johdettu.

Ensimmäinen tutkimuskysymys käsitteli mikropalveluarkkitehtuurin liiketoiminnallisia hyötyjä. Tutkitun kirjallisuuden perusteella keskeisimmiksi hyödyiksi nousi kehityksen ketteryys ja nopeammat julkaisusyklit, mitkä juontuvat palveluiden itsenäisestä kehitettävyydestä ja löyhästä kytkennästä. Ominaisuuksien myötä ohjelmiston ylläpidettävyys ja joustavuus tulevaisuuden muutoksiin korostuu. Tämä nousee erityisesti ominaiseksi piirteeksi ohjelmiston laajuuden kasvaessa. Skaalautuvuuden osalta mikropalveluarkkitehtuuri mahdollistaa resurssien kohdentamisen yksittäisiin palveluihin, mikä voi tuoda kustannusetuja. Arkkitehtuurimalli parantaa myös järjestelmän vikasietoisuutta ja mahdollistaa heterogeenisen teknologiapinon käytön. Heterogeeninen teknologiapino on erityinen etu uusien teknologioiden liittämisessä jo olemassa olevaan järjestelmään.

Toinen tutkimuskysymys käsitteli liiketoiminnallisia haasteita. Merkittäviä haasteita oli käyttöönoton alkuinvestoinnit, järjestelmän kokonaiskompleksisuus sekä osaamisvaatimukset. Mikropalveluarkkitehtuuri edellyttää esimerkiksi DevOps-käytänteiden omaksumista, huolellista palvelurajojen suunnittelua ja laajoja osaamisvaatimuksia henkilöstöltä. Haasteet voivat näkyä liiketoiminnassa resursseja sito-

vina. Kokonaiskompleksisuuden kasvu edellyttää panostuksia järjestelmän monitorointiin ja ylläpitoon. Riskinä voidaan myös nähdä, että mikropalveluarkkitehtuurin toteuttamisessa epäonnistutaan. Kirjallisuudessa ei otettu merkittävästi kantaa huonosti toteutetun mikropalveluarkkitehtuurin vaikutuksiin. Tutkimus epäonnistuneiden toteutusten vaikutuksesta olisi arvokas lisä aiheen tutkimukselle. Kirjallisuuden havaintojen perusteella voidaan kuitenkin päätellä, että epäonnistuneen toteutuksen seurauksena voisi olla useista monoliittisista sovelluksista koostuva järjestelmä, jossa yhdistyvät molempien arkkitehtuurien huonot puolet.

Kolmas tutkimuskysymys tarkasteli organisaatiomallien vaikutusta mikropalveluarkkitehtuurin hyötyihin ja haasteisiin. Pohdinta osoitti, että mikropalveluarkkitehtuurin soveltuvuus ei sitoudu pelkästään organisaation kokoon. Merkittävänä tekijänä on organisaation kypsyys toteuttaa arkkitehtuurin vaatimat tekniset ja kulttuuriset vaatimukset. Varhaisen vaiheen startup-yritykselle monoliittinen arkkitehtuuri on tyypillisesti perustellumpi valinta, sillä vähäiset resurssit kohdennetaan tyypillisesti liiketoiminnan aloittamiseen eikä arkkitehtuurin kehittämiseen. Pk-yrityksessä mikropalveluarkkitehtuurin harkinta on perusteltua erityisesti silloin, kun järjestelmä on kasvamassa ja monoliittisen arkkitehtuurin rajoitteet alkavat hidastaa kehitystyötä. Suuryrityksessä mikropalveluarkkitehtuurin potentiaali on suurin, mutta siirtyminen edellyttää sitoutumista sekä tekniseen infrastruktuuriin että organisatoriseen kulttuurimuutokseen.

Lähdeluettelo

[1] J. Carriere, R. Kazman ja I. Ozkaya, "A Cost-Benefit Framework for Making Architectural Decisions in a Business Context", teoksessa *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 2, toukokuu 2010, s. 149–157. DOI: 10.1145/1810295.1810317.

[2] S. Baškarada, V. Nguyen ja A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges", *Journal of Computer Information Systems*, vol. 60, nro 5, s. 428–436, syyskuu 2020, ISSN: 0887-4417. DOI: 10.1080/08874417.2018.1520056.

[3] P. Clements ja L. Bass, "Relating Business Goals to Architecturally Significant Requirements for Software Systems", *Software Engineering Institute, SEI*, toukokuu 2010. DOI: 10.1184/R1/6582992.v1.

[4] T. Kiblawi, M. Muhanna ja A. Qusef, "The Role of Software Architecture in Business Model Transformability", teoksessa *2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, toukokuu 2023, s. 68–73. DOI: 10.1109/JEEIT58638.2023.10185701.

[5] R. L. Nord, I. Ozkaya, P. Kruchten ja M. Gonzalez-Rojas, "In Search of a Metric for Managing Architectural Technical Debt", teoksessa *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, elokuu 2012, s. 91–100. DOI: 10.1109/WICSA-ECSA.212.17.

[6] G. Blinowski, A. Ojdowska ja A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation", *IEEE access*, vol. 10, s. 20 357–20 374, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3152803.

[7] P. Jatkiewicz ja S. Okrój, "Differences in Performance, Scalability, and Cost of Using Microservice and Monolithic Architecture", teoksessa *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, sarja SAC '23, New York, NY, USA: Association for Computing Machinery, kesäkuu 2023, s. 1038–1041, ISBN: 978-1-4503-9517-5. DOI: 10.1145/3555776.3578725.

[8] R. P. Singh, M. Thakur, S. M. A. Razavi, S. Sankhla, K. K. Singh ja I. H. Limbasiya, "Monolithic and Microservice Architecture: A Sustainable Approach", teoksessa *2025 3rd IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)*, helmikuu 2025, s. 1–6. DOI: 10.1109/ICIDeA64800.2025.10962984.

[9] U. Chouhan, V. Tiwari ja H. Kumar, "Comparing Microservices and Monolithic Applications in a DevOps Context", teoksessa *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, elokuu 2023, s. 1–7. DOI: 10.1109/ASIANCON58793.2023.10270721.

[10] B. Abdelghani, "Exploring the distinctions between Startups and small and medium-sized enterprises (SMEs): A comparative analysis", *International journal of economic perspectives*, vol. 18, nro 12, s. 2499–2511, joulukuu 2024, ISSN: 1307-1602.

[11] O.-I. Dobre, "Differences of organizational culture between small and large enterprises", *Ovidius University Annals, Economic Sciences Series*, 2016. url: https://ideas.repec.org/a/ovi/oviste/vxviy2016i1p296-301.html.

[12] R. P. Silva ja H. S. Mamede, "The Software Architecture Driving A Successful Digital Transformation within Small and Medium Enterprises", teoksessa *2022*

the 5th International Conference on Information Science and Systems, Beijing China: ACM, elokuu 2022, s. 1–6, ISBN: 978-1-4503-9683-7. DOI: 10.1145/3561877.3561906.

[13] N. Sytnik ja M. Kravchenko, "Application of knowledge management tools: Comparative analysis of small, medium, and large enterprises", *Journal of Entrepreneurship, Management and Innovation*, vol. 17, nro 4, s. 121–156, 2021, ISSN: 2299-7326. DOI: 10.7341/20211745.

[14] D. Kindström, P. Carlborg ja T. Nord, "Challenges for Growing SMEs: A Managerial Perspective", *Journal of Small Business Management*, vol. 62, nro 2, s. 700–723, maaliskuu 2024, ISSN: 0047-2778. DOI: 10.1080/00472778.2022.2082456.

[15] D. Pereira, *Apple Organizational Structure Analysis*, kesäkuu 2025. url: https://businessmodelanalyst.com/apple-organizational-structure-analysis/.

[16] A. Sablis, D. Smite ja N. Moe, "Team-external coordination in large-scale software development projects", *Journal of Software: Evolution and Process*, vol. 33, nro 3, e2297, 2021, ISSN: 2047-7481. DOI: 10.1002/smr.2297.

[17] K. H. Rolland ja K. Lyytinen, "Managing Tensions between Architectural Debt and Digital Innovation: The Case of a Financial Organization", teoksessa *Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS-54)*, tammikuu 2021, s. 6722–6731.

[18] P. Vitharana ja S. A. Daya, "Adopting and Sustaining Microservice-Based Software Development", *Communications of the ACM*, vol. 67, nro 7, s. 34–41, heinäkuu 2024, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3651620.