Ohjelmistotekniikka

Tietotekniikan laitos, Teknillinen tiedekunta, Turun yliopisto

Diplomityö

Kesäkuu 2022

Santeri Halkivaha

PALVELINARKKITEHTUURIN LAATUMALLI JA SEN SOVELTAMINEN PILVIPALVELUISSA

DIPLOMITYÖ

TIETOTEKNIIKAN LAITOS, TEKNILLINEN TIEDEKUNTA, TURUN YLIOPISTO

Ohjelmistotekniikka

Kesäkuu 2022 | 84 sivua

Ohjaajat: Sampsa Rauti, Tuomas Mäkilä

Santeri Halkivaha

PALVELINARKKITEHTUURIN LAATUMALLI JA SEN SOVELTAMINEN PILVIPALVELUISSA

Tässä diplomityössä perehdytään ohjelmistoarkkitehtuurin laatumalleihin sekä laatuattribuutteihin ja löydetään modernin palvelinarkkitehtuurin olennaiset laatuattribuutit sekä kehitetään palvelinarkkitehtuurin laatumalli. Työssä pohditaan konesalien ja pilvipalveluiden välistä suhdetta ja niiden hyviä ja huonoja puolia, tutkitaan pilvipalveluita yleisellä tasolla ja perehdytään niiden tarjoamiin mahdollisuuksiin ja haasteisiin sekä pohditaan palvelinarkkitehtuurimalleja. Työssä tutkitaan pilvipalveluita ja niiden tarjoamia ominaisuuksia peilaten niitä palvelinarkkitehtuurin laatuattribuutteja ja laatumallia vasten.

Tutkimusmenetelmänä työssä käytetään kirjallisuuskatsausta. Työn tuloksena kehitettiin palvelinarkkitehtuurin laatumalli, joka sisältää seuraavat olennaiset laatuattribuuttikategoriat: tietoturva, saatavuus, vikasietoisuus, skaalautuvuus ja ylläpidettävyys. Laatumallin mukaisen suunnittelun tukea tutkitaan kolmen suuren pilvipalvelutarjoajan (Microsoft Azure, Amazon AWS ja Google Cloud) kautta tullen siihen tulokseen, että suuret pilvipalvelut tukevat hyvin laatumallin mukaista suunnittelua.

Jatkotoimenpiteinä suositellaan palvelinarkkitehtuurin laatumallin validointia käytännössä ja syvällisempää tutkimusta aiheesta kuin myös pilvipalveluiden tarpeellisen kehityssuunnan tutkimista, jotta palvelinarkkitehtuurin laatuattribuuttien kaltaista rakennetta pystyttäisiin tukemaan entistä paremmin.

ASIASANAT:

Pilvipalvelut, Palvelimet, Palvelinarkkitehtuuri, Ohjelmistoarkkitehtuuri, Laatumalli, Laatuattribuutit

MASTER´S THESIS | ABSTRACT

DEPARTMENT OF COMPUTING, FACULTY OF TECHNOLOGY, UNIVERSITY OF TURKU

Software engineering

June 2022 | 84 pages

Thesis supervisors: Sampsa Rauti, Tuomas Mäkilä

Santeri Halkivaha

SERVER ARCHITECTURE QUALITY MODEL AND ITS APPLICATION IN CLOUD COMPUTING

This thesis examines the quality models and quality attributes of software architecture and finds the essential quality attributes of modern server architecture and develops the quality model of server architecture. Thesis examines the relationship between traditional data centers and cloud computing services and their pros and cons, examines cloud computing in general and gets acquainted with the opportunities and challenges they offer, and examines server architecture models. The thesis examines cloud computing services and the features they offer, mirroring them against the quality attributes and quality model of the server architecture.

The research method used in the thesis is a literature review. As a result of the thesis, server architecture quality model was developed, which includes the following essential quality attribute categories: data security, availability, fault tolerance, scalability and maintainability. Quality model design support is being explored through three major cloud computing service providers (Microsoft Azure, Amazon AWS and Google Cloud), with the conclusion that large cloud computing services support quality model design well.

As a follow-up, it is proposed that the server architecture quality model would be validated in practice and that more in-depth research would be carried out on the subject, as well as the necessary development of cloud computing services to better support a structure such as server architecture quality attributes.

KEYWORDS:

Cloud computing, Servers, Server architecture, Software architecture, Quality model, Quality attributes

SISÄLTÖ

KUVAT

TAULUKOT

1 JOHDANTO

Modernit ohjelmistot ja digitaaliset palvelut sekä alustat vaativat toimiakseen modernit ja luotettavat palvelinratkaisut. Yhteiskuntamme toimii yhä enenevissä määrin tietojärjestelmien varassa ja niitä hyödyksi käyttäen. Tämä luo tarpeen luotettavista, ympäri vuorokauden saavutettavista palveluista. Palveluiden käyttäjät eivät halua käyttökatkoksia. Alalla kilpailu on kovaa ja vaihtoehtoja kunkin tarpeisiin on runsaasti. Tällöin palveluiden saavutettavuudella on suuri merkitys, varsinkin jos kyseessä on liiketoiminnan kannalta kriittisistä järjestelmistä.

Palvelinratkaisut ovat olleet viime vuosina murroksessa. Pilvipalveluiden yleistyessä perinteisemmät keskitetyt konesalit ja fyysiset palvelinlaitteistot yritysten omissa tiloissa ovat saaneet varteenotettavan ja kustannustehokkaan haastajan. Pilvipalvelut ovat mullistaneet tapamme suunnitella ja kehittää ohjelmistoja ja digitaalisia palveluita. Tämä herättää kysymyksen siitä, onko perinteisillä konesaleilla enää mahdollisuuksia pärjätä taistossa pilvipalveluita vastaan.

Myös palvelinarkkitehtuuri on kokenut muutoksia pilvipalveluiden yleistyessä. Digitaalisten palveluiden saavutettavuus on yksi nyky-yhteiskuntamme kulmakivistä. Monet yhteiskuntamme kriittisistä tietojärjestelmistä eivät siedä käyttökatkoksia. Trendinä on kriittistenkin palveluiden vieminen pilveen ja muuttaminen digitaalisiksi, helposti saavutettaviksi ja skaalautuviksi tietojärjestelmiksi, joiden tietoturvan edellytetään olevan korkealla tasolla. Tämä edellyttää paljon sekä itse palvelinratkaisulta mutta myös palvelinarkkitehtuurilta. Tarve modernien palvelinratkaisujen vaatimusten selvittämiselle on suuri.

Tämän hetken trendinä on luopua vanhoista konesalipalveluista ja siirtää palvelimet pilveen. Uusia sovelluksia ja ohjelmistoja suunniteltaessa ja kehitettäessä tämä ei aiheuta suuria haasteita. On kuitenkin olemassa paljon useita vuosia, jopa vuosikymmeniä, vanhoja tietojärjestelmiä, joita nykytrendin mukaisesti siirretään käyttämään pilvipalveluita. Tällaisia perinteisiä järjestelmiä ei useinkaan ole suunniteltu arkkitehtuuriltaan mukautuviksi, mikä aiheuttaa ylimääräisiä kuluja ja jopa estää käyttämästä pilvipalveluita. Pilvipalveluihin siirtymisen haasteiden selvittäminen on osa onnistunutta projektisuunnittelua ja säästää sekä aikaa että rahaa.

Tämän työn tavoitteena on tutustua ohjelmistoarkkitehtuurin laatumalleihin sekä laatuattribuutteihin ja löytää modernin palvelinarkkitehtuurin olennaiset laatuattribuutit sekä kehittää palvelinarkkitehtuurin laatumalli. Tavoitteena on pohtia konesalien ja pilvipalveluiden välistä suhdetta ja niiden hyviä ja huonoja puolia, tutkia pilvipalveluita yleisellä tasolla ja perehtyä niiden tarjoamiin mahdollisuuksiin ja haasteisiin sekä pohtia palvelinarkkitehtuurimalleja. Tarkoituksena on tutkia pilvipalveluita ja niiden tarjoamia ominaisuuksia peilaten niitä palvelinarkkitehtuurin laatuattribuutteja ja laatumallia vasten.

Työn tutkimuskysymykset ovat

1. Mitkä ovat olennaiset palvelinarkkitehtuurin laatuattribuutit?
2. Miten pilvipalvelut tukevat palvelinarkkitehtuurin laatuattribuuttien mukaista suunnittelua?

Toistaiseksi tutkimuskysymysten kaltaisia aiheita on tutkittu varsin vähän. Tämä työ pyrkii osaltaan tutkimaan aihetta. Tutkimusmenetelmänä käytetään kirjallisuuskatsausta. Luvussa kaksi käydään läpi pilvipalveluiden suhdetta konesaleihin sekä pohditaan pilvipalveluiden hyviä ja huonoja puolia. Lisäksi perehdytään palvelinarkkitehtuurimalleihin. Luvussa kolme käydään läpi ohjelmistoarkkitehtuurin laatumalleja ja niiden pohjalta rakentuvaa palvelinarkkitehtuurin laatumallia. Luvussa neljä tutkitaan, miten pilvipalvelut vastaavat ja tukevat palvelinarkkitehtuurin laatumallin kaltaista suunnittelua.

2 PILVIPALVELUT

Tässä luvussa perehdytään tarkemmin pilvipalveluihin ja niiden hyviin sekä huonoihin puoliin. Lisäksi tutustutaan yleisiin pilviarkkitehtuurimalleihin. Pilvipalveluiden hyötyjen ja haasteiden osalta näkökulma on osittain liiketoiminnallinen, teknisempää näkökulmaa on mukana alaluvussa 2.2 ja luvussa 4.

Pilvipalvelut ovat tietokonejärjestelmäresurssien, erityisesti tietojen tallennuksen ja laskentatehon, saatavuutta tarpeen vaatiessa ilman käyttäjän suoraa aktiivista hallintaa [9]. Sen sijaan, että ostetaan, omistetaan ja ylläpidetään fyysisiä konesaleja ja palvelimia, voidaan käyttää teknologiapalveluja, kuten laskentatehoa, tallennustilaa ja tietokantoja, tarvittaessa pilvipalveluntarjoajalta [10].

Kaikentyyppiset ja -kokoiset organisaatiot käyttävät pilvipalveluita monenlaisiin eri tarpeisiin, kuten tietojen varmuuskopiointiin ja palauttamiseen, sähköpostiin, virtualisointiin, ohjelmistojen kehittämiseen ja testaamiseen, big data -analytiikkaan ja verkkosovelluksiin. [10].

2.1 Konesalit ja pilvipalvelut

Konesali on palvelin tai palvelinkokoelma, joka ostetaan ja säilytetään paikan päällä palvelinkeskuksessa palvelemaan tallennustarpeita. Tämä voidaan nähdä haittana, erikseen on ostettava palvelinlaitteisto ja verkkolaitteisto. Kuten mikä tahansa tekniikka, se vanhenee ja on vaihdettava aika ajoin. Kaikkien näiden laitteiden hankintakustannusten lisäksi on myös palkattava henkilöstö, joka ylläpitää palvelimia. Näiden työntekijöiden on oltava hyvin koulutettuja ja pidettävä taitonsa ajan tasalla. [11].

Suurin ero julkisen pilven ja konesalin välillä on tietojen tallennuspaikka. Konesalissa tiedot tallennetaan useimmiten organisaation tiloihin. Jotkin palvelinkeskukset voivat sijaita paikoissa, jotka eivät ole organisaation omistamia. Pilvi on täysin paikasta riippumaton ja tiedot ovat käytettävissä mistä tahansa internetin kautta. [11].

2.1.1 Pilvipalveluiden hyödyt

Kustannussäästö (cost savings): Kustannussäästö on yksi pilvipalvelujen suurimmista eduista. Pilvipalvelut auttavat yrityksiä säästämään huomattavia pääomakustannuksia, koska ne eivät vaadi fyysisiä laitteistoinvestointeja. Yritykset eivät myöskään tarvitse koulutettua henkilöstöä laitteiston ylläpitoon ja yrityksen sähkönkulutus voi pienentyä. Laitteiden ostamisesta ja hallinnasta vastaa pilvipalveluntarjoaja. [12], [13], [14], [15], [16], [17], [18], [19].

Tietoturva (security): Pilvipalveluiden ylläpitäjien kokopäiväinen tehtävä on valvoa tietoturvaa, joka on huomattavasti tehokkaampi kuin perinteinen sisäinen järjestelmä, jossa organisaation on jaettava resurssinsa lukemattomien IT-huolenaiheiden kesken, ja turvallisuus on vain yksi niistä. Ja vaikka useimmat yritykset eivät halua avoimesti käsitellä sisäisten tietovarkauksien mahdollisuutta, totuus on, että hämmästyttävän suuri prosenttiosuus tietovarkauksista tapahtuu sisäisesti ja työntekijät syyllistyvät niihin. Tässä tapauksessa voi olla paljon turvallisempaa pitää arkaluonteiset tiedot muualla. [12], [13], [15], [16], [18].

RapidScalen [20] mukaan 94 % yrityksistä koki tietoturvan parantuneen pilveen siirtymisen jälkeen ja 91 % sanoi, että pilvi helpottaa lakiperusteisten vaatimusten täyttämistä. Avain tähän vahvistettuun turvallisuuteen on verkkojen kautta lähetettävien ja tietokantoihin tallennettujen tietojen salaus. Lisäsuojatoimenpiteenä useimmille pilvipohjaisille palveluille voidaan asettaa ylimääräisiä suojausasetuksia. [12].

Joustavuus (flexibility): Yrityksillä on vain rajallinen määrä resursseja jaettavaksi kaikille vastuualueille. Jos IT-ratkaisut pakottavat kiinnittämään liikaa huomiota tietokone ja tallennusväline ongelmiin, yritys ei voi keskittyä liiketoimintatavoitteiden saavuttamiseen ja asiakkaiden tyydyttämiseen. Luotettaessa ulkopuoliseen organisaatioon, joka huolehtii kaikesta IT-isännöinnistä ja infrastruktuurista, yrityksellä on enemmän aikaa oman liiketoiminnan osa-alueisiin, jotka vaikuttavat suoraan tulokseen. [12], [13], [16], [17], [18], [19].

Pilvi tarjoaa yrityksille enemmän joustavuutta verrattuna paikallisen palvelimen isännöintiin. Jos tarvitaan ylimääräistä kaistanleveyttä, pilvipohjainen palvelu voi vastata tähän kysyntään välittömästi skaalautumalla sen sijaan, että suorittaisiin monimutkainen (ja kallis) IT-infrastruktuurin päivitys. Tämä parantunut vapaus ja joustavuus voivat vaikuttaa merkittävästi organisaation yleiseen tehokkuuteen. 65 % enemmistö

InformationWeekin [21] kyselyyn vastanneista sanoi, että ”kyky vastata nopeasti liiketoiminnan vaatimuksiin” oli yksi tärkeimmistä syistä, miksi yrityksen pitäisi siirtyä pilviympäristöön. [12], [13].

Liikkuvuus (mobility): Pilvipalvelut mahdollistavat pääsyn yritystietoihin älypuhelimien ja liikkuvien laitteiden kautta. Maailmanlaajuisesti käytössä on yli 6,3 miljardia älypuhelinta vuonna 2021 ja älypuhelimien käyttäjien ennustetaan kasvavan 7,5 miljardiin vuoteen 2026 mennessä [22]. Työntekijät, joilla on kiireinen aikataulu tai jotka asuvat kaukana yrityksen toimistosta, voivat käyttää pilvipalveluiden mahdollistamaa liikkuvuutta pysyäkseen ajan tasalla asiakkaiden ja työtovereiden kanssa. Pilven kautta voidaan tarjota kätevästi saatavilla olevia tietoja matkustavalle myyntihenkilöstölle, freelance-työntekijöille tai etätyöntekijöille työ- ja yksityiselämän tasapainottamiseksi. Organisaatiot, joissa työntekijöiden tyytyväisyys on korkealla tasolla, laajentavat pilvipalveluiden käyttöä 24 % todennäköisemmin. [12], [13], [14], [15], [16], [17].

Tieto (insight): Siirryttäessä yhä syvemmälle digitaaliseen aikakauteen, tulee yhä selvemmäksi, että vanha sanonta ”tieto on valtaa” on saanut nykyaikaisemman ja tarkemman muodon: ”data on rahaa”. Asiakkaiden tapahtumia ja liiketoimintaprosessia ympäröivien miljoonien datatiedostojen joukossa on korvaamatonta, käytännöllistä tietoa, joka odottaa löytämistään. Datan läpikäynti tarvittavien tiedostojen löytämiseksi voi olla erittäin vaikeaa, ellei käytössä ole oikeaa pilvipalveluratkaisua. [12], [15], [19].

Monet pilvipohjaiset tallennusratkaisut tarjoavat integroidun pilvianalytiikan datan läpikäyntiin lintuperspektiivistä. Kun tiedot on tallennettu pilveen, voidaan helposti ottaa käyttöön seurantamekanismeja ja luoda räätälöityjä raportteja tietojen analysoimiseksi organisaatio tasolla. Näiden tietojen avulla voidaan lisätä tehokkuutta ja rakentaa toimintasuunnitelmia organisaation tavoitteiden saavuttamiseksi. Esimerkiksi juomayhtiö Sunny Delight [23] pystyi kasvattamaan voittojaan noin 2 miljoonalla dollarilla vuodessa ja leikkaamaan 195 000 dollaria henkilöstökustannuksia pilvipohjaisten liiketoimintatietojen avulla. [12], [19].

Yhteistyö (collaboration): Jos yrityksessä on kaksi tai useampi työntekijä, yrityksen pitäisi tehdä yhteistyöstä ensisijainen tavoite. Loppujen lopuksi ei ole paljon järkeä luoda tiimiä, jos se ei pysty toimimaan tiimin tavoin. Pilvipalvelut tekevät yhteistyöstä yksinkertaisen prosessin. Tiimin jäsenet voivat tarkastella ja jakaa tietoja helposti ja turvallisesti pilvipohjaisella alustalla. Jotkut pilvipohjaiset palvelut tarjoavat jopa sosiaalisia tiloja työntekijöiden yhdistämiseksi organisaatiossa, mikä lisää kiinnostusta ja sitoutumista.

Yhteistyö voi olla mahdollista ilman pilvipalveluratkaisua, mutta se ei koskaan ole yhtä helppoa eikä tehokasta. [12], [13], [14], [15], [16], [17].

Laaduntarkkailu (quality control): Huono laatu ja epäjohdonmukainen raportointi heikentävät yrityksen menestysmahdollisuuksia. Pilvipohjaisessa järjestelmässä kaikki tiedostot tallennetaan yhteen paikkaan ja yhteen muotoon. Kun kaikki käyttävät samoja tietoja, voidaan säilyttää tietojen johdonmukaisuus, välttää inhimilliset erehdykset ja pitää kirjaa kaikista muutoksista tai päivityksistä. Sitä vastoin tietojen hallinta eri soluissa voi johtaa siihen, että työntekijät tallentavat vahingossa tiedostojen eri versioita, mikä johtaa sekaannukseen ja tietojen virheellisyyteen. Muutenkin tarjotun palvelun laaduntarkkailu ja asioiden mittaaminen ovat keskiössä. [12], [15], [17].

Palautettavuus (disaster recovery): Yksi yrityksen menestymiseen vaikuttavista tekijöistä on hallinta. Valitettavasti riippumatta siitä, kuinka organisaation omat prosessit ovat hallinnassa, aina on asioita, jotka eivät ole täysin organisaation hallinnassa, ja nykypäivän markkinoilla pienelläkin tuottamattomalla häiriöajalla voi olla valtava kielteinen vaikutus. Palveluiden häiriöajat johtavat tuottavuuden, tulojen ja brändimaineen menetykseen. [12], [13], [14], [15], [16].

Vaikka yritys ei ehkä voi mitenkään estää tai edes ennakoida katastrofeja, jotka voivat vahingoittaa organisaatiota, voidaan jotain tehdä palauttamisen nopeuttamiseksi. Pilvipalvelut tarjoavat nopean tietojen palauttamisen kaikenlaisille hätätilanteille luonnonkatastrofeista sähkökatkoihin. Vaikka 20 % pilvipalvelukäyttäjistä onnistuu palautumaan katastrofista neljässä tunnissa tai nopeammin, vain 9 % ei-pilvikäyttäjistä pystyy samaan [20]. Kyselyssä 43 % IT-johtajista sanoi aikovansa investoida pilvipohjaisiin katastrofipalautusratkaisuihin tai parantaa niitä [24]. Pilvipalveluntarjoajien SLA-sopimukset myös takaavat tietyn palvelun laadun, saatavuuden. [12].

Vahingontorjuntatoimenpiteet (loss prevention): Jos organisaatio ei investoi pilvipalveluratkaisuun, kaikki arvokkaat tiedot ovat erottamattomasti sidoksissa toimistokoneisiin, joissa ne sijaitsevat. Tämä ei ehkä vaikuta ongelmalta, mutta todellisuus on, että jos paikallinen laitteisto kokee ongelmia, tiedot saatetaan menettää pysyvästi. Tämä on yleinen ongelma, tietokoneissa voi olla toimintahäiriöitä monista eri syistä virusinfektioista ikään liittyvään laitteiston heikkenemiseen ja yksinkertaisiin käyttäjävirheisiin. Parhaista aikomuksista huolimatta tietokoneet voidaan sijoittaa väärään paikkaan tai varastaa. Jos yritys ei ole pilvessä, vaarassa on menettää kaikki paikallisesti tallennetut tiedot. Pilvipohjaisella palvelimella kaikki pilveen ladatut tiedot

pysyvät turvassa ja helposti saatavilla miltä tahansa tietokoneelta, jolla on internet-yhteys, vaikka säännöllisesti käytetty tietokone ei toimisi. [12], [13], [14], [15], [16].

Automaattiset ohjelmistopäivitykset (automatic software updates): Pilvipohjaiset sovellukset päivittävät itsensä automaattisesti sen sijaan, että pakottaisivat IT-osaston suorittamaan manuaalisen organisaation laajuisen päivityksen. Tämä säästää arvokasta IT-henkilöstön aikaa ja rahaa ulkoiseen IT-konsultointiin. Jatkuva iterointi ja toimitus perustuvat siihen, että uusia ohjelmistoversioita voidaan helposti testata ja ottaa käyttöön pilviympäristössä, mikä mahdollistaa nopeamman tuoteinnovaation ja vapauttaa yhä enemmän ominaisuuksia loppukäyttäjille kuukausittain, viikoittain ja joissakin tapauksissa jopa päivittäin. Pilviympäristöt integroituvat myös yleisiin DevOps-työkaluihin ja lokijärjestelmiin, mikä helpottaa tuotannon ongelmien seurantaa ja havaitsemista. [12], [13], [14], [15], [17].

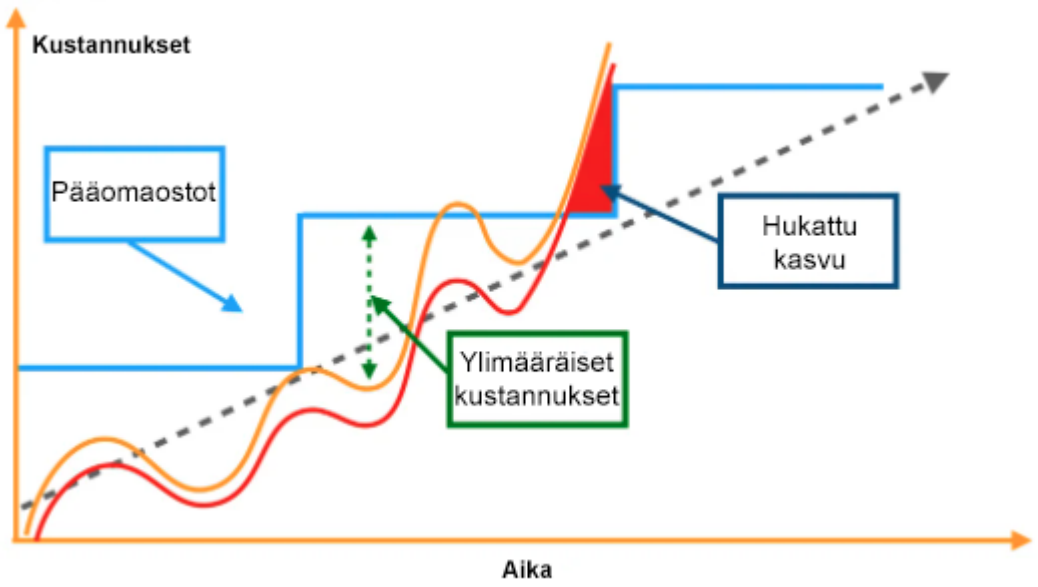
Kilpailuetu (competitive edge): Vaikka pilvipalvelujen suosio kasvaa, on silti niitä, jotka haluavat pitää kaiken paikallisena. Tämä asettaa heidät selkeästi epäedulliseen asemaan kilpaillessaan niiden kanssa, joilla on pilven edut käden ulottuvilla. Jos yritys ottaa pilvipohjaisen ratkaisun käyttöön ennen kilpailijoita, ovat nämä yritykset pidemmällä oppimiskäyrässä, kun kilpailijat saavuttavat heidät. Verizonin [25] tutkimus osoitti, että 77% yrityksistä kokee pilvitekniikan antavan heille kilpailuetua ja 16% uskoo, että tämä etu on merkittävä. [12], [13], [14], [15].

Ekologisuus (sustainability): Kun otetaan huomioon ympäristön nykytila, ei enää riitä, että organisaatiot asettavat kierrätysastian taukotilaan ja väittävät tekevänsä osansa auttaakseen planeettaa. Todellinen kestävyys edellyttää ratkaisuja, jotka vastaavat epätaloudellisen liiketoiminnan vähentämiseen kaikilla tasoilla. Tiedon säilyttäminen ja laskenta pilvessä on ympäristöystävällisempää ja vähentää hiilijalanjälkeä. [12].

Pilvi-infrastruktuurit ovat ympäristöystävällisiä ratkaisuja. Virtuaalipalvelujen käyttö fyysisten tuotteiden ja laitteistojen sijasta, paperihukan vähentämistä, energiatehokkuuden parantamista ja työmatkaliikenteeseen liittyvien päästöjen vähentäminen. Pike Researchin [26] raportin mukaan datakeskusten energiankulutus laskee 31 % vuodesta 2010 vuoteen 2020 pilvipalvelujen ja muiden virtuaalisten datavaihtoehtojen käyttöönoton vuoksi. [12].

Kuva 1 havainnollistaa hyvin pilvipalveluiden tuomia etuja perinteisiin konesaleihin nähden. Oletetaan, että yrityksellä on hyvä käsitys IT-infrastruktuurin ennakoidusta kysynnästä, jota kuvaa harmaa katkoviiva. Tyypillisessä konesaliympäristössä voidaan

suunnitella ja vastata tähän kysyntään hankkimalla määräajoin laitteita ja palveluita, ennen kuin niitä todella tarvitaan, suurilla pääomaostoilla. Tämä on usein pitkä priorisointi-, budjetointi- ja hyväksymisprosessi, joka näkyy kuvassa sinisenä. Yrityksessä luotetaan siihen, että kysyntä on osattu ennustaa oikein. Potentiaalisen kysynnän vaihtelu näkyy kuvassa punaisena viivana. Tämän vaihtelun vaikutus on se, että käytössä on liikaa resursseja ja osa resursseista menee hukkaan. Hukkaan menevät resurssit on esitetty vihreänä ylimääräisinä kustannuksina. [27].



Kuva 1. Pilvipalvelun hyödyt suhteessa konesaliin [27].

Mahdollisesti yritys on varannut liian vähän resursseja ja jättänyt hyödyntämättä mahdollisen kasvun, joka näkyy kuvassa punaisessa varjostetussa ruudussa. Pilvipalvelut auttavat välttämään näitä sudenkuoppia antamalla mahdollisuuden ostaa joustavasti vain sen määrän resursseja mitä tarvitaan (oranssi viiva) ja skaalata resursseja vain tarvittaessa. Pilvipalvelut vastaavat kysyntään tarjontaa lisäämällä ja yritys säästää tällä tavalla kustannuksissa. [27].

2.1.2 Pilvipalveluiden haasteet

Lukuisista hyödyistään huolimatta pilvipalveluilla on myös haasteita ja ongelmakohtia suhteessa perinteisempiin konesaliratkaisuihin. Monet haasteista ovat samalla myös pilvipalveluiden etuja – kolikolla on kaksi puolta. Usein se, onko jokin ominaisuus

hyödyllinen vai haitallinen, riippuu ratkaisusta ja siitä, mitä osa-alueita halutaan painottaa.

Suorituskyky (performance): Pilviympäristössä sovellukset toimivat palvelimilla, jotka tarjoavat samanaikaisesti resursseja myös muille käyttäjille. Kaikki resursseja runsaasti vaativat prosessit voivat vaikuttaa jaetun palvelimen suorituskykyyn ja näin ollen heikentää muiden asiakkaiden palvelun laatua. [14].

Tekniset haasteet (technical issues): Pilvitekniikka on aina altis katkoksille ja muille teknisille ongelmille. Jopa parhaat pilvipalveluyritykset voivat kohdata tämäntyyppisiä ongelmia huolimatta korkeista ylläpitostandardeista. Pilvipalveluntarjoajat määrittelevät SLA-sopimuksissaan mahdolliset sanktiot ja korvaukset käyttökatkoksien johdosta. [14].

Tietoturva (security): Vaikka tietoturva on yksi pilvipalveluiden merkittävimmistä eduista, voidaan se nähdä myös niiden haittapuolena. Ennen pilvitekniikan käyttöönottoa tulee olla tietoinen siitä, että kaikki yrityksen arkaluontoiset tiedot jaetaan kolmannen osapuolen, pilvipalvelujen tarjoajan, kanssa. Näin ollen tiedot ovat mahdollisten tietoturvahyökkäysten kohteena. [14], [16], [28], [31].

Vaikka pilvipalveluntarjoajat noudattavat parhaita tietoturvastandardeja ja alan sertifikaatteja, tietojen ja tärkeiden tiedostojen tallentamisessa ulkoiselle palveluntarjoajalle on aina riski. Kaikissa tietoja koskevissa keskusteluissa on käsiteltävä tietoturvaa ja yksityisyyttä, erityisesti arkaluonteisten tietojen hallinnassa. Code Spacen AWS hakkeroitiin, mikä johti tietojen poistamiseen ja yrityksen konkurssiin [30]. Heidän riippuvuutensa pilvipohjaisesta infrastruktuurista tarkoitti kaiken ulkoistamisen riskien ottamista. [28], [31].

Kaikkien pilvipalveluntarjoajien odotetaan hallitsevan ja suojaavan käytön taustalla olevaa laitteistoinfrastruktuuria. Käyttäjien velvollisuuksiin kuuluu kuitenkin käyttäjien käyttöoikeuksien hallinta ja kaikki riskiskenaariot on punnittava huolellisesti. [28].

Pilvipalveluissa jokainen komponentti on verkossa, mikä paljastaa mahdolliset haavoittuvuudet. Jopa parhaat toteutukset kärsivät ajoittain vakavista hyökkäyksistä ja tietoturvaloukkauksista. Koska pilvipalvelut on rakennettu julkiseksi palveluksi, niitä on helppo käyttää ilman sen suurempaa asiantuntemusta ja osaamista. Loppujen lopuksi kukaan pilvipalveluntarjoajalta ei tarkista osaamista ennen tilin myöntämistä: aloittamiseen tarvitaan yleensä vain luottokortti. [28], [31].

Palvelukatkot (downtime): Myös palvelukatkoja tulisi harkita pilvipalvelujen kanssa työskennellessä. Tämä johtuu siitä, että pilvipalveluntarjoaja saattaa kärsiä sähkökatkoista, heikoista internet-yhteyksistä, palvelun ylläpidosta jne. Nämä vaikuttavat suoraan palvelun saatavuuteen. Amazon Web Services -palvelun käyttökatko vuonna 2017 maksoi julkisesti noteeratuille yrityksille jopa 150 miljoonaa dollaria [29]. Valitettavasti mikään organisaatio ei ole immuuni, varsinkin kun kriittisillä liiketoimintaprosesseilla ei ole varaa keskeytyä. [14], [28], [31].

Internet-yhteys (internet connectivity): Hyvä internet-yhteys on välttämätöntä pilvipalveluissa. Pilvipalveluita ei voi käyttää ilman internet-yhteyttä, ei ole muuta tapaa päästä käsiksi tietoihin pilvessä. [14], [16], [31].

Siirtonopeus (bandwidth): Monet pilvipalvelujen tarjoajat rajoittavat käyttäjiensä siirtonopeutta. Joten jos organisaatio tarvitsee suurta siirtonopeutta, lisämaksut voivat tulla kalliiksi. [14], [31].

Tuki (support): Pilvipalveluyritykset eivät välttämättä pysty tarjoamaan asianmukaista käyttötukea asiakkaille. Pilvipalvelut tarjoavat käyttäjilleen UKK-osioita tai online apua, mikä voi olla työlästä ja vaikeaselkoista ei-teknisille henkilöille. Tukea ei välttämättä ole tarpeeksi nopeasti saatavilla. [14], [31].

Toimittajaloukku (vendor lock-in): Toimittajaloukku on yksi pilvipalvelujen suurimmista haitoista. Organisaatiot voivat kohdata ongelmia siirtäessään palveluitaan pilvipalvelusta toiseen. Koska eri pilvipalvelut tarjoavat erilaisia alustoja ja rajapintoja, se voi vaikeuttaa siirtymistä pilvestä toiseen. Helppo siirtyminen pilvipalvelujen välillä on osa-alue, joka ei ole vielä täysin kehittynyt, ja organisaatioiden voi olla vaikeaa siirtää palvelujaan toimittajalta toiselle. Toimittajaympäristöjen väliset erot voivat aiheuttaa vaikeuksia siirtyä pilvialustalta toiselle, mikä voi merkitä lisäkustannuksia. Siirron aikana tehdyt kompromissit voivat myös altistaa tietoturva- ja tietosuojahaavoittuvuuksille. [16], [28].

Kontrolli (control): Pilvi-infrastruktuuri on täysin palveluntarjoajan omistuksessa, hallinnassa ja valvonnassa, joten pilvikäyttäjät eivät pysty hallitsemaan palvelujen toimintaa ja suorittamista pilvi-infrastruktuurissa. Pilvipalveluntarjoajan loppukäyttäjän lisenssisopimus ja hallintakäytännöt saattavat asettaa rajoituksia sille, mitä asiakkaat voivat tehdä palvelullaan. Asiakkaat voivat hallita sovelluksiaan, tietojaan ja palveluitaan, mutta heillä ei ehkä ole yhtä suurta hallintaa taustainfrastruktuurista. [16], [28].

Kulut (cost concerns): Pilviratkaisujen käyttöönottoa pienessä mittakaavassa ja lyhyen aikavälin hankkeissa voidaan pitää kalliina. Merkittävin pilvilaskennan hyöty on kuitenkin IT-kustannussäästöt. Etukäteen maksettavat pilvipalvelut voivat tarjota enemmän joustavuutta ja alentaa laitteistokustannuksia, mutta kokonaishinta voi olla korkeampi kuin on osattu arvioida. Ennen kuin ollaan varmoja siitä, mikä ratkaisu toimii parhaiten, on hyvä kokeilla erilaisia palveluita. Palveluntarjoajat tarjoavat myös kustannuslaskureita. [28].

Fyysinen turvallisuus: Kun yrityksellä ei ole fyysistä pääsyä palvelinkeskuksiin, joudutaan luottamaan pilvipalveluntarjoajaan siinä, että palvelinten fyysisestä turvallisuudesta on huolehdittu asianmukaisesti.

Kaikista pilvipalvelujen hyödyistä ja haitoista huolimatta kiistatonta on se tosiasia, että pilvipalvelut ovat nopeimmin kasvava osa verkkopohjaista tietojenkäsittelyä. Ne tarjoavat suuren edun kaikenkokoisille asiakkaille: yksilöille, kehittäjille, yrityksille ja kaikenlaisille organisaatioille. [14].

2.2 Palvelinarkkitehtuurimallit

Pilviarkkitehtuuri määrittelee teknologiakomponentit, jotka yhdistetään muodostamaan pilvi, jossa resurssit yhdistetään virtualisointitekniikan avulla ja jaetaan verkon kautta. Pilviarkkitehtuurin komponentteja ovat:

- Käyttöliittymä/päätelaite (laite, jota käytetään pilven käyttämiseen)
- Yksi tai useampi taustajärjestelmä (palvelimet ja tallennustila)
- Pilvipohjainen toimitusmenetelmä
- Verkko pilviasiakkaiden, palvelimien ja tallennustilan yhdistämiseen

Yhdessä nämä tekniikat luovat pilvipalveluarkkitehtuurin, jolla sovelluksia voidaan käyttää ja tarjoavat loppukäyttäjille mahdollisuuden hyödyntää pilviresurssien tehoa. Pilvipalveluarkkitehtuuri määrittää sen, miten kaikki pilven rakentamiseen tarvittavat komponentit ja ominaisuudet yhdistetään, jotta voidaan tarjota alusta, jolla sovelluksia voidaan käyttää. Pilvi-infrastruktuuri sisältää kaikki komponentit/materiaalit ja pilvipalveluarkkitehtuuri on suunnitelma näiden käytölle. [32], [33].

Pilviarkkitehtuurissa on kolme suurta mallia, jotka ohjaavat organisaatioita pilveen. Jokaisella näistä on omat etunsa ja tärkeimmät ominaisuutensa.

- Software as a Service (SaaS): SaaS-arkkitehtuurin tarjoajat toimittavat ja ylläpitävät sovelluksia ja ohjelmistoja organisaatioille internetin välityksellä, jolloin loppukäyttäjien ei tarvitse asentaa ohjelmistoa paikallisesti. SaaS-sovelluksia käytetään tyypillisesti web-käyttöliittymän kautta, joka on saatavana useissa eri laitteissa ja käyttöjärjestelmissä. [33], [34], [35].
- Platform as a Service (PaaS): Tässä pilvimallissa palveluntarjoaja tarjoaa palveluna tietokonealustan ja ratkaisupinon (solution stack), usein myös väliohjelmiston (middleware). Organisaatiot voivat rakentaa sovelluksen tai palvelun tälle alustalle. Pilvipalveluntarjoaja toimittaa sovelluksen isännöintiin tarvittavat verkot, palvelimet ja tallennustilan, kun taas loppukäyttäjä valvoo ohjelmistojen käyttöönottoa ja kokoonpanoasetuksia. [33], [34], [35].
- Infrastructure as a Service (IaaS): Tässä pilvimallissa kolmannen osapuolen tarjoaja eliminoi organisaatioiden tarpeen ostaa palvelimia, verkkoja tai tallennuslaitteita tarjoamalla tarvittavan infrastruktuurin. Organisaatiot puolestaan hallinnoivat ohjelmistojaan ja sovelluksiaan ja maksavat vain tarvitsemastaan kapasiteetista. [33], [34], [35].

Vaikka ei ole kahta samanlaista pilveä, on olemassa useita yhteisiä pilviarkkitehtuurimalleja. Näitä ovat julkiset, yksityiset, hybridi- ja monipilviarkkitehtuurit. Näin ne vertautuvat toisiinsa:

- Julkinen pilviarkkitehtuuri (public cloud architecture): Julkisessa pilviarkkitehtuurissa laskentaresurssit ovat pilvipalveluntarjoajan omistuksessa ja hallinnassa. Nämä resurssit jaetaan uudelleen useiden käyttäjien kesken internetin kautta. Julkisen pilven etuja ovat alhaiset käyttökustannukset, helppo skaalautuvuus ja vähäinen huollon tarve. [33], [36], [37].
- Yksityinen pilviarkkitehtuuri (private cloud architecture): Yksityinen pilvipalvelu viittaa pilveen, joka on yksityisesti omistettu ja jota hallitaan yleensä yrityksen omissa tiloissa. Yksityinen pilvi voi kuitenkin kattaa myös useita palvelinsijainteja tai vuokrattua tilaa sijoiteltuna maantieteellisesti hajallaan oleviin tiloihin. Vaikka yksityinen pilviarkkitehtuuri on tyypillisesti kalliimpaa kuin julkiset pilviratkaisut, se on muokattavampi ja voi tarjota parempaa tietoturvaa ja täyttää paremmin organisaation vaatimukset. [33], [36], [37].
- Hybridipilvi (hybrid cloud architecture): Hybridiympäristö yhdistää julkisen pilven toiminnan tehokkuuden ja yksityisen pilven tietoturvaominaisuudet. Hyödyntämällä sekä julkisia että yksityisiä pilviarkkitehtuureja hybridipilvet

auttavat yhdistämään IT-resursseja ja mahdollistavat organisaatioiden siirtää työmäärää eri ympäristöjen välillä riippuen IT- ja tietoturvavaatimuksista. [33], [36], [37].

- Monipilviarkkitehtuuri (multi-cloud architecture): Monipilviarkkitehtuuri käyttää useita julkisia pilvipalveluja. Monipilviympäristön etuihin kuuluu suurempi joustavuus valita ja ottaa käyttöön pilvipalvelut, jotka todennäköisesti täyttävät erilaiset organisaation vaatimukset. Toinen positiivinen puoli on vähentynyt riippuvuus yksittäisistä pilvipalvelujen tarjoajista, mikä säästää kustannuksia ja pienentää todennäköisyyttä toimittajaloukkuun. Lisäksi monipilviarkkitehtuuria voidaan tarvita mikropalvelupohjaisten konttiratkaisujen tukemiseen, jos palvelut sijaitsevat useissa pilvissä. [33], [36], [37].

Pilviarkkitehtuurin peruskomponentteja ovat:

- Virtualisointi: Pilvet perustuvat palvelimien, tallennustilan ja verkkojen virtualisointiin. Virtualisoidut resurssit ovat ohjelmistopohjaisia tai virtuaalisia esityksiä fyysisestä resurssista, kuten palvelimista tai tallennustilasta. Tämä abstraktinen taso mahdollistaa useiden sovellusten käyttää samoja fyysisiä resursseja ja parantaa siten palvelimien, tallennustilan ja verkottumisen tehokkuutta. [33], [1].
- Infrastruktuuri: Pilvi-infrastruktuuri sisältää kaikki perinteisten palvelinkeskusten komponentit, mukaan lukien palvelimet ja tallennuslevyt sekä verkkolaitteet mukaan lukien reitittimet ja kytkimet. [33].
- Väliohjelmisto (middleware): Kuten perinteisissä konesaleissa, nämä ohjelmistokomponentit, kuten tietokannat ja viestintäsovellukset, mahdollistavat verkkoon liitettyjen tietokoneiden, sovellusten ja ohjelmistojen kommunikoinnin keskenään. [33].
- Hallinta: Nämä työkalut mahdollistavat pilviympäristön suorituskyvyn ja kapasiteetin jatkuvan seurannan. IT-tiimit voivat seurata käyttöä, ottaa käyttöön uusia sovelluksia, hallinnoida dataa ja varmistaa varmuuskopioinnin ja palautusmahdollisuudet. [33].
- Automaatio-ohjelmisto: Kriittisten IT-palvelujen toimittaminen automaation ja ennalta määriteltyjen käytäntöjen avulla voi helpottaa merkittävästi työkuormia, virtaviivaistaa sovellusten toimittamista ja alentaa kustannuksia. Pilviarkkitehtuurissa automaatiolla voidaan helposti skaalata järjestelmäresursseja vastaamaan laskentatehon kysyntää, ottaa käyttöön

sovelluksia vastaamaan vaihtelevia markkinoiden vaatimuksia tai varmistaa hallinto pilviympäristössä. [33].

2.2.1 Mikropalveluarkkitehtuuri

Mikropalveluarkkitehtuuri on lähestymistapa, jossa yksi sovellus koostuu monista löyhästi kytketyistä ja itsenäisesti käyttöön otettavista pienistä palveluista. Jokainen palvelu toimii omassa prosessissaan ja kommunikoi ohjelmointirajapintojen avulla (esim. REST). Nämä palvelut on rakennettu liiketoimintaominaisuuksien ympärille ja ovat itsenäisesti käyttöönotettavissa. Mikropalveluja hallitaan keskitetysti ja ne voidaan kirjoittaa eri ohjelmointikielillä eri tiimien toimesta. [38], [39].

Mikropalveluiden hyödyiksi voidaan lukea:

- Ketteryys (agility): Koska mikropalvelut otetaan käyttöön itsenäisesti, on helpompi hallita ohjelmointivirheiden korjauksia ja ominaisuuksien julkaisuja. Palveluja voidaan päivittää siirtämättä koko sovellusta uudelleen ja peruuttaa päivitys, jos jokin menee pieleen. Monissa perinteisissä sovelluksissa, jos virhe löytyy sovelluksesta, se voi estää koko julkaisuprosessin. Uusia ominaisuuksia joudutaan viivästyttämään, kun korjaus integroidaan, testataan ja julkaistaan. [40].
- Pienet, keskittyneet tiimit: Mikropalvelun tulee olla riittävän pieni, jotta yksi tiimi voi rakentaa, testata ja ottaa sen käyttöön. Pienet tiimikoot lisäävät ketteryyttä. Suuret tiimit ovat yleensä vähemmän tuottavia, koska viestintä on hitaampaa, hallinnon yleiskustannukset nousevat ja ketteryys vähenee. [40], [41].
- Pieni koodiperusta: Monoliittisessa sovelluksessa koodi pirstaloituu ajan myötä. Uuden ominaisuuden lisääminen vaatii koodin muuttamista monissa paikoissa. Mikropalveluarkkitehtuuri minimoi riippuvuudet ja helpottaa uusien ominaisuuksien lisäämistä, koska se ei jaa koodia tai tietovarastoja. [40], [41].
- Tekniikoiden monimuotoisuus: Tiimit voivat valita palveluunsa parhaiten sopivat tekniikat. [40], [41].
- Virheiden eristäminen: Jos yksittäinen mikropalvelu ei ole käytettävissä, se ei häiritse koko sovellusta, kunhan kaikki sitä käyttävät mikropalvelut on suunniteltu käsittelemään viat oikein. [40], [41].

- Skaalautuvuus: Palveluja voidaan skaalata itsenäisesti, jolloin voidaan skaalata enemmän resursseja vaativia osajärjestelmiä skaalaamatta koko sovellusta [40], [41].
- Tietojen eristäminen: Skeemojen päivittäminen on paljon helpompaa, koska päivitys koskee vain yhtä mikropalvelua. Monoliittisessa sovelluksessa skeemojen päivityksistä voi tulla erittäin haastavia, koska sovelluksen eri osat voivat kaikki käyttää samoja tietoja, mikä tekee skeema muutoksista riskialttiita. [40].
- Hahmotettavuus: Modulaarisuus helpottaa kokonaisuuden hahmotettavuutta; yksi palvelu hoitaa yhden asian hyvin (Do One Thing And Do It Well -periaate).

Mikropalveluiden haasteita ovat:

- Monimutkaisuus: Mikropalvelusovelluksessa on enemmän liikkuvia osia kuin vastaavassa monoliittisessa sovelluksessa. Jokainen palvelu on yksinkertaisempi, mutta koko järjestelmä kokonaisuutena on monimutkaisempi. [40], [41].
- Kehitys ja testaus: Pienen palvelun kirjoittaminen, joka perustuu muihin riippuvaisiin palveluihin, vaatii erilaista lähestymistapaa kuin perinteisen monoliittisen tai kerrostetun sovelluksen kirjoittaminen. Olemassa olevia työkaluja ei ole aina suunniteltu toimimaan palveluiden riippuvuuksien kanssa. Uudelleenrakentaminen palvelurajojen yli voi olla vaikeaa. Palveluiden riippuvuuksien testaaminen on myös haastavaa, varsinkin jos sovellus kehittyy nopeasti. [40], [41].
- Hallinnon puute: Hajautetulla lähestymistavalla mikropalvelujen rakentamiseen on etuja, mutta se voi myös johtaa ongelmiin. Saatetaan päätyä niin moniin eri ohjelmointikieliin ja -kehyksiin, että sovellusta on vaikea ylläpitää. Voi olla hyödyllistä ottaa käyttöön joitakin koko sovellusta koskevia standardeja rajoittamatta liikaa tiimien joustavuutta. [40], [41].
- Verkon ruuhkautuminen ja latenssi: Monien pienten, rakeisten (granular) palvelujen käyttö voi lisätä palveluiden välistä viestintää. Lisäksi, jos palveluiden riippuvuusketju tulee liian pitkäksi (palvelu A kutsuu B:tä, joka kutsuu C:tä jne.), viiveestä voi tulla ongelma. Sovellusliittymät on suunniteltava huolellisesti. On syytä välttää liian ketjuttuneita ohjelmointirajapintoja. [40].
- Tietojen eheys: Jokainen mikropalvelu vastaa omasta datan eheydestään. Tämän seurauksena tietojen johdonmukaisuus voi olla haaste. [40].

- Jäljitettävyys: Mikropalveluiden hallinta edellyttää huolellista DevOps-kulttuuria. Palveluihin liittyvä lokien kirjoitus voi olla haastavaa. Tyypillisesti lokiin on pystyttävä kirjaamaan useita palvelukutsuja yhdeltä käyttäjältä. [40].
- Versionhallinta: Palvelun päivitykset eivät saa rikkoa palveluita, jotka ovat riippuvaisia siitä. Useita palveluita voidaan päivittää milloin tahansa, joten ilman huolellista suunnittelua yhteensopivuudessa voi olla ongelmia. [40].
- Tietotaidot: Mikropalvelut ovat erittäin hajautettuja järjestelmiä, joten on syytä arvioida huolellisesti, onko tiimillä taitoja ja kokemusta niiden rakentamiseen ja ylläpitoon. [40].
- Kommunikaatio: Useita palveluja kattavien käyttötapausten käsittely on vaikeaa ja se vaatii myös viestintää ja yhteistyötä eri tiimien välillä [41].

2.2.2 Tapahtumapohjainen arkkitehtuuri (event-driven architecture)

Tapahtumapohjainen arkkitehtuurimalli on suosittu hajautettu asynkroninen arkkitehtuurimalli, jota käytetään erittäin skaalautuvien sovellusten tuottamiseen. Se on myös erittäin mukautuva ja sitä voidaan käyttää niin pienissä kuin suurissa, monimutkaisissa sovelluksissa. Tapahtumapohjainen arkkitehtuuri koostuu irrotetuista, yksikäyttöisistä tapahtumakäsittelykomponenteista, jotka vastaanottavat ja käsittelevät tapahtumia asynkronisesti. [42].

Tapahtuma on mikä tahansa merkittävä tapahtuma tai järjestelmän tai ohjelmiston tilan muutos. Tapahtuma ei ole sama asia kuin tapahtumailmoitus, joka on järjestelmän lähettämä viesti tai ilmoitus, joka ilmoittaa järjestelmän toiselle osalle tapahtumasta. Tapahtuman lähde voi olla sisäinen tai ulkoinen syöte. Tapahtumat voivat tulla käyttäjältä, kuten hiiren napsautus tai näppäinpainallus, ulkoisesta lähteestä, kuten sensorista, tai järjestelmästä, kuten ohjelman lataaminen. [43].

Tapahtumapohjainen arkkitehtuuri koostuu tapahtumien tuottajista ja kuluttajista. Tapahtuman tuottaja havaitsee tapahtuman ja esittää tapahtuman viestinä. Se ei tunne tapahtuman kuluttajaa tai tapahtuman tulosta. Kun tapahtuma on havaittu, se välitetään tapahtuman tuottajalta tapahtuman kuluttajille tapahtumakanavien kautta, joissa tapahtumankäsittelyalusta käsittelee tapahtuman asynkronisesti. Tapahtuman kuluttajille on ilmoitettava tapahtumasta. Ne saattavat käsitellä tapahtuman tai tapahtuma voi vaikuttaa niihin. Tapahtumienkäsittelyalusta käsittelee tapahtuman ja lähettää aktiviteetin loppupään kuluttajille. Tässä loppupään toiminnassa nähdään

tapahtuman tulos. Tuottajat ja kuluttajat voivat olla mikropalveluita ja hyvinkin riippumattomia toisistaan. [43].

Tapahtumapohjaisen arkkitehtuurin hyötyjä ovat:

- Skaalautuminen ja epäonnistuminen itsenäisesti: Erottamalla palvelut, palvelut tuntevat vain tapahtumavälittäjän, eivät toisiaan. Tämä tarkoittaa, että palvelut ovat yhteen toimivia, mutta jos yhdessä palvelussa on vikaa, muut jatkavat toimintaansa. Tapahtumavälittäjä toimii joustavana puskurina, joka sopeutuu työkuormien kasvuun. [44], [45].
- Ketterä kehitys: Mukautettua koodia ei tarvitse kirjoittaa tapahtumien kyselyyn, suodatukseen ja välitykseen. Tapahtumavälittäjä suodattaa ja lähettää tapahtumat automaattisesti kuluttajille. Välittäjä poistaa myös tuottajien ja kuluttajapalvelujen välisen voimakkaan koordinoinnin tarpeen, mikä nopeuttaa kehitysprosessia. [44].
- Auditoinnin helppous: Tapahtumavälittäjä toimii keskitetysti auditoiden sovelluksen ja määritellen toimintatavat. Nämä toimintatavat voivat rajoittaa sitä, miten tapahtumat (käyttäjät) voivat julkaista ja tilata tietoja välittäjän kautta, sekä hallita, millä käyttäjillä ja resursseilla on oikeus käyttää tietoja. Tapahtumia voidaan myös salata sekä kuljetuksen aikana että lepotilassa. Single source of truth -periaate on oleellinen osa tietojen oikeellisuuden varmistamisessa ja auditoinnissa. [44], [45].
- Kustannustehokkuus: Tapahtumapohjaiset arkkitehtuurit ovat push-pohjaisia, joten kaikki tapahtuu tarpeen mukaan, kun tapahtuma esiintyy. Tällä tavalla ei makseta jatkuvista kyselyistä tapahtuman tarkistamiseksi. Tämä tarkoittaa vähemmän verkon kaistanleveyden kulutusta, vähemmän suorittimen käyttöä, vähemmän toimetonta kapasiteettia ja vähemmän SSL/TLS-kättelyjä. [44], [45].

Tapahtumapohjainen arkkitehtuuri on suhteellisen monimutkainen malli toteuttaa pääasiassa asynkronisen hajautetun luonteensa vuoksi. Mallia käyttöön otettaessa on mietittävä erilaisia hajautettuun arkkitehtuuriin liittyviä ongelmia, kuten etäprosessin saatavuutta, viivettä ja välittäjän uudelleenkytkentälogiikkaa välittäjän vian sattuessa. [42].

Yksi näkökulma, joka on otettava huomioon tätä arkkitehtuurimallia valittaessa, on toimintavarmuuden puute yhdelle prosessille. Koska tapahtumavälittäjäkomponentit on irrotettu ja hajautettu voimakkaasti, on erittäin vaikeaa ylläpitää transaktioita niiden

välillä. Tästä syystä, kun sovellusta suunnitellaan tämän mallin mukaisesti, on jatkuvasti pohdittava, mitkä tapahtumat voivat ja eivät voi suoriutua itsenäisesti, ja suunniteltava tapahtumakäsittelijöiden yksityiskohtaisuus sen mukaan. Jos yksi työyksikkö on jaettava tapahtumavälittäjien kesken, eli jos käytetään erillisiä välittäjiä/käsittelijöitä johonkin, jonka pitäisi olla jakamaton tapahtuma, tämä ei todennäköisesti ole oikea malli sovellukselle. [42].

Kenties yksi tapahtumapohjaisen arkkitehtuurimallin vaikeimmista osista on tapahtuma-välittäjäkomponenttisopimusten luominen, ylläpito ja hallinta. Jokaiseen tapahtumaan liittyy yleensä erityinen sopimus (esim. data-arvot ja tietomuoto välitetään tapahtumavälittäjälle). Tätä mallia käytettäessä on elintärkeää ratkaista vakiomuotoinen datamuoto (esim. XML, JSON, Java-objekti jne. [45]) ja luoda sopimuksen versiokäytäntö heti alusta alkaen. [42].

Pilvipalvelut tukevat tapahtumapohjaista arkkitehtuuria ennen kaikkea palvelittoman (serverless) toteutuksen kautta. Erilaisia palvelittomia ratkaisuja tapahtumapohjaisen arkkitehtuurin toteutukseen ovat:

Functions-as-a-Service (FaaS): FaaS on pilvilaskentapalvelu, jonka avulla kehittäjät voivat rakentaa, laskea, ajaa ja hallita sovelluspaketteja funktioina ilman, että heidän tarvitsee ylläpitää omaa infrastruktuuriaan. FaaS on tapahtumapohjainen suoritusmalli, joka toimii konteissa (container) ja nämä toiminnot hallitsevat palvelinpuolen logiikkaa ja tilaa käyttämällä FaaS-palveluntarjoajan palveluita. FaaS-ratkaisuja on saatavana suurista julkisista pilvistä, ja ne voidaan järjestää paikan päällä, mikä lisää merkittäviä uusia ominaisuuksia yrityksen IT-sovellusten kehittämiseen. [46], [47], [49].

Backend-as-a-Service (BaaS): BaaS on pilvipalvelumalli, jossa kehittäjät ulkoistavat kaikki verkko- tai mobiilisovelluksen kulissien takana olevat asiat niin, että heidän tarvitsee vain kirjoittaa ja ylläpitää käyttöliittymää. BaaS-toimittajat tarjoavat valmiiksi kirjoitettua ohjelmistoa palvelimilla tapahtuvaan toimintaan, kuten käyttäjän todentamiseen, tietokannan hallintaan, etäpäivitykseen ja push-ilmoituksiin (mobiilisovelluksiin) sekä pilvitallennukseen ja -hotelliin. [48], [49].

Mobile-Backend-as-a-Service (MBaaS): MBaaS on BaaS, joka on tarkoitettu erityisesti mobiilisovellusten rakentamiseen. Joidenkin lähteiden mukaan BaaS ja MBaaS ovat periaatteessa keskenään samaa tarkoittavia termejä, mutta BaaS-palveluita ei välttämättä tarvitse käyttää mobiilisovellusten rakentamiseen. [48], [49].

2.2.3 Saga

Saga-suunnittelumalli on tapa hallita tietojen yhdenmukaisuutta mikropalveluiden välillä hajautetuissa tapahtumaskenaarioissa. Saga on tapahtumasarja, joka päivittää jokaisen palvelun ja julkaisee viestin tai tapahtuman käynnistääkseen seuraavan tapahtumavaiheen. Jos vaihe epäonnistuu, saga suorittaa korvaavat tapahtumat, jotka peruuttavat edelliset tapahtumat. [50].

Tapahtumien on oltava atomisia, eheitä, eristettyjä ja pysyviä (atomic, consistent, isolated, and durable (ACID)). Yhden palvelun sisällä tapahtuvat tapahtumat ovat ACID-muotoisia, mutta palvelujen välinen tietojen johdonmukaisuus edellyttää palvelujen välisen tapahtuman hallintastrategiaa. [50].

Monipalveluarkkitehtuureissa:

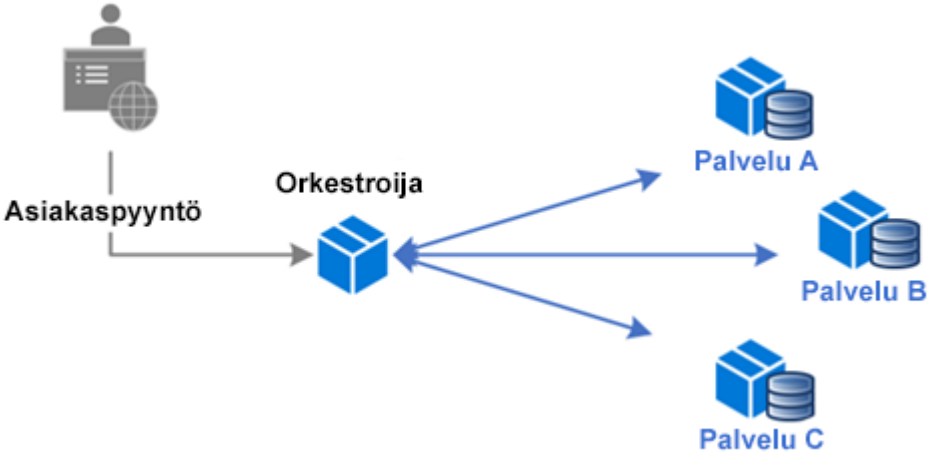
- Atomisuus on jakamaton ja pelkistymätön joukko toimintoja, joiden on täytyttävä tai niitä ei tapahdu [50].
- Eheys tarkoittaa, että tapahtuma tuo tiedot vain yhdestä kelvollisesta tilasta toiseen kelvolliseen tilaan [50].
- Eristyneisyys takaa, että samanaikaiset tapahtumat tuottavat saman tietotilan, jonka peräkkäiset suoritukset olisivat tuottaneet [50].
- Pysyvyys varmistaa, että toteutuneet tapahtumat pysyvät toteutuneina myös järjestelmävian tai sähkökatkon sattuessa [50].

Saga on sarja paikallisia tapahtumia. Jokainen paikallinen tapahtuma päivittää tietokannan ja julkaisee viestin tai tapahtuman, joka käynnistää seuraavan paikallisen tapahtuman. Jos paikallinen tapahtuma epäonnistuu, saga suorittaa sarjan korvaavia tapahtumia, jotka kumoavat edellisten paikallisten tapahtumien tekemät muutokset. [51].

Saga-mallissa korvattavat tapahtumat ovat transaktioita, jotka voidaan mahdollisesti peruuttaa käsittelemällä toinen tapahtuma, jolla on päinvastainen vaikutus. Pivot-tapahtuma on sagan go/no-go-piste. Jos pivot-tapahtuma toteutuu, saga kestää loppuun asti. Pivot-tapahtuma voi olla tapahtuma, jota ei voida korvata eikä yrittää uudelleen, tai se voi olla viimeinen korvattava tapahtuma tai ensimmäinen toistettava tapahtuma. Toistettavat tapahtumat ovat tapahtumia, jotka seuraavat pivot-tapahtumaa ja joiden taataan onnistuvan. [50].

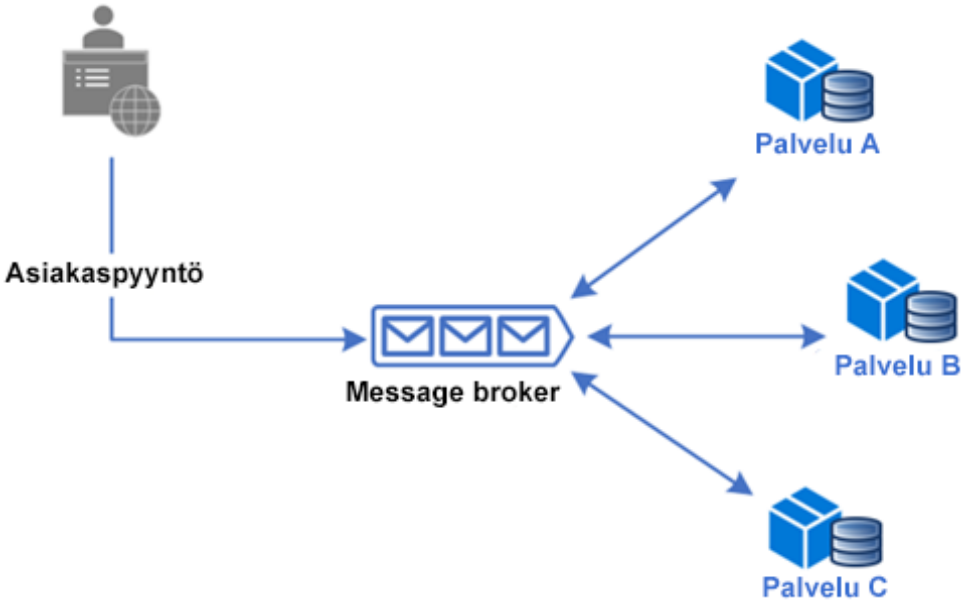
Sagasta on käytössä kaksi mallia:

- Orchestration-based saga: Tässä lähestymistavassa on saga-orkestroija, joka hallitsee kaikkia tapahtumia ja ohjaa osallistujapalvelut suorittamaan paikallisia transaktioita tapahtumien perusteella. Orkestroija suorittaa saagapyynnöt, tallentaa ja tulkitsee kunkin tehtävän tilat ja hoitaa virheenpalautuksen kompensoivilla tapahtumilla. [52], [50].



Kuva 2. Orchestration-based saga [50].

- Choreography-based saga: Tässä lähestymistavassa ei ole keskitettyä orkestroijaa. Jokainen sagaan osallistuva palvelu suorittaa transaktionsa ja julkaisee tapahtumansa. Muut palvelut reagoivat näihin tapahtumiin ja suorittavat transaktionsa. Ne voivat myös julkaista tai olla julkaisematta muita tapahtumia tilanteen mukaan. Jokainen paikallinen tapahtuma julkaisee alatapahtumia, jotka käynnistävät paikallisia tapahtumia muissa palveluissa. [52], [50].



Kuva 3. Choreography-based saga [50].

2.2.4 Pilvinatiivi arkkitehtuuri (cloud native architecture)

Pilvipohjaiset teknologiat antavat organisaatioille mahdollisuuden rakentaa ja ajaa skaalautuvia sovelluksia moderneissa, dynaamisissa ympäristöissä, kuten julkisissa, yksityisissä ja hybridipilvissä. Kontit (containers), palveluverkot, mikropalvelut, muuttumaton infrastruktuuri ja deklaratiiviset APIt ovat esimerkkejä tästä lähestymistavasta. Nämä tekniikat mahdollistavat löyhästi toisiinsa kytketyt järjestelmät, jotka ovat joustavia, hallittavissa ja havaittavissa. Yhdessä automaation kanssa ne antavat kehittäjien tehdä suuria muutoksia usein ja ennustettavasti vähäisellä vaivalla. [53].

Pilvinatiivissa arkkitehtuurissa on kyse nopeudesta ja ketteryydestä. Liiketoimintajärjestelmät kehittyvät mahdollistavista liiketoimintaominaisuuksista strategisen muutoksen aseiksi, jotka nopeuttavat liiketoiminnan nopeutta ja kasvua. On välttämätöntä saada uusia ideoita markkinoille välittömästi. Samaan aikaan myös liiketoimintajärjestelmät ovat muuttuneet yhä monimutkaisemmiksi käyttäjien vaatiessa enemmän. Käyttäjät odottavat nopeaa reagointikykyä, innovatiivisia ominaisuuksia ja korkeaa saatavuutta. Suorituskykyongelmat, toistuvat virheet ja kyvyttömyys nopeisiin muutoksiin eivät ole enää hyväksyttäviä; muutoin käyttäjät hakeutuvat kilpailijoiden luo. Pilvipohjaiset järjestelmät on suunniteltu ottamaan vastaan nopeita muutoksia, laajamittaisuutta ja joustavuutta. [54].

Pilvinatiivin arkkitehtuurin suunnitteluperiaatteet:

- Suunniteltu löyhästi yhdistetyiksi mikropalveluiksi. Mikropalvelut ovat tapa kehittää yksi sovellus sarjana pieniä palveluita, joista jokainen toimii omassa prosessissaan ja kommunikoi kevyiden protokollien, kuten HTTP:n, avulla. Nämä palvelut rakentuvat liiketoimintaominaisuuksien ympärille ja ne voidaan ottaa käyttöön itsenäisesti täysin automatisoiduilla käyttöönottokoneistoilla. [55], [38], [39].
- Kehitetty parhaiden kielten ja viitekehysten avulla. Kukin pilvipohjaisen sovelluksen palvelu kehitetään käyttämällä toiminnallisuuksiin parhaiten soveltuvaa kieltä ja viitekehystä. Pilvipohjaiset sovellukset ovat monikielisiä. Palvelut käyttävät useita kieliä, ajoaikoja ja kehyksiä. Kehittäjät voivat esimerkiksi rakentaa Node.js:ssä kehitettyyn WebSockettiin perustuvan reaaliaikaisen suoratoistopalvelun, samalla kun he valitsevat Pythonin koneoppimiseen perustuvan palvelun rakentamiseen ja Spring-bootin REST-sovellusliittymien tekoon. Hienostunut lähestymistapa mikropalvelujen kehittämiseen antaa kehittäjille mahdollisuuden valita paras kieli ja puitteet tiettyä työtä varten. [55], [57].
- Keskitetty APIhin vuorovaikutusta ja yhteistyötä varten. Pilvipohjaiset palvelut esittävät toiminnallisuutensa käyttämällä kevyitä API-liittymiä, jotka perustuvat API arkkitehtuuriin, kuten REST. Sisäiset palvelut kommunikoivat keskenään käyttämällä binääriprotokollia, kuten Thrift, Protobuff, GRPC jne. parantaakseen suorituskykyä. [55].
- Tilaton ja massiivisesti skaalautuva. Pilvipohjainen sovellus tallentaa tilansa tietokantaan tai johonkin muuhun ulkoiseen kokonaisuuteen, jotta instansseja voi tulla ja mennä. Mikä tahansa instanssi voi käsitellä pyynnön. Niitä ei ole sidottu taustalla olevaan infrastruktuuriin, mikä mahdollistaa sovelluksen toimimisen erittäin hajautetusti ja ylläpitää silti tilaansa taustalla olevan infrastruktuurin joustavasta luonteesta riippumatta. Skaalautuvuuden näkökulmasta arkkitehtuuri on niin yksinkertainen, että sovelluksen pitäisi olla mahdollista skaalautua vain lisäämällä palvelinklusteriin uusia palvelimia. [55].
- Resilienssi (resiliency) arkkitehtuurin ytimessä. Murphyn lain mukaan "kaikki mikä voi epäonnistua, epäonnistuu". Kun tätä sovelletaan ohjelmistojärjestelmiin, hajautetussa järjestelmässä tulee tapahtumaan virheitä. Laitteisto voi hajota. Verkossa voi olla ohimeneviä vikoja. Harvoin koko palvelu tai osa-alue on alhaalla samanaikaisesti, mutta niihinkin on varauduttava. Resilienssi on

järjestelmän kyky toipua vioista ja jatkaa toimintaansa. Kyse ei ole virheiden välttämisestä, vaan virheisiin vastaamisesta tavalla, joka välttää seisokit tai tietojen katoamisen. Resilienssin tavoitteena on palauttaa sovellus täysin toimivaan tilaan vian jälkeen. [55].

- Pakattu kevyiksi konteiksi (containers) ja orkestroitu. Konteilla on mahdollista eristää sovelluksia pieniin, kevyisiin suoritusympäristöihin, jotka jakavat käyttöjärjestelmän ytimen. Tyypillisesti megatavuina mitattuna säiliöt käyttävät paljon vähemmän resursseja kuin virtuaalikoneet ja käynnistyvät lähes välittömästi. Dockerista on tullut konttitekniikan standardi. Niiden suurin etu on siirrettävyys. Pilvipohjaiset sovellukset otetaan yleensä käyttöön Kubernetesilla, joka on avoimen lähdekoodin alusta, joka on suunniteltu automatisoimaan konttisovellusten käyttöönottoa, skaalausta ja hallintaa. Alun perin Googlen kehittämästä Kubernetesista on tullut käyttöjärjestelmä pilvipohjaisten sovellusten käyttöönotolle. Se on myös yksi ensimmäisistä projekteista, jotka valmistuivat CNCF:ssä (Cloud Native Computing Foundation). [55], [56], [57].
- Ketterä DevOps ja automaatio. DevOps kuvaa organisaatiorakennetta, käytäntöjä ja kulttuuria, joita tarvitaan nopean ketterän kehityksen ja skaalautuvan, luotettavan toiminnan mahdollistamiseksi. DevOpsissa on kyse kulttuurista, yhteistoiminnallisista käytännöistä ja automaatiosta, joka yhdistää kehitys- ja operatiiviset tiimit niin, että tiimeillä on yksi ajattelutapa asiakaskokemuksen parantamiseksi, nopeammista liiketoiminnan tarpeista vastaamiseksi ja sen varmistamiseksi, että innovaatiot ovat tasapainossa tietoturvan ja toiminnallisten tarpeiden kanssa. Nykyaikaiset organisaatiot uskovat kehitys- ja operatiivisten ihmisten ja vastuiden yhdistämiseen niin, että yksi DevOps-tiimi kantaa molemmat vastuut. Tällä tavalla on vain yksi tiimi, joka ottaa vastuun ohjelmiston kehittämisestä, käyttöönotosta ja käytöstä tuotannossa. [55], [56], [57].

  DevOpsissa jatkuva integrointi (CI) ja jatkuva toimitus (CD) ovat joukko toimintaperiaatteita, joiden avulla sovelluskehitystiimit voivat toimittaa koodimuutokset useammin ja luotettavammin. CI:n tekninen tavoite on luoda johdonmukainen ja automatisoitu tapa rakentaa, pakata ja testata sovelluksia. Kun integrointiprosessi on johdonmukainen, tiimit tekevät todennäköisemmin koodimuutoksia useammin, mikä johtaa parempaan yhteistyöhön ja ohjelmistojen laatuun. [55], [56], [57].

Jatkuva toimitus jatkuu siellä, missä jatkuva integraatio päättyy. CD automatisoi sovellusten toimituksen valittuihin infrastruktuuriympäristöihin. Se poimii CI:n rakentaman paketin, julkaisee sen käyttöön useissa ympäristöissä. Paketille suoritetaan erilaisia testejä, kuten integraatiotestejä, suorituskykytestejä jne., ja lopulta se otetaan käyttöön tuotantoon. Jatkuvassa toimituksessa on yleensä vain vähän manuaalisia vaiheita, kun taas jatkuva käyttöönotto on täysin automatisoitu prosessi, joka automatisoi koko prosessin koodin tarkistuksesta tuotannon käyttöönottoon. [55], [56].

- Elastinen ja dynaaminen skaalautuminen. Pilvipohjaiset sovellukset hyödyntävät pilven joustavuutta käyttämällä lisääntyneitä resursseja käyttöpiikin aikana. Jos pilvipohjaisessa sovelluksessa on käytössä piikki, voidaan se asettaa käyttämään ylimääräisiä laskentaresursseja, kunnes piikki laantuu, ja sitten sammuttaa nämä resurssit. Pilvipohjainen sovellus voi mukautua lisääntyneisiin resursseihin ja mittakaavaan tarpeen mukaan. [55], [56], [57].

Twelve-factor application on laajasti hyväksytty menetelmä pilvinatiivien sovellusten rakentamiseen. Twelve-factor metodologia koostuu pitkälti samoista suunnitteluperiaatteista, joita edellä on esitetty. Se kuvaa joukon periaatteita ja käytäntöjä, joita kehittäjät noudattavat rakentaessaan sovelluksia, jotka on optimoitu nykyaikaisiin pilviympäristöihin. Erityistä huomiota kiinnitetään siirrettävyyteen eri ympäristöissä ja deklaratiiviseen automaatioon. [54].

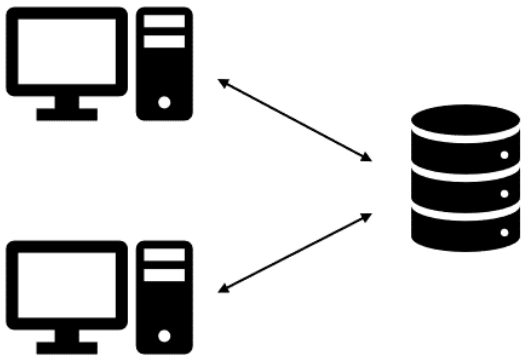
Vaikka twelve-factor application soveltuu kaikkiin verkkopohjaisiin sovelluksiin, monet pitävät twelve-factoria vankkana perustana pilvipohjaisten sovellusten rakentamiselle. Näille periaatteille rakennetut järjestelmät voivat julkaista päivityksiä, skaalautua nopeasti ja lisätä ominaisuuksia, joilla voidaan reagoida nopeasti markkinoiden muutoksiin. [54].

2.2.5 Asiakas-palvelin-malli

Asiakas-palvelin-malli ja monoliittinen arkkitehtuuri ovat perinteisiä tapoja toteuttaa sovelluksia, joidenka päälle monet uudemmat mallit ovat rakentuneet. Edellä mainitut uudemmat mallit ovat osittain syrjäyttäneet näitä perinteisempiä toteutusmalleja, mutta nämä perinteisemmät mallit ovat kuitenkin edelleen laajalti käytössä varsinkin hieman vanhemmissa toteutuksissa pilvipalveluiden ulkopuolella kuin myös yksinkertaisissa, pienissä sovelluksissa.

Monoliittinen sovellus on sovellus, jossa käyttöliittymä, toimintalogiikka ja tietokantalogiikka yhdistetään yhdeksi suoritettavaksi ohjelmaksi ja otetaan käyttöön yhdellä alustalla. Monoliittinen sovellus toimii muista sovelluksista riippumattomasti ja suorittaa kaikki prosessin vaiheet, joita tarvitaan koko liiketoimintakokonaisuuden suorittamiseen. Se ei jaa logiikkaa tai dataa järjestelmän tai organisaation rajojen yli. Tietokannat on suunniteltu käytettäviksi yhden sovelluksen sisällä, ei käytettäviksi samanaikaisesti useissa sovelluksissa. [58].

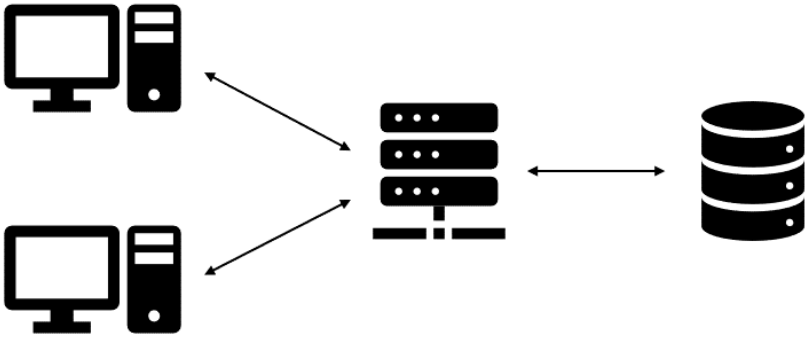
Yksinkertaisinta asiakas-palvelin-arkkitehtuuria kutsutaan kaksitasoiseksi arkkitehtuuriksi (two-tier architecture). Itse asiassa useimmat asiakas-palvelin-arkkitehtuurit ovat kaksitasoisia. Termi "kaksitasoinen" kuvaa tapaa, jolla sovelluskäsittely voidaan jakaa asiakas-palvelinsovelluksessa. Kaksitasoisessa sovelluksessa useat päätelaitteet kommunikoivat keskitetyn palvelimen kanssa. Kuva 4 näyttää, kuinka kaksitasoiset järjestelmät antavat asiakkaille pääsyn keskitettyyn tietoon. Esimerkiksi webissä asiakaspuolen verkkoselain hakee verkkopalvelimelle tallennetut tiedot. [59].



Kuva 4. Kaksitasoinen asiakas-palvelin-arkkitehtuuri [59].

Kaksitasoista asiakas-palvelin-arkkitehtuuria voidaan laajentaa kolmanteen tasoon. Kolmikerroksinen arkkitehtuuri (three-tier architecture) lisää kaksitasoiseen malliin uuden kerroksen, joka keskittää tietojenkäsittelyn ja maksimoi objektien uudelleenkäytön. Tämä kolmas kerros mahdollistaa sen, että tietokantayhteys voidaan piilottaa päätelaitteilta palvelinobjekteille, joihin päätelaitteet ovat yhteydessä. Näin ollen, toisin kuin kaksitasoisessa arkkitehtuurissa, kyseisiä palvelinobjekteja voidaan uudelleen käyttää päätelaitteiden ja tietokannan väliseen kommunikointiin.

Keskimmäinen taso toimii ikään kuin rajapintana päätelaitteiden ja tietokannan välillä. Kuva 5 näyttää, kuinka tämä uusi kolmas kerros voisi sopia sovellusarkkitehtuuriin. [59].



Kuva 5. Kolmikerroksinen asiakas-palvelin-arkkitehtuuri [59].

Yksi suurimmista ongelmista perinteisessä asiakas-palvelin-kokoonpanossa on kuormituksen arvaamaton luonne. Kun asiakas-palvelinjärjestelmät määritellään vertikaalisesti skaalautuviksi ja keskustietovarastoja käyttäviksi järjestelmiksi, ennakoimattomien työkuormien hallinnasta tulee haasteellista. Redundanssi- ja saatavuusvyöhykkeet sekä vikasietoisuus ovat keinoja/käsitteitä, joilla voidaan pitää järjestelmät toiminnassa sujuvasti kysynnän muutoksista tai muista ongelmista huolimatta. [60].

Redundanssi tässä yhteydessä voi tarkoittaa esim. sitä, että palvelimesta pidetään offline-tilassa olevaa kopiota, joka voidaan ottaa käyttöön tarvittaessa, jos pääpalvelin on syystä tai toisesta pois käytöstä. Tämä palvelinkopio voi sijaita toisella saatavuusvyöhykkeellä joko samassa konesalissa taikka kokonaan eri konesalissa. Eri saatavuusvyöhykkeet mahdollistavat palvelun saatavuuden sillä periaatteella, että toinen saatavuusvyöhyke on aina käytössä, vaikka toinen vyöhyke olisikin ylläpidon kohteena ja näin ollen hetkellisesti poissa käytöstä.

Toinen suuri ongelma on hajautetun palvelunestohyökkäyksen (DDoS) torjumisen vaikeus. Tämäntyyppisessä hyökkäyksessä hallitsematon asiakastoiminta (suuri määrä pyyntöjä) kaataa palvelimen. Vielä pari vuosikymmentä sitten DDoS-hyökkäyksellä oli melko helppoa kaataa sivusto, koska keskimääräistä asiakas-palvelin-mallia ei ollut asetettu tietyn liikennemäärän ylittäville kynnyksille. [60].

Vertaisverkkojärjestelmät (peer-to-peer) voivat olla ratkaisu moniin näistä ongelmista ja suojata järjestelmät DDoS-hyökkäyksiltä ja vastaavilta kyberhyökkäyksiltä. Vertaisverkko on hyödyllinen myös joidenkin muiden häiriöiden käsittelyssä yksittäisen vikakohdan perusteella. Hajautettujen järjestelmien, esimerkiksi lohkoketjujen muuttumattomien tietokantojen, ilmaantumisen myötä vertaisverkkojärjestelmät ovat tulossa suositummiksi ja alkavat korvata asiakas-palvelin-arkkitehtuuria pilvipalveluiden ohella. [60].

3 PALVELINTEN NYKYTARPEET

Tämän luvun tarkoituksena on käydä palvelimia läpi yleisellä tasolla, jonka jälkeen perehdytään ohjelmiston laatuattribuutteihin. Näiden laatuattribuuttien pohjalta on tarkoitus rakentaa palvelinarkkitehtuuria tukeva laatumalli. Tämä sen vuoksi, että palvelinratkaisujen ja palvelinarkkitehtuurin laatua mittaavia malleja ei ole yleisesti saatavilla tai niitä ei ole tutkittu vastaavia määriä kuin yleisemmällä tasolla ohjelmistojen laatua. Tarkoitus on pohtia palvelinten nykytarpeita laatuattribuuttien näkökulmasta.

3.1 Palvelimet

Palvelin on tyypillisesti suuritehoinen tietokone, joka hoitaa verkkopalvelun toiminnan edellyttämät prosessit. Nämä voivat vaihdella tietokannan hallinnasta, sovelluslogiikan suorittamisesta, HTTP-pyyntöjen käsittelystä, salauksesta ja suojauksesta aina sähköpostin hallintaan. [2].

Jos sovelluksen koko on pieni eikä ole suunnitelmia projektin skaalaamiseksi, yksi palvelin voi tyypillisesti hoitaa kaikki edellä mainitut tehtävät kerralla. Tätä kutsutaan virtuaalipalvelinarkkitehtuuriksi. Sen avulla useampi kuin yksi palvelin voi toimia samalla koneella. Jos projektia kuitenkin halutaan skaalata ja varmistaa, että sovellus ei kuormitu suuresta käyttäjämäärästä, on palvelinarkkitehtuuriin kiinnitettävä erityistä huomiota. [2].

Virtualisointi on tekniikka, jonka avulla voidaan luoda useita simuloituja ympäristöjä tai jakaa resursseja yhdestä fyysisestä laitteistojärjestelmästä. Virtualisointialusta muodostaa yhteyden suoraan kyseiseen laitteistoon ja jakaa yhden järjestelmän erillisiksi ja suojatuiksi ympäristöiksi, joita kutsutaan virtuaalikoneiksi. Nämä virtuaalikoneet luottavat virtualisointialustan kykyyn erottaa koneen resurssit laitteistosta ja jakaa ne asianmukaisesti. [1].

Virtualisointialustalla varustettua fyysistä laitteistoa kutsutaan isännäksi, kun taas sen resursseja käyttävät virtuaalikoneet ovat vieraita. Nämä vieraat käsittelevät tietokoneresursseja, kuten prosessoria, muistia ja tallennustilaa, resurssikokonaisuutena, jota voidaan helposti siirtää. Virtualisointialustat voivat hallita suorittimen, muistin, tallennustilan ja muiden resurssien virtuaalisia esiintymiä, joten vieraat saavat tarvitsemansa resurssit, kun ne tarvitsevat niitä. [1].

Palvelimet voidaan jakaa itsenäisiin vastuualueisiin, jotta resurssit voidaan jakaa mahdollisimman tehokkaasti käsiteltäessä suurta määrää palvelupyyntöjä. Nämä yksittäiset palvelimet voivat hallita tehtäviään paljon suuremmassa mittakaavassa sekä luotettavammin ja tehokkaammin kuin yksi virtuaalipalvelin. [2].

Nykyisin yhä suositumpi tapa hallita dataa ja toiminnallisuuksia on sovelluksen osien tai koko sovelluksen verkkoisännöinti pilvipalvelussa. Tämä tarkoittaa pohjimmiltaan, että sen sijaan, että isännöit sovellusta paikallisesti paikan päällä olevilla palvelimilla, ulkoistat sovellusten isännöinnin organisaatiolle, joka tarjoaa pilvipalvelintoiminnot. Organisaation palvelimet hoitavat minkä tahansa osa-alueen tietojenkäsittelystä. [2].

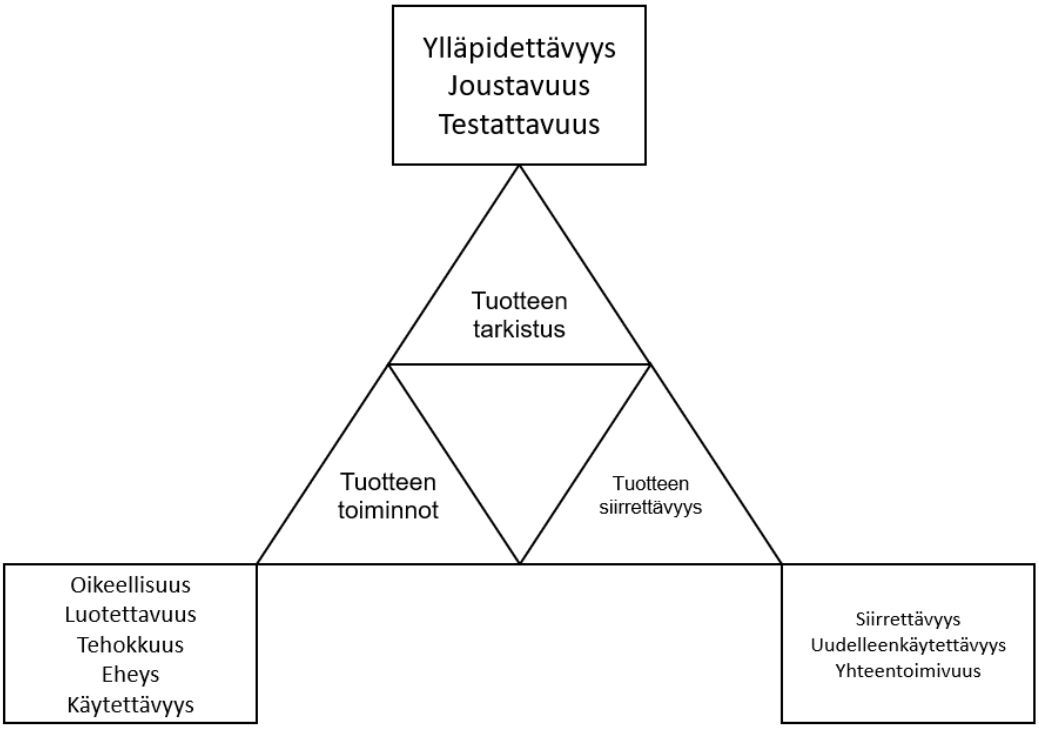
Pilvipalvelut rakentuvat pitkälti virtualisoinnin päälle. Virtualisointi luo laitteistoverkon ja resurssikokonaisuuden, pilvipalveluiden tarjoajat tarjoavat hallinta- ja käyttöjärjestelmäohjelmiston virtuaalikoneiden luomiseen ja resurssien hallintaan/varaamiseen. Nykyisin yleisin lähestymistapa palvelinten hallintaan on hybridilähestymistapa eli sekoitus paikallisen palvelimen käyttöä ja pilvipalvelua [2].

3.2 Ohjelmiston laatuattribuutit

3.2.1 Jim McCallin laatumalli (General Electrics Model 1977)

Yksi nykypäivän laatumallien tunnetuimmista edeltäjistä on Jim McCallin esittämä laatumalli [3] (tunnetaan myös nimellä General Electrics Model 1977). Tämä malli, samoin kuin moni muu nykyaikaisista malleista, on peräisin Yhdysvaltain armeijasta ja on suunnattu ensisijaisesti järjestelmän kehittäjille ja järjestelmän kehittämisprosessille [4]. Laatumallissaan McCall yrittää kaventaa kuilua käyttäjien ja kehittäjien välillä keskittymällä useisiin ohjelmistojen laatutekijöihin, jotka heijastavat sekä käyttäjien näkemyksiä että kehittäjien prioriteetteja.

McCall-laatumallilla on kolme päänäkymää ohjelmiston laadun määrittelemiseksi ja tunnistamiseksi: tuotteen tarkistus (kyky muutoksiin), tuotteen siirrettävyys (sopeutuvuus uuteen ympäristöön) ja tuotteen toiminnot (sen toimintaominaisuudet) (kuva 6.).



Kuva 6. McCall-laatumalli (McCall's Triangle of Quality) jakautuu kolmen laatuominaisuuden ympärille [4].

Tuotteen tarkistus sisältää ylläpidettävyyden (työmäärä, joka tarvitaan ohjelman vian paikantamiseen ja korjaamiseen ohjelman toimintaympäristössä), joustavuuden (toimintaympäristön muutosten edellyttämien muutosten tekemisen helppous) sekä testattavuuden (ohjelman testaamisen helppous sen varmistamiseksi, että se on virheetön ja täyttää määrittelynsä).

Tuotteen siirrettävyydessä on kyse siirrettävyydestä (työmäärä, joka tarvitaan ohjelman siirtämiseen yhdestä ympäristöstä toiseen), uudelleenkäytettävyydestä (ohjelmistojen uudelleenkäytön helppous eri tilanteissa) sekä yhteentoimivuudesta (tarvittava työmäärä järjestelmän yhdistämiseksi toiseen järjestelmään).

Tuotetoimintojen laatu riippuu oikeellisuudesta (missä määrin ohjelma täyttää määrittelynsä), luotettavuudesta (järjestelmän kyky olla epäonnistumatta), tehokkuudesta (luokiteltu edelleen suoritustehokkuuteen sekä tallennustehokkuuteen), eheydestä (suojaus ohjelman luvattomalta käytöltä) sekä käytettävyydestä (ohjelmiston käytön helppous).

3.2.2 Boehmin laatumalli (1978)

Toinen nykypäivän laatumallien perustoista on Barry W. Boehmin esittämä laatumalli [5]. Boehm käsittelee puutteita malleissa, jotka arvioivat ohjelmistojen laatua automaattisesti ja kvantitatiivisesti. Pohjimmiltaan hänen mallinsa yrittää määritellä ohjelmiston laadun laadullisesti tietyillä määritteillä ja mittareilla. Boehmin malli on samanlainen kuin McCallin laatumalli siinä mielessä, että se on myös hierarkkinen laatumalli, joka on rakennettu korkean tason ominaisuuksien, keskitason ominaisuuksien ja primitiivisten ominaisuuksien ympärille, joista jokainen vaikuttaa yleiseen laatutasoon. [4].

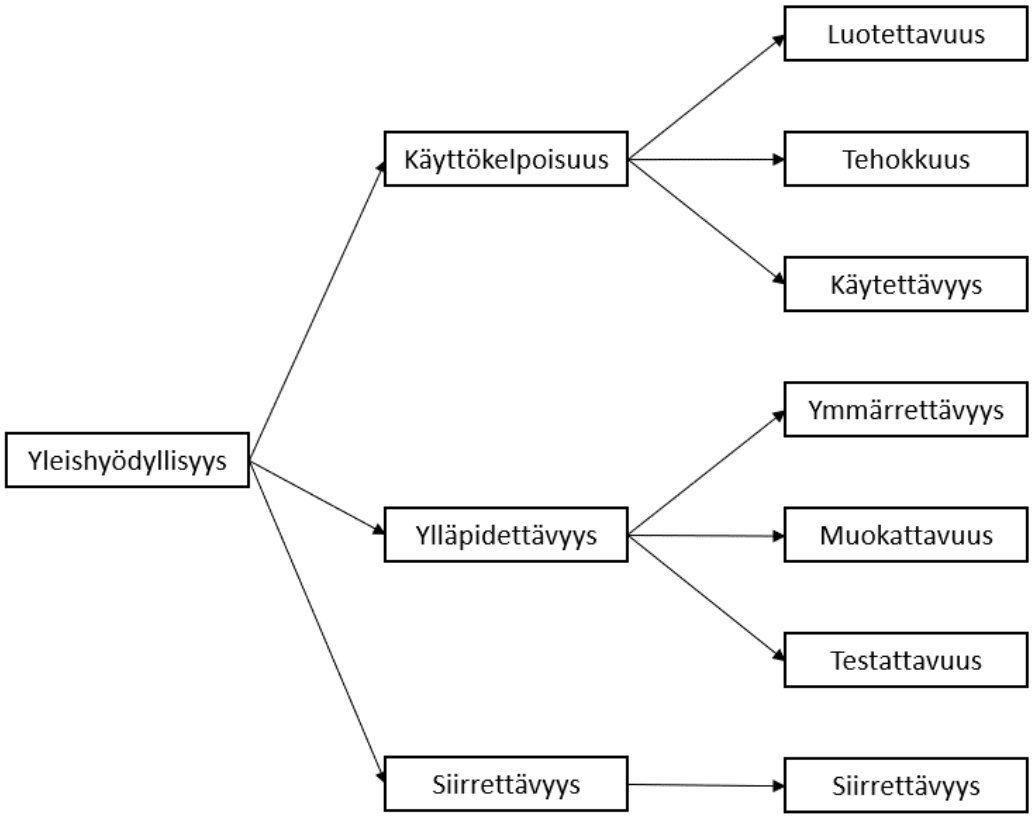
Korkean tason ominaisuudet edustavat käytön perusvaatimuksia, joihin ohjelmiston laadun arviointi voidaan perustaa. Korkean tason ominaisuuksissa käsitellään kolmea pääkysymystä, jotka ohjelmiston ostajalla on:

- Käyttökelpoisuus: Kuinka hyvin (helposti, luotettavasti, tehokkaasti) ohjelmaa voidaan käyttää sellaisenaan.
- Ylläpidettävyys: Kuinka helppoa on ymmärtää, muokata ja testata.
- Siirrettävyys: Voiko ohjelmaa käyttää myös ympäristön vaihtuessa.

Keskitason ominaisuus edustaa Boehmin seitsemää laatutekijää, jotka yhdessä edustavat ohjelmistojärjestelmältä odotettuja ominaisuuksia:

- Siirrettävyys (yleinen ominaisuus): Koodi on siinä määrin siirrettävissä, että sitä voidaan käyttää helposti myös muissa kuin nykyisessä ympäristössä.
- Luotettavuus (käyttökelpoisuuden ominaisuus): Koodi on siinä määrin luotettavaa, että sen voidaan olettaa hoitavan aiottuja toimintoja tyydyttävästi.
- Tehokkuus (käyttökelpoisuuden ominaisuus): Koodi on siinä määrin tehokasta, että se täyttää tarkoituksensa resursseja tuhlaamatta.
- Käytettävyys (käyttökelpoisuuden ominaisuus): Koodi on siinä määrin käytettävää inhimillisen suunnittelun rajoissa, että se on luotettavaa ja tehokasta.
- Testattavuus (ylläpidettävyyden ominaisuus): Koodi on siinä määrin testattavaa, että se helpottaa tarkastuskriteerien asettamista ja tukee sen suorituskyvyn arviointia.
- Ymmärrettävyys (ylläpidettävyyden ominaisuus): Koodi on siinä määrin ymmärrettävää, että sen tarkoitus on tarkastajalle selvä.
- Joustavuus (ylläpidettävyyden ominaisuus): Koodi on siinä määrin muokattavaa, että se helpottaa muutosten sisällyttämistä, kun haluttu muutos on määritelty.

Ominaisuushierarkian matalin tasorakenne Boehmin mallissa on primitiivinen ominaisuustietojen hierarkia. Alkeelliset ominaisuudet antavat perustan laatumittareiden määrittelemiselle, mikä oli yksi tavoitteista, kun Boehm rakensi laatumallinsa. Näin ollen malli esittää yhden tai useamman muuttujan, joka oletettavasti mittaa tiettyä primitiivistä ominaisuutta. Käyttökelpoisuus, ylläpidettävyys ja siirrettävyys ovat välttämättömiä ehtoja yleiselle käyttökelpoisuudelle. Käyttökelpoisuus vaatii ohjelman olevan luotettava, riittävän tehokas ja käytettävä. Ylläpidettävyys edellyttää, että käyttäjä kykenee ymmärtämään, muokkaamaan ja testaamaan ohjelmaa inhimillisen suunnittelun asettamissa rajoissa. (Kuva 7.) [4].



Kuva 7. Boehmin mallin laatuominaisuuksien puu [6].

Vaikka Boehmin ja McCallin mallit näyttävät hyvin samankaltaisilta, McCallin malli keskittyy ensisijaisesti korkean tason käyttökelpoisuus ominaisuuksien (As-is utility, kts. Kuva 7.) tarkkaan mittaamiseen, kun taas Boehmin laatutilamalli perustuu laajempaan valikoimaan ominaisuuksia, keskittyen laajemmin ja yksityiskohtaisesti ensisijaisesti ylläpidettävyyteen.

3.2.3 ISO-standardi ISO/IEC 25010:2011

ISO-standardilla ISO/IEC 25010:2011 [7] on kaksi käyttöaluetta:

- Käytön laatumalli, joka koostuu viidestä ominaisuudesta (joista osa on edelleen jaettu aliominaisuuksiin), jotka liittyvät vuorovaikutukseen, kun tuotetta käytetään tietyssä käyttöympäristössä. Tätä järjestelmämallia voidaan soveltaa kaikkiin tietojärjestelmiin, mukaan lukien sekä käytössä olevat tietokonejärjestelmät että käytössä olevat ohjelmistot.
- Tuotteen laatumalli, joka koostuu kahdeksasta ominaisuudesta (jotka jaetaan edelleen aliominaisuuksiin), jotka liittyvät ohjelmiston staattisiin ominaisuuksiin ja tietojärjestelmän dynaamisiin ominaisuuksiin. Malli soveltuu sekä tietokonejärjestelmiin että ohjelmistotuotteisiin.

Käytön laatumalli (Kuva 8.):

- Vaikuttavuus (effectiveness): Tarkkuus ja valmius, jolla käyttäjät saavuttavat määritellyt tavoitteet.
- Tehokkuus (efficiency): Kulutetut resurssit suhteessa tarkkuuteen ja valmiuteen, jolla käyttäjät saavuttavat tavoitteet. Resurssit voivat sisältää tehtävän suorittamiseen kuluvan ajan (henkilöresurssit), materiaalit tai käytön taloudelliset kustannukset.
- Tyytyväisyys (satisfaction): Missä määrin käyttäjien tarpeet tyydytetään, kun tuotetta tai järjestelmää käytetään tietyssä käyttöympäristössä. Käyttäjälle, joka ei ole suorassa vuorovaikutuksessa tuotteen tai järjestelmän kanssa, vain tarkoituksenmukaisuus ja luottamus ovat merkityksellisiä. Tyytyväisyys on käyttäjän reaktio vuorovaikutukseen tuotteen tai järjestelmän kanssa, ja sisältää ennakkoasenteet tuotteen käyttöä kohtaan.
    - Käyttökelpoisuus (usefulness): Missä määrin käyttäjä on tyytyväinen havaittuun käytännön tavoitteiden saavuttamiseen, mukaan lukien käytön tulokset ja käytön seuraukset.
    - Luottamus (trust): Missä määrin käyttäjä tai muu sidosryhmä luottaa siihen, että tuote tai järjestelmä toimii tarkoitetulla tavalla.
    - Nautinto (pleasure): Missä määrin käyttäjä saa nautintoa täyttämällä henkilökohtaiset tarpeet. Henkilökohtaisiin tarpeisiin voi kuulua tarve

hankkia uutta tietoa ja uusia taitoja, välittää henkilökohtaista identiteettiä ja herättää miellyttäviä muistoja.
    - Mukavuus (comfort): Missä määrin käyttäjä on tyytyväinen fyysiseen mukavuuteen.
- Vapaus riskeistä (freedom from risk): Missä määrin tuote tai järjestelmä vähentää potentiaalista riskiä taloudelliselle asemalle, ihmisen elämälle, terveydelle tai ympäristölle. Riski riippuu tietyn uhan esiintymisen todennäköisyydestä ja sen mahdollisista haitallisista seurauksista.
    - Taloudellisten riskien vähentäminen (economic risk mitigation): Missä määrin tuote tai järjestelmä vähentää potentiaalista riskiä taloudelliselle asemalle, tehokkaalle toiminnalle, kaupalliselle omaisuudelle, maineelle tai muille resursseille suunnitelluissa käyttöolosuhteissa.
    - Terveys- ja turvallisuusriskien vähentäminen (health and safety risk mitigation): Missä määrin tuote tai järjestelmä vähentää potentiaalista riskiä ihmisille aiotussa käyttöyhteydessä.
    - Ympäristöriskien vähentäminen (environmental risk mitigation): Missä määrin tuote tai järjestelmä vähentää omaisuudelle tai ympäristölle mahdollisesti aiheutuvaa riskiä suunnitelluissa käyttöolosuhteissa.
- Kontekstin kattavuus (context coverage): Missä määrin tuotetta tai järjestelmää voidaan käyttää vaikuttavasti, tehokkaasti, riskivapaasti ja tyytyväisesti sekä määritellyissä käyttöolosuhteissa että sellaisissa yhteyksissä, jotka ovat alun perin nimenomaisesti määriteltyjä. Käyttökontekstilla on merkitystä sekä käytön laadulle että joillekin tuotteen laatuominaisuuksille.
    - Kontekstin valmius (context completeness): Missä määrin tuotetta tai järjestelmää voidaan käyttää vaikuttavasti, tehokkaasti, riskivapaasti ja tyytyväisesti kaikissa määritellyissä käyttöympäristöissä. Kontekstin valmius voidaan määritellä tai mitata joko määränä, jolla tietyt käyttäjät voivat käyttää tuotetta saavuttaakseen määritellyt tavoitteet vaikuttavasti, tehokkaasti, riskivapaasti ja tyytyväisesti kaikissa aiotuissa käyttöolosuhteissa, tai tuotteen ominaisuuksien läsnäolona, jotka tukevat käyttöä kaikissa käyttötarkoituksissa. Esimerkiksi missä määrin ohjelmistoa voidaan käyttää pienellä näytöllä, hitaalla verkkoyhteydellä, ei-asiantuntijakäyttäjän toimesta vikasietotilassa (esim. ei verkkoyhteyttä).

- Joustavuus (flexibility): Missä määrin tuotetta tai järjestelmää voidaan käyttää vaikuttavasti, tehokkaasti, riskivapaasti ja tyytyväisesti vaatimuksissa alun perin määritellyistä olosuhteista poikkeavassa ympäristössä. Joustavuus voidaan saavuttaa mukauttamalla tuote uusille käyttäjäryhmille, tehtäville ja kulttuureille. Joustavuuden ansiosta tuotteet voivat ottaa huomioon olosuhteet, mahdollisuudet ja yksilölliset mieltymykset, joita ei ollut ennakoitu etukäteen. Jos tuotetta ei ole suunniteltu joustavuuteen, ei ehkä ole turvallista käyttää tuotetta tarkoituksenmukaisuudesta poikkeavissa yhteyksissä. Joustavuus voidaan mitata joko määränä, jolla muun tyyppiset käyttäjät voivat käyttää tuotetta saavuttaakseen erityyppisiä tavoitteita, joiden vaikuttavuus, tehokkuus, riskivapaus ja tyytyväisyys toteutuvat erityyppisissä käyttöympäristöissä, tai kyvyksi olla muokattavissa tukemaan uudentyyppisiä käyttäjiä, tehtäviä ja ympäristöjä sekä yksilöllistämistä.

Vaikuttavuus
Tehokkuus
Tyytyväisyys
Käyttökelpoisuus
Luottamus
Nautinto
Mukavuus
Vapaus riskeistä
Taloudellisten riskien vähentäminen
Terveys- ja turvallisuusriskien vähentäminen
Ympäristöriskien vähentäminen
Kontekstin kattavuus
Kontekstin valmius
Joustavuus

Kuva 8. Käytön laatumalli [7].

Tuotteen laatumalli (Kuva 9.):

- Toiminnallinen soveltuvuus (functional suitability): Missä määrin tuote tai järjestelmä tarjoaa toimintoja, jotka täyttävät ilmoitetut ja oletetut tarpeet, kun niitä

käytetään tietyissä olosuhteissa. Toiminnallinen soveltuvuus koskee vain sitä, täyttävätkö toiminnot ilmoitetut ja oletetut tarpeet, ei toiminnallista spesifikaatiota.

  - Toiminnallinen valmius (functional completeness): Missä määrin toimintojoukko kattaa kaikki määritellyt tehtävät ja käyttäjän tavoitteet.
  - Toiminnallinen oikeellisuus (functional correctness): Missä määrin tuote tai järjestelmä tuottaa oikeat tulokset tarvittavalla tarkkuudella.
  - Toiminnallinen tarkoituksenmukaisuus (functional appropriateness): Missä määrin toiminnot helpottavat tiettyjen tehtävien ja tavoitteiden saavuttamista. Käyttäjälle esitetään vain tarvittavat vaiheet tehtävän suorittamiseen, ei tarpeettomia vaiheita.

- Luotettavuus (reliability): Missä määrin järjestelmä, tuote tai komponentti suorittaa määriteltyjä toimintoja tietyissä olosuhteissa tietyn ajanjakson ajan. Ohjelmistoissa ei tapahdu kulumista. Luotettavuusrajoitukset johtuvat vaatimusten, suunnittelun ja toteutuksen vioista tai asiayhteyteen tehdyistä muutoksista. Luotettavuus pitää sisällään saatavuuden ja sen luontaiset tai ulkoisesti vaikuttavat tekijät, kuten saatavuus, luotettavuus (mukaan lukien vikasietoisuus ja palautettavuus), turvallisuus (mukaan lukien luottamuksellisuus ja eheys), ylläpidettävyys, kestävyys ja huoltotuki.
  - Kypsyys (maturity): Missä määrin järjestelmä, tuote tai komponentti täyttää luotettavuuden vaatimukset normaalikäytössä. Kypsyyden käsitettä voidaan soveltaa myös muihin laatuominaisuuksiin osoittamaan, missä määrin ne täyttävät vaaditut tarpeet normaalissa käytössä.
  - Saatavuus (availability): Missä määrin järjestelmä, tuote tai komponentti on toiminnassa ja käytettävissä, kun sitä tarvitaan käyttöön. Ulkopuolisesti saatavuutta voidaan arvioida kokonaisajan osuudella, jonka aikana järjestelmä, tuote tai komponentti on toiminnassa. Saatavuus on siis yhdistelmä kypsyyttä (joka säätelee vikatiheyttä), vikasietoisuutta ja palautettavuutta (joka säätelee jokaisen vian jälkeistä seisokkiaikaa).
  - Vikasietoisuus (fault tolerance): Missä määrin järjestelmä, tuote tai komponentti toimii tarkoitetulla tavalla laitteisto- tai ohjelmistovioista huolimatta.
  - Palautettavuus (recoverability): Missä määrin tuote tai järjestelmä voi keskeytyksen tai vian sattuessa palauttaa suoraan vaikuttavat tiedot ja

järjestelmän halutun tilan. Vian seurauksena tietojärjestelmä on joskus poissa käytöstä jonkin aikaa, jonka pituuden määrää sen palautettavuus.

- Käytettävyys (usability): Missä määrin tietyt käyttäjät voivat käyttää tuotetta tai järjestelmää tiettyjen tavoitteiden saavuttamiseksi vaikuttavasti, tehokkaasti ja tyytyväisyydellä tietyssä käyttöyhteydessä. Käyttökelpoisuus voidaan joko määritellä tuotteen laatuominaisuudeksi sen aliominaisuuksien perusteella tai määritellä ja mitata suoraan toimenpiteillä, jotka ovat käytön laadun osajoukko.
    - Tarkoituksenmukaisuuden tunnistettavuus (appropriateness recognizability): Missä määrin käyttäjät voivat tunnistaa, onko tuote tai järjestelmä sopiva heidän tarpeisiinsa. Tarkoituksenmukaisuuden tunnistettavuus riippuu kyvystä tunnistaa tuotteen tai järjestelmän toimintojen tarkoituksenmukaisuus tuotteen tai järjestelmän alkukuvausten ja/tai minkä tahansa siihen liittyvän dokumentaation perusteella. Tuotteen tai järjestelmän tarjoamat tiedot voivat sisältää esittelyjä, opetusohjelmia, dokumentaatiota tai verkkosivustoa varten kotisivulta löytyviä tietoja.
    - Opittavuus (learnability): Missä määrin käyttäjät voivat käyttää tuotetta tai järjestelmää saavuttaakseen määritellyt tavoitteet oppia käyttämään tuotetta tai järjestelmää vaikuttavasti, tehokkaasti, vailla riskiä ja tyytyväisyydellä tietyssä käyttöympäristössä. Voidaan määrittää tai mitata joko siten, kuinka käyttäjät voivat käyttää tuotetta tai järjestelmää saavuttaakseen määritellyt tavoitteet oppia käyttämään tuotetta tai järjestelmää vaikuttavasti, tehokkaasti, vailla riskiä ja tyytyväisyydellä tietyssä käyttöympäristössä, tai tuotteen ominaisuuksien perusteella.
    - Toimivuus (operability): Missä määrin tuotteella tai järjestelmällä on ominaisuuksia, jotka helpottavat sen käyttöä ja hallintaa. Toimivuus tarkoittaa hallittavuutta, (käyttäjän) virhetoleranssia ja käyttäjän odotusten noudattamista.
    - Käyttäjän virhesuojaus (user error protection): Missä määrin järjestelmä suojaa käyttäjiä virheiltä.
    - Käyttöliittymän estetiikka (user interface aesthetics): Missä määrin käyttöliittymä mahdollistaa miellyttävän ja tyydyttävän vuorovaikutuksen käyttäjälle. Tämä viittaa tuotteen tai järjestelmän ominaisuuksiin, jotka lisäävät käyttäjän mielihyvää ja tyytyväisyyttä, kuten värien käyttö ja graafisen suunnittelun luonne.

    - Saavutettavuus (accessibility): Missä määrin tuotetta tai järjestelmää voivat käyttää ihmiset, joilla on laajimmat ominaisuudet ja kyvyt saavuttaa määritelty tavoite tietyssä käyttöympäristössä. Ominaisuuksien valikoima sisältää iän mukanaan tuomat toimintakyvyn alenemat. Vammaisten esteettömyys voidaan määritellä tai mitata joko siten, missä määrin vammaiset käyttäjät voivat käyttää tuotetta tai järjestelmää saavuttaakseen määritellyt tavoitteet vaikuttavasti, tehokkaasti, vailla riskiä ja tyytyväisyydellä tietyssä käyttöyhteydessä tai tukevien tuoteominaisuuksien avulla.
- Suorituskyky (performance efficiency): Suorituskyky suhteessa käytettyjen resurssien määrään määritellyissä olosuhteissa. Resurssit voivat sisältää muita ohjelmistotuotteita, järjestelmän ohjelmisto- ja laitteistokokoonpanoja sekä materiaaleja (esim. painopaperi, tallennusväline).
    - Suoritusaika (time behaviour): Missä määrin tuotteen tai järjestelmän vaste- ja prosessointiajat sekä läpimenonopeudet täyttävät vaatimukset.
    - Resurssien käyttö (resource utilization): Missä määrin tuotteen tai järjestelmän käyttämät resurssien määrät ja tyypit täyttävät vaatimukset. Henkilöresurssit sisältyvät käytön laatumallin tehokkuuteen.
    - Kapasiteetti (capacity): Missä määrin tuotteen tai järjestelmäparametrin enimmäisrajat täyttävät vaatimukset. Parametrit voivat sisältää tallennettavien kohteiden lukumäärän, samanaikaisten käyttäjien määrän, viestinnän kaistanleveyden, tapahtumien läpimenon ja tietokannan koon.
- Tietoturva (security): Missä määrin tuote tai järjestelmä suojaa tietoja siten, että henkilöillä tai muilla tuotteilla tai järjestelmillä on pääsy tietoihin heidän tyyppinsä ja lupatasonsa mukaisesti. Tuotteeseen tai järjestelmään tallennettujen tietojen lisäksi tietoturva koskee myös lähetettäviä tietoja. Selviytyvyys (kuinka hyvin tuote tai järjestelmä jatkaa tehtävänsä suorittamista tarjoamalla olennaisia palveluja ajallaan hyökkäyksistä huolimatta) sisältyy palautettavuuteen. Immuniteetti (missä määrin tuote tai järjestelmä kestää hyökkäyksiä) kuuluu eheyden piiriin.
    - Luottamuksellisuus (confidentiality): Missä määrin tuote tai järjestelmä varmistaa, että tietoihin pääsevät vain ne, joilla on siihen lupa.
    - Eheys (integrity): Missä määrin järjestelmä, tuote tai komponentti estää tietokoneohjelmien tai tietojen luvattoman käytön tai muokkaamisen.

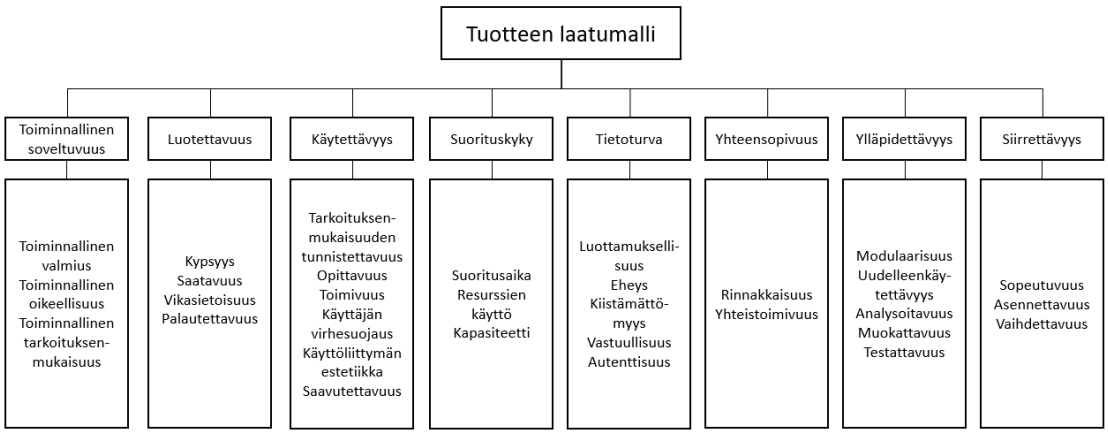
    - Kiistämättömyys (non-repudiation): Missä määrin toimet tai tapahtumat voidaan todistaa tapahtuneiksi, jotta tapahtumia tai toimia ei voida kumota myöhemmin.
    - Vastuullisuus (accountability): Missä määrin entiteetin toimet voidaan jäljittää.
    - Autenttisuus (authenticity): Missä määrin kohde tai resurssi voidaan tunnistaa.
- Yhteensopivuus (compatibility): Missä määrin tuote, järjestelmä tai komponentti voi vaihtaa tietoja muiden tuotteiden, järjestelmien tai komponenttien kanssa ja/tai suorittaa vaaditut toiminnot samalla laitteisto- tai ohjelmistoympäristöllä.
    - Rinnakkaisuus (co-existence): Missä määrin tuote voi suorittaa vaaditut toiminnot tehokkaasti jakamalla yhteisen ympäristön ja resurssit muiden tuotteiden kanssa ilman haitallisia vaikutuksia muihin tuotteisiin.
    - Yhteistoimivuus (interoperability): Missä määrin kaksi tai useampi järjestelmä, tuote tai komponentti voi vaihtaa tietoja ja käyttää vaihdettua tietoa.
- Ylläpidettävyys (maintainability): Vaikuttavuus ja tehokkuus, jolla ylläpitäjä voi muokata tuotetta tai järjestelmää. Muutokset voivat sisältää korjauksia, parannuksia tai ohjelmiston mukauttamista ympäristön, vaatimusten ja toiminnallisten edellytysten muutoksiin. Muutokset sisältävät tukihenkilöstön tekemät muutokset sekä liike- tai operatiivisen henkilöstön tai loppukäyttäjien tekemät muutokset. Ylläpidettävyys sisältää päivitysten asentamisen. Ylläpidettävyys voidaan tulkita joko tuotteelle tai järjestelmälle ominaiseksi kyvyksi kunnossapitotoimien helpottamiseen tai laaduksi, jonka ylläpitäjät kokevat tuotetta tai järjestelmää ylläpitäessään.
    - Modulaarisuus (modularity): Missä määrin järjestelmä tai tietokoneohjelma koostuu erillisistä komponenteista siten, että yhden komponentin muutoksella on vähäinen vaikutus muihin komponentteihin.
    - Uudelleenkäytettävyys (reusability): Missä määrin koodia voidaan käyttää useammassa kuin yhdessä järjestelmässä tai muiden omaisuuksien rakentamiseen.
    - Analysoitavuus (analysability): Vaikuttavuuden ja tehokkuuden taso, jolla on mahdollista arvioida yhden tai useamman järjestelmän osan suunnitellun muutoksen vaikutusta tuotteeseen tai järjestelmään tai diagnosoida tuotteessa puutteita tai vikojen syitä tai tunnistaa muutettavat

osat. Toteutus voi sisältää mekanismien tarjoamisen tuotteelle tai järjestelmälle omien vikojensa analysoimiseksi ja raporttien antamiseksi ennen vikaa tai muuta tapahtumaa.

  - Muokattavuus (modifiability): Missä määrin tuotetta tai järjestelmää voidaan muokata vaikuttavasti ja tehokkaasti aiheuttamatta vikoja tai heikentämättä tuotteen nykyistä laatua. Toteutus sisältää koodauksen, suunnittelun, dokumentoinnin ja muutosten tarkistamisen. Modulaarisuus ja analysoitavuus voivat vaikuttaa muokattavuuteen. Muokattavuus on yhdistelmä vaihdettavuutta ja vakautta.
  - Testattavuus (testability): Vaikuttavuus ja tehokkuus, jolla testikriteerit voidaan vahvistaa järjestelmälle, tuotteelle tai komponentille, ja testit voidaan suorittaa sen määrittämiseksi, täyttyvätkö nämä kriteerit.

- Siirrettävyys (portability): Vaikuttavuuden ja tehokkuuden taso, jolla järjestelmä, tuote tai komponentti voidaan siirtää laitteistosta, ohjelmistosta tai muusta käyttöympäristöstä toiseen. Siirrettävyys voidaan tulkita tuotteen tai järjestelmän luontaiseksi kyvyksi siirtotoiminnan helpottamiseen.
  - Sopeutuvuus (adaptability): Missä määrin tuote tai järjestelmä voidaan vaikuttavasti ja tehokkaasti mukauttaa erilaisille tai kehittyville laitteistoille, ohjelmistoille tai muille käyttöympäristöille. Sopeutuvuus sisältää sisäisen kapasiteetin skaalautuvuuden (esim. näyttökentät, taulukot, tapahtumamäärät, raporttimuodot jne.). Mukautukset sisältävät tukihenkilöstön tekemät sekä liike- tai operatiivisen henkilöstön tai loppukäyttäjien tekemät mukautukset. Jos loppukäyttäjän on mukautettava järjestelmää, sopeutuvuus vastaa soveltuvuutta yksilöllistämiseen.
  - Asennettavuus (installability): Vaikuttavuuden ja tehokkuuden taso, jolla tuote tai järjestelmä voidaan asentaa ja/tai poistaa onnistuneesti tietyssä ympäristössä. Jos tuotteen tai järjestelmän tulee asentamaan loppukäyttäjä, asennettavuus voi vaikuttaa toiminnalliseen tarkoituksenmukaisuuteen ja käytettävyyteen.
  - Vaihdettavuus (replaceability): Missä määrin tuote voi korvata toisen samaan tarkoitukseen määritetyn ohjelmistotuotteen samassa ympäristössä. Ohjelmistotuotteen uuden version vaihdettavuus on tärkeää käyttäjälle päivitettäessä ohjelmistoa. Vaihdettavuus voi sisältää sekä asennettavuuden että sopeutuvuuden ominaisuuksia. Käsite on

otettu käyttöön aliominaisuutena sen tärkeyden vuoksi. Vaihdettavuus vähentää riippuvuutta: muita ohjelmistotuotteita voidaan käyttää nykyisen sijasta, esimerkiksi käyttämällä standardoituja tiedostomuotoja.



Kuva 9. Standardin ISO/IEC 25010:2011 mukainen lajittelu tuotteen laatumaalista [8].

3.3 Palvelinarkkitehtuurin laatuattribuutit

Kiinteä ohjelmiston laatumalli on usein hyödyllinen, kun otetaan huomioon yleinen käsitys ohjelmiston laadusta. Käytännössä laatuattribuuttien suhteellinen merkitys riippuu tyypillisesti ohjelmistoalueesta, tuotetyypistä ja aiotusta käytöstä. Siksi ohjelmiston ominaisuudet olisi määriteltävä ja niiden ohjattava kunkin tuotteen kehitystä. Näin ollen palvelinarkkitehtuuria suunniteltaessa edellä mainittua listaa laatuattribuuteista on syytä soveltaa vain niiltä osin, kun ne palvelevat tulevaa käyttötarkoitusta.

Palvelinarkkitehtuurin laatumallia ei ole tutkittu yhtä laajasti kuin ohjelmistojen laatumalleja. Näin ollen on luontevaan käyttää ohjelmistojen laatumalleja ja niistä tehtyä tutkimusta pohjana palvelinarkkitehtuurin laatumallia pohdittaessa. Yleisesti ohjelmistoalan tuotteiden laatua tutkivat laatumallit toimivat hyvänä pohjana palvelinarkkitehtuurin laatumallin pohtimiseen niiden sisällöllisen ja kontekstuaalisen laajuuden vuoksi.

Olen jakanut palvelinarkkitehtuuria koskevat laatuvaatimukset seuraaviin osa-alueisiin: tietoturva, saatavuus, vikasietoisuus, skaalautuvuus ja ylläpidettävyys. Nämä viisi kategoriaa kattavat käsityksen siitä, mitä palvelinratkaisulta vaaditaan, jotta se toimisi

oikein ja luotettavasti kaikissa tilanteissa. Keskiössä on loppukäyttäjän kokema käytettävyys, palvelun pitää olla loppukäyttäjän käytettävissä tilanteesta riippumatta ja palvelinarkkitehtuurin on tuettava tätä. Lisäksi ylläpidettävyyden näkökulma on tärkeä ottaa huomioon ohjelmistokehittäjien vuoksi.

Osin nämä viisi kategoriaa ovat muodostuneet myös työkokemukseni perusteella palvelinratkaisujen suunnittelun saralta, osittain nykyisen työnantajani, LogiSystems Oyn, vaatimuksista yrityksen palvelinarkkitehtuurin suhteen. Kategorioiden valintaa perustellaan tarkemmin seuraavissa alaluvuissa.

3.3.1 Tietoturva

Tietoturva on osa-alue, joka on keskiössä kaikessa ohjelmistoalaan liittyvässä, ja jonka merkitys on korostunut viimeisten vuosien aikana entisestään muun muassa lainsäädännön kiristyessä. McCallanin ja Boehmin laatumalleista näkee, että ne on tehty 70-luvulla ja tietoturva kysymykset eivät ole näiden laatumallien keskiössä. McCallan puhuu eheydestä omassa laatumallissaan tarkoittaen suojausta ohjelman luvattomalta käytöltä. Boehm ei mainitse tietoturvaa omassa laatumallissaan lainkaan.

ISO-standardi ISO/IEC 25010:2011 ottaa laajemmin kantaa aiheeseen. Standardissa mainitaan myös eheys, niin kuin McCallaninkin mallissa, mutta sen rinnalle on nostettu myös luottamuksellisuus, kiistämättömyys, vastuullisuus ja autenttisuus. Standardikaan ei anna tietoturvalle suurta painoarvoa. Nykyään kuitenkin ohjelmistoissa ja digitaalisissa palveluissa on siirrytty entistä enemmän SaaS-malliin, jolloin palvelinten merkitys korostuu ja asiakaspuolen ohjelmien merkitys on kutistunut pelkäksi käyttöliittymäksi. Näin ollen SaaS-ohjelmistoratkaisut nojaavat vahvasti palvelinten päälle ja palvelinarkkitehtuurin tietoturvan painoarvo korostuu yhdessä tietoturvaa koskevan yleisen ilmapiirin muutoksen kanssa lainsäädäntöineen.

3.3.2 Saatavuus

Saatavuus on tekijä, joka on noussut jopa tärkeimmäksi attribuutiksi palvelun laatua mitattaessa. Kuten todettua, ohjelmistojen siirtyessä entistä enemmän kohti SaaS-mallia, saatavuuden merkitys korostuu offline-käytön haasteiden/vähäisyyden vuoksi. Laajasti käytössä olevan ohjelmiston pitää olla saatavilla kaikissa tilanteissa kaikilla

laitteilla, sillä usein yrityksen koko liiketoiminta saattaa pyöriä kyseisen ohjelmiston varassa ja käyttökatkokset saattavat pahimmillaan tarkoittaa liiketoiminnan täydellistä keskeytymistä käyttökatkon ajaksi. Näin ollen palvelinarkkitehtuuria suunniteltaessa saatavuuteen on kiinnitettävä erityistä huomiota.

Saatavuutta voidaan pitää kohtuullisen laajana kokonaisuutena palvelinarkkitehtuurista puhuttaessa ja sen voidaan nähdä sisältävän pelkkää palvelun ylläoloaikaa laajemman soveltamisalan, johon voidaan laskea kuuluvaksi myös loppukäyttäjän käytön kokemusta mittaavia attribuutteja.

McCallanin laatumallissa puhutaan luotettavuudesta ja sillä tarkoitetaan järjestelmän kykyä olla epäonnistumatta. Tämä ei varsinaisesti vastaa saatavuuden käsitettä, mutta on lähin vastine mitä McCallanilla on tarjota ongelmaan. McCallan mainitsee myös tehokkuuden (suoritustehokkuus ja tallennustehokkuus), mutta tämäkään ei varsinaisesti täytä saatavuusattribuutin tarpeita.

Boehmin malli ei myöskään tarjoa suoria ratkaisuja modernimpaan saatavuuden tarpeeseen. Myös Boehm puhuu luotettavuudesta (ohjelmiston voidaan olettaa hoitavan aiottuja toimintoja tyydyttävästi) ja tehokkuudesta (ohjelmisto täyttää tarkoituksensa resursseja tuhlaamatta). Lisäksi Boehm mainitsee käytettävyyden inhimillisen suunnittelun rajoissa, niin että ratkaisu on luotettava ja tehokas.

Kumpikaan malli, McCallan taikka Boehm, ei tarjoa paljoa vastauksia etsittäessä ratkaisuja modernimpaan saatavuuden tarpeeseen. Mallien ikä ja keskittyminen erityisesti ohjelmistojen laadun mittaamiseen ei tarjoa juurikaan ratkaisuja palvelujen saatavuuden laatuattribuutteja pohdittaessa.

ISO-standardi ISO/IEC 25010:2011 puolestaan modernimpana laatumallina ottaa suoraan kantaa saavutettavuuteen luotettavuuden (reliability) alla. Standardissa saavutettavuuteen (järjestelmä on toiminnassa ja käytettävissä, kun sitä tarvitaan käyttöön) yhdistetään kypsyys, vikasietoisuus ja palautettavuus. Saavutettavuuteen voidaan yhdistää myös standardissa mainittu luottamus (trust). Jossain määrin saavutettavuuden yhteydessä voidaan käsitellä myös luottamuksen yläkategoriaa tyytyväisyyttä laajemminkin.

Korkealla saatavuudella voi olla myös riskejä vähentävä vaikutus. Korkean saatavuuden järjestelmä vähentää potentiaalista riskiä taloudelliselle asemalle, ihmisen elämälle,

terveydelle tai ympäristölle [7]. Riski riippuu tietyn uhan esiintymisen todennäköisyydestä ja sen mahdollisista haitallisista seurauksista.

Saatavuuden yhteydessä voidaan käyttää myös toiminnallisen soveltuvuuden käsitettä. Toiminnallisella soveltuvuudella mitataan järjestelmän tarjoamia toimintoja, jotka täyttävät ilmoitetut ja oletetut tarpeet, kun niitä käytetään tietyissä olosuhteissa. Erityisesti toiminnallisen soveltuvuuden alakäsitteet voivat palvella saatavuutta: toiminnallinen valmius (toimintojoukko kattaa kaikki määritellyt tehtävät ja käyttäjän tavoitteet), toiminnallinen oikeellisuus (järjestelmä tuottaa oikeat tulokset tarvittavalla tarkkuudella) ja toiminnallinen tarkoituksenmukaisuus (missä määrin toiminnot helpottavat tiettyjen tehtävien ja tavoitteiden saavuttamista).

McCallanin ja Boehmin laatumalleissa on nähtävissä ne elementit mitkä ISO standardi ISO/IEC 25010:2011 laajentaa saatavuuden kannalta paremmaksi kokonaisuudeksi. Kaikissa laatumalleissa käsitellään luotettavuutta ja tehokkuutta, standardi ottaa paremmin kantaa käyttäjäkokemukseen saatavuuden yhteydessä ja siihen liittyen.

3.3.3 Vikasietoisuus

Moderneissa palvelinratkaisuissa vikasietoisuus on tärkeä attribuutti, koska monesti palvelimet ja palvelinohjelmat toimivat itsenäisesti ilman ylläpitäjien manuaalista valvontaa automaattisen valvonnan piirissä. Näin ollen vian sattuessa häiriön vaikutus tulisi koskea mahdollisimman pientä osaa järjestelmää ja viasta pitäisi pystyä palautumaan mahdollisimman nopeasti ja mielellään automaattisesti. Tässä mielessä vikasietoisuus liittyy läheisesti saatavuuteen, vian sattuessa palvelu ei ole saatavissa. Näin ollen vikasietoisuuden voidaan ajatella vaikuttavan suoraan saatavuuteen.

McCallan mainitsee ylläpidettävyyden tarkoittaen työmäärä, joka tarvitaan ohjelman vian paikantamiseen ja korjaamiseen ohjelman toimintaympäristössä. Boehmilta esiin voidaan nostaa luotettavuus vikasietoisuuden yhteydessä, vaikka Boehmin mallista ei suoria yhtymäkohtia vikasietoisuuteen löydykään.

ISO-standardi ISO/IEC 25010:2011 tarjoaa vikasietoisuuteen ylläpidettävyyden alta löytyviä modulaarisuutta (järjestelmä koostuu erillisistä komponenteista siten, että yhden komponentin muutoksella on vähäinen vaikutus muihin komponentteihin), analysoitavuutta (mekanismi järjestelmälle omien vikojensa analysoimiseksi) ja

muokattavuutta (järjestelmää voidaan muokata aiheuttamatta vikoja tai heikentämättä tuotteen nykyistä laatua).

Standardissa mainittua yhteensopivuutta (järjestelmä tai komponentti voi vaihtaa tietoja muiden järjestelmien tai komponenttien kanssa ja/tai suorittaa vaaditut toiminnot samalla laitteisto- tai ohjelmistoympäristöllä) voidaan soveltaa sellaisenaan vikasietoisuutta arvioitaessa. Yhteensopivuus kattaa myös rinnakkaisuuden (suorittaa vaaditut toiminnot tehokkaasti jakamalla yhteisen ympäristön ja resurssit muiden komponenttien kanssa ilman haitallisia vaikutuksia muihin komponentteihin) ja yhteistoimivuuden (järjestelmä voi vaihtaa tietoja ja käyttää vaihdettua tietoa).

Standardin luotettavuuskategoria limittyy osittain saatavuuden kanssa ja monet sen kohdat sopivat myös vikasietoisuuteen. Luotettavuuden alta löytyy suoraan attribuutti vikasietoisuus (järjestelmä toimii tarkoitetulla tavalla laitteisto- tai ohjelmistovioista huolimatta). Palautettavuus (järjestelmä voi keskeytyksen tai vian sattuessa palauttaa suoraan vaikuttavat tiedot ja järjestelmän halutun tilan) on myös tärkeä osa vikasietoisuuden käsitettä. Kypsyyden ja saatavuuden käsitteitä voidaan soveltaa myös vikasietoisuuden yhteydessä rajallisesti.

Myös toiminnallisen soveltuvuuden alta löytyvää toiminnallista oikeellisuutta (järjestelmä tuottaa oikeat tulokset tarvittavalla tarkkuudella) voidaan soveltaa vikasietoisuuden yhteydessä. Kontekstin kattavuuteen liittyvää joustavuutta (järjestelmää voidaan käyttää vaatimuksissa alun perin määritellyistä olosuhteista poikkeavassa ympäristössä) voidaan käyttää vikasietoisuutta pohdittaessa, vaikka tämä kohta standardissa liittyy enemmän käytettävyyteen kuin tekniseen toteutukseen.

Jälleen on huomattavissa, että McCallanin ja Boehmin laatumallit vanhempina (ja suppeampina) eivät kykene tarjoamaan riittävästi ratkaisuja vikasietoisuuden käsitteeseen. ISO-standardi ISO/IEC 25010:2011 kykenee tarjoamaan lukuisia attribuutteja vikasietoisuudelle palvelinarkkitehtuurin laatua pohdittaessa. On kuitenkin huomioitava, että standardi pohjautuu McCallanin ja Boehmin laatumalleihin ja samat elementit toistuvat kaikissa malleissa vaikkakin hieman eri laajuudella ja eri näkökulmista.

3.3.4 Skaalautuvuus

Modernien palvelinratkaisujen pitää olla skaalautuvia. Näin ollen ruuhkahuippuja pystytään tasaamaan niin, että palvelun saavutettavuus ei kärsi. Kun käyttöä on vähän, palveluun varattuja resursseja voidaan vapauttaa muuhun käyttöön ja näin ollen ohjata resursseja niitä tarvitseville komponenteille taikka säästää kustannuksia.

Sekä McCallanin että Boehmin mallit vastaavat varsin huonosti skaalautuvuuden tarpeeseen, mikä lienee täysin ymmärrettävää ottaen huomioon mallien iän. Molemmissa malleissa esiintyvää joustavuutta ja tehokkuutta voidaan soveltaa tietyin rajoittein. Tehokkuutta voidaan soveltaa resurssien käytön kontekstissa: Palvelinten ja komponenttien tulisi toimia mahdollisimman tehokkaasti resursseja tuhlaamatta. Joustavuus voidaan mieltää mahdollisuudeksi uudelleen ohjata käytössä olevia resursseja.

McCallanin mallin kohta tuotteen siirrettävyydestä voidaan muuntaa muotoon resurssien siirrettävyys. Näin ollen siirrettävyys voidaan määritellä työmääräksi, joka tarvitaan resurssien siirtämiseen yhdestä ympäristöstä toiseen. Yhteentoimivuus puolestaan tarvittavaksi työmääräksi resurssien siirtämiseksi toiseen järjestelmään. Boehmin mallissa käsitellään pitkälti samoja käsitteitä ja niistä on johdettavissa vastaavia määritelmiä.

ISO-standardi ISO/IEC 25010:2011 tarjoaa attribuutteja skaalautuvuuden määrittelemiseksi, joskin attribuutteja on mukautettava ja laajennettava vastaamaan uutta sovellusalaa. Tämä on sinänsä ymmärrettävää, onhan kyseinen standardi ajalta ennen SaaS-ohjelmistoratkaisujen yleistymistä ja käsittelee nimenomaan ohjelmistoja, ei palvelinarkkitehtuuria.

Joustavuuden käsitettä voidaan laajentaa kattamaan resurssien joustavan käsittelyn. Toiminnallinen soveltuvuus voidaan uudelleenkirjoittaa muotoon missä määrin järjestelmä tarjoaa toimintoja, jotka täyttävät komponenttien ilmoitetut ja oletetut resurssien tarpeet, kun niitä käytetään tietyissä olosuhteissa.

Skaalautuvuuteen liittyy myös luotettavuusnäkökulma. Toimiakseen luotettavasti järjestelmän on pystyttävä allokoimaan resursseja palvelinten ja komponenttien välillä. Toisin sanoen luotettavuudessa on kyse siitä, missä määrin järjestelmä pystyy suorittamaan määriteltyjä toimintoja tietyissä olosuhteissa tietyn ajanjakson ajan.

Luotettavuusrajoitukset johtuvat vaatimusten, suunnittelun ja toteutuksen vioista tai asiayhteyteen tehdyistä muutoksista. Näin ollen skaalautuvuus linkittyy saatavuuden ja vikasietoisuuden kanssa.

Skaalautuvuus liittyy suoraan suorituskykyyn. Standardissa suorituskyky määritellään seuraavasti: Suorituskyky suhteessa käytettyjen resurssien määrään määritellyissä olosuhteissa. Suorituskykyyn vaikuttavat suoritusaika (vaste- ja prosessointiajat sekä läpimenonopeudet), resurssien käyttö (resurssien määrät ja tyypit) ja kapasiteetti (järjestelmäparametrien enimmäisrajat). Suorituskyky vaikuttaa suoraan saatavuuteen.

Standardin mukainen siirrettävyys voidaan laajentaa kattamaan resurssien siirrettävyys, samaan tapaan kuin McCallaninkin mallin kohdalla. Näin ollen siirrettävyys voidaan mieltää vaikuttavuuden ja tehokkuuden tasoksi, jolla resursseja voidaan siirtää käyttöympäristöstä toiseen. Siirrettävyys voidaan tulkita järjestelmän luontaiseksi kyvyksi siirtotoiminnan helpottamiseen. Siirrettävyyteen voidaan liittää sopeutuvuuden (kapasiteetin skaalautuvuus) käsite.

3.3.5 Ylläpidettävyys

Edelliset neljä laatukokonaisuutta liittyvät lähtökohtaisesti käyttäjän kokemaan käytettävyyteen, ylläpidettävyys on kehittäjien ja ylläpitäjien kannalta keskeisessä asemassa. Siinä missä perinteistä ohjelmistoa ei välttämättä tarvitse ylläpitää tai jatkokehittää, on SaaS-palvelut ja palvelinkokonaisuudet usein jatkuvassa muutoksen tilassa. Tällöin ylläpidettävyyden merkitys korostuu, henkilöstöresursseja ei useinkaan ole tuhlattavaksi asti. Toki on huomioitava, että ylläpidettävyys on ollut aina keskeisessä osassa tietoteknisiä ratkaisuja, mikä näkyy myös McCallanin ja Boehmin kaltaisissa, historiallisissa ohjelmiston laatumalleissa.

McCallanin laatumalli tarjoaa hyvin näkökulmia ylläpidettävyyteen. Ensinnäkin laatumalli käsittelee suoraan ylläpidettävyyttä (työmäärä, joka tarvitaan vian paikantamiseen ja korjaamiseen), mutta myös ylläpidettävyyteen liittyviä attribuutteja kuten joustavuus (toimintaympäristön muutosten edellyttämien muutosten tekemisen helppous), testattavuus, siirrettävyys, uudelleenkäytettävyys sekä yhteentoimivuus (tarvittava työmäärä järjestelmän yhdistämiseksi toiseen järjestelmään).

Myös Boehmilla on tarjota paljon ylläpidettävyydelle. Ensinnäkin yksi Boehmin laatumallin kolmesta korkean tason ominaisuudesta on ylläpidettävyys (kuinka helppoa

on ymmärtää, muokata ja testata). Lisäksi keskitason ominaisuuksista siirrettävyys (voidaan käyttää myös muissa kuin nykyisessä ympäristössä), testattavuus (helpottaa tarkastuskriteerien asettamista ja tukee suorituskyvyn arviointia), ymmärrettävyys ja joustavuus (helpottaa muutosten sisällyttämistä, kun haluttu muutos on määritelty) voidaan katsoa liittyvän ylläpidettävyyteen.

Myös ISO-standardi ISO/IEC 25010:2011 käsittelee ylläpidettävyyttä omana kategorianaan. Standardissa ylläpidettävyys määritellään seuraavasti: Vaikuttavuus ja tehokkuus, jolla ylläpitäjä voi muokata järjestelmää. Muutokset voivat sisältää korjauksia, parannuksia tai järjestelmän mukauttamista ympäristön, vaatimusten ja toiminnallisten edellytysten muutoksiin. Ylläpidettävyys sisältää päivitysten asentamisen. Ylläpidettävyys voidaan tulkita joko järjestelmälle ominaiseksi kyvyksi kunnossapitotoimien helpottamiseen tai laaduksi, jonka ylläpitäjät kokevat tuotetta tai järjestelmää ylläpitäessään.

Ylläpidettävyyden attribuutteja ovat modulaarisuus (yhden komponentin muutoksella on vähäinen vaikutus muihin komponentteihin), uudelleenkäytettävyys, analysoitavuus (yhden tai useamman järjestelmän osan muutoksen vaikutusta järjestelmään, järjestelmän puutteiden tai vikojen syiden diagnosointi, muutettavien osien tunnistus), muokattavuus (järjestelmän muokkaaminen aiheuttamatta vikoja tai heikentämättä nykyistä laatua) ja testattavuus.

Ylläpidettävyyteen voidaan liittää myös siirrettävyyden käsite. Siirrettävyys voidaan tulkita järjestelmän luontaiseksi kyvyksi siirtotoiminnan helpottamiseen. Siirrettävyyteen kuuluu sopeutuvuus (järjestelmä voidaan mukauttaa erilaisille tai kehittyville käyttöympäristöille), asennettavuus ja vaihdettavuus (toisen samaan tarkoitukseen määritetyn komponentin korvaaminen).

Myös vapautta riskeistä voidaan käyttää ylläpidettävyyden yhteydessä puhtaasti operatiivisesta näkökulmasta käsittäen korkean ylläpidettävyyden riskejä vähentävää luonnetta. Operatiivista riskiä liittyen taloudellisiin sekä turvallisuusriskeihin on mahdollista vähentää ylläpidettävyyden avulla. Itse palvelinratkaisun riskejä vähentävä vaikutus loppukäyttäjälle kuuluu saatavuuden alle.

Lisäksi ylläpidettävyyden yhteydessä voidaan käsitellä tyytyväisyyttä. On tärkeää ylläpidettävyyden kannalta, että ylläpitäjät ovat tyytyväisiä järjestelmän ylläpidettävyyteen. Standardin tyytyväisyys liittyy ohjelmiston loppukäyttäjän kokemaan tyytyväisyyteen, mutta tässä yhteydessä se voidaan muuttaa käsittelemään

palvelinkokonaisuuden ylläpitäjien kokemaa tyytyväisyyttä palvelinkokonaisuuden ylläpidettävyyteen. Tyytyväisyyteen liittyviä attribuutteja ovat käyttökelpoisuus (missä määrin ylläpitäjä on tyytyväinen havaittuun käytännön tavoitteiden saavuttamiseen) ja luottamus (missä määrin ylläpitäjä luottaa siihen, että järjestelmä toimii tarkoitetulla tavalla).

3.4 Palvelinarkkitehtuurin laatumalli

Tiivistelmä kehitetystä palvelinarkkitehtuurin laatumallista. Laatumalli perustuu alaluvun 2.3 pohdintaan.

- Tietoturva: Järjestelmä suojaa tietoja siten, että henkilöillä tai muilla tuotteilla tai järjestelmillä on pääsy tietoihin heidän tyyppinsä ja lupatasonsa mukaisesti. Järjestelmään tallennettujen tietojen lisäksi tietoturva koskee myös lähetettäviä tietoja.
    - Eheys: Järjestelmä estää tietokoneohjelmien tai tietojen luvattoman käytön tai muokkaamisen.
    - Luottamuksellisuus: Järjestelmä varmistaa, että tietoihin pääsevät vain ne, joilla on siihen lupa.
    - Kiistämättömyys: Toimet tai tapahtumat voidaan todistaa tapahtuneiksi.
    - Vastuullisuus: Entiteetin toimet voidaan jäljittää.
    - Autenttisuus: Kohde tai resurssi voidaan tunnistaa.
- Saatavuus: Järjestelmä on toiminnassa ja käytettävissä, kun sitä tarvitaan käyttöön.
    - Luotettavuus: Järjestelmän kyky olla epäonnistumatta. Voidaan olettaa hoitavan aiottuja toimintoja tyydyttävästi.
    - Tehokkuus: Täyttää tarkoituksensa resursseja tuhlaamatta.
        - Suoritustehokkuus
        - Tallennustehokkuus.
    - Kypsyys: Järjestelmä täyttää luotettavuuden vaatimukset normaalikäytössä.
    - Vikasietoisuus: Järjestelmä toimii tarkoitetulla tavalla laitteisto- tai ohjelmistovioista huolimatta. Yhteydessä vikasietoisuuden pääkategoriaan.

    - Palautettavuus: Järjestelmä voi keskeytyksen tai vian sattuessa palauttaa suoraan vaikuttavat tiedot ja järjestelmän halutun tilan. Yhteydessä vikasietoisuuden pääkategoriaan.
- Vikasietoisuus: Järjestelmä toimii tarkoitetulla tavalla laitteisto- tai ohjelmistovioista huolimatta. Yhteydessä saatavuuden pääkategoriaan.
    - Modulaarisuus: Järjestelmä koostuu erillisistä komponenteista siten, että yhden komponentin muutoksella on vähäinen vaikutus muihin komponentteihin.
    - Analysoitavuus: Mekanismi järjestelmälle omien vikojensa analysoimiseksi.
    - Muokattavuus: Järjestelmää voidaan muokata aiheuttamatta vikoja tai heikentämättä tuotteen nykyistä laatua.
    - Yhteensopivuus: Järjestelmä tai komponentti voi vaihtaa tietoja muiden järjestelmien tai komponenttien kanssa ja/tai suorittaa vaaditut toiminnot samalla laitteisto- tai ohjelmistoympäristöllä.
        - Rinnakkaisuus: Järjestelmä suorittaa vaaditut toiminnot tehokkaasti jakamalla yhteisen ympäristön ja resurssit muiden komponenttien kanssa ilman haitallisia vaikutuksia muihin komponentteihin.
        - Yhteistoimivuus: Järjestelmä voi vaihtaa tietoja ja käyttää vaihdettua tietoa.
    - Palautettavuus: Järjestelmä voi keskeytyksen tai vian sattuessa palauttaa suoraan vaikuttavat tiedot ja järjestelmän halutun tilan. Yhteydessä saatavuuden pääkategoriaan.
- Skaalautuvuus: Järjestelmä on siinä määrin mukautuva, että ruuhkahuiput pystytään tasaamaan ja resursseja ohjaamaan niitä tarvitseville komponenteille.
    - Tehokkuus: Palvelinten ja komponenttien tulisi toimia mahdollisimman tehokkaasti resursseja tuhlaamatta.
    - Joustavuus/Siirrettävyys: Mahdollisuus uudelleen ohjata käytössä olevia resursseja.
        - Resurssien siirrettävyys: Työmäärä, joka tarvitaan resurssien siirtämiseen yhdestä ympäristöstä toiseen.
        - Yhteentoimivuus: Työmäärä, joka tarvitaan resurssien siirtämiseen yhdestä järjestelmästä toiseen.

    - Toiminnallinen soveltuvuus: Järjestelmä tarjoaa toimintoja, jotka täyttävät komponenttien ilmoitetut ja oletetut resurssien tarpeet, kun niitä käytetään tietyissä olosuhteissa.
    - Suorituskyky: Suorituskyky suhteessa käytettyjen resurssien määrään määritellyissä olosuhteissa.
        - Suoritusaika: Vaste- ja prosessointiajat sekä läpimenonopeudet.
        - Resurssien käyttö: Resurssien määrät ja tyypit.
        - Kapasiteetti: Järjestelmäparametrien enimmäisrajat.
- Ylläpidettävyys: Vaikuttavuus ja tehokkuus, jolla ylläpitäjä voi muokata järjestelmää. Työmäärä, joka tarvitaan vian paikantamiseen ja korjaamiseen. Kuinka helppoa on ymmärtää, muokata ja testata.
    - Joustavuus: Toimintaympäristön muutosten edellyttämien muutosten tekemisen helppous. Helpottaa muutosten sisällyttämistä, kun haluttu muutos on määritelty.
    - Testattavuus: Helpottaa tarkastuskriteerien asettamista ja tukee suorituskyvyn arviointia.
    - Siirrettävyys: Järjestelmää voidaan käyttää myös muissa kuin nykyisessä ympäristössä.
        - Sopeutuvuus: Järjestelmä voidaan mukauttaa erilaisille tai kehittyville käyttöympäristöille.
        - Asennettavuus: Järjestelmä voidaan asentaa ja/tai poistaa onnistuneesti tietyssä ympäristössä.
        - Vaihdettavuus: Toisen samaan tarkoitukseen määritetyn komponentin korvaaminen.
    - Uudelleenkäytettävyys: Komponentteja/koodia voidaan käyttää useammassa kuin yhdessä järjestelmässä tai muiden omaisuuksien rakentamiseen.
    - Yhteentoimivuus: Tarvittava työmäärä järjestelmän yhdistämiseksi toiseen järjestelmään.
    - Ymmärrettävyys/Analysoitavuus: Yhden tai useamman järjestelmän osan muutoksen vaikutusta järjestelmään, järjestelmän puutteiden tai vikojen syiden diagnosointi, muutettavien osien tunnistus.
    - Modulaarisuus: Yhden komponentin muutoksella on vähäinen vaikutus muihin komponentteihin.

- Muokattavuus: Järjestelmän muokkaaminen aiheuttamatta vikoja tai heikentämättä nykyistä laatua.
- Tyytyväisyys: Palvelinkokonaisuuden ylläpitäjien kokema tyytyväisyys palvelinkokonaisuuden ylläpidettävyyteen.
    - Käyttökelpoisuus: Missä määrin ylläpitäjä on tyytyväinen havaittuun käytännön tavoitteiden saavuttamiseen.
    - Luottamus: Missä määrin ylläpitäjä luottaa siihen, että järjestelmä toimii tarkoitetulla tavalla.

Edellä kuvattu kokonaisuus kattaa hyvin palvelinarkkitehtuurin laatua kuvaavia ja mittaavia attribuutteja. Monet ohjelmistojen laatua mittaavista attribuuteista on sovellettavissa pienin muokkauksin tai jopa sellaisenaan mittaamaan palvelinarkkitehtuurin laatua. Edellä kuvattu palvelinarkkitehtuurin laatumalli tarjoaa hyvät lähtökohdat palvelinarkkitehtuurin suunnitteluun.

4 PILVIPALVELUIDEN TUKI PALVELINARKKITEHTUURIN LAATUMALLILLE

Luvussa 3 pohdittiin palvelinarkkitehtuurin laatumallia. Luvussa 4 käydään läpi tarkemmin kyseistä laatumallia peilattuna pilvipalveluita ja niiden tarjoamia ratkaisuja vasten. Erityisesti käsittelyssä on Microsoftin Azure, Amazonin AWS sekä Google Cloud.

4.1 Tietoturva

Pilvitietoturva on kyberturvallisuuden ala, joka on erikoistunut pilvilaskentajärjestelmien suojaamiseen. Tämä sisältää tietojen pitämisen yksityisinä ja turvassa verkkopohjaisessa infrastruktuurissa, sovelluksissa ja alustoissa. Näiden järjestelmien suojaaminen edellyttää pilvipalveluntarjoajien ja niitä käyttävien asiakkaiden vaivannäköä riippumatta siitä, käyttävätkö niitä yksityiset tai yritykset. [61].

Pilvipalveluntarjoajat isännöivät palveluita palvelimillaan aina päällä olevien internet-yhteyksien kautta. Koska palveluntarjoajien liiketoiminta perustuu asiakkaiden luottamukseen, pilviturvamenetelmiä käytetään pitämään asiakkaiden tiedot yksityisinä ja turvallisesti tallennettuina. Pilvitietoturva on kuitenkin osittain myös asiakkaan käsissä. Molempien puolien ymmärtäminen on avainasemassa toimivan pilvitietoturvaratkaisun kannalta. [61].

Pilvitietoturva on joukko teknologioita, protokollia ja parhaita käytäntöjä, jotka suojaavat pilvilaskentaympäristöjä, pilvessä toimivia sovelluksia ja pilvessä olevia tietoja. Pilvipalveluiden turvaaminen alkaa ymmärtämällä, mitä tarkalleen suojataan, sekä siitä, mitä järjestelmänäkökohtia pitää hallita. [61].

Pilvitietoturva on suunniteltu suojaamaan seuraavia asioita:

- Fyysiset verkot (reitittimet, sähkövirta, kaapelointi, ilmastointilaitteet jne.)
- Tietojen tallennus (kiintolevyt jne.)
- Datapalvelimet, ydinverkon laskentalaitteistot ja -ohjelmistot
- Tietokoneen virtualisointikehykset, virtuaalikoneen ohjelmistot, isäntäkoneet ja vieraskoneet
- Käyttöjärjestelmät

- Middleware, sovellusohjelmointirajapintojen (API) hallinta
- Ajonaikaiset ympäristöt, käynnissä olevan ohjelman suorittaminen ja ylläpito
- Data, kaikki tallennetut, muokatut ja käytettävät tiedot
- Sovellukset, perinteiset ohjelmistopalvelut (sähköposti, vero-ohjelmistot, tuottavuusohjelmistot jne.)
- Loppukäyttäjien laitteet (tietokoneet, mobiililaitteet, IoT-laitteet jne.) [61].

Pilvipalveluissa näiden komponenttien omistajuus voi vaihdella suuresti. Tämä voi tehdä asiakkaan tietoturvavastuiden laajuuden epäselväksi. Pilven suojaaminen voi näyttää erilaiselta riippuen siitä, kenellä on määräysvalta kussakin komponentissa. [61].

Microsoftin Azure tarjoaa laajan valikoiman ominaisuuksia tietoturvan ylläpitämiseen ja rakentamiseen pilviympäristössä. Vastaavat ominaisuudet löytyvät myös kilpailevilta pilvipalveluiden tarjoajilta. Azuressa tietoturvaa koskevat sisäänrakennetut ominaisuudet on järjestetty kuuteen toiminta-alueeseen: toiminnot (operations), sovellukset (applications), tietojen tallennus (storage), tietoverkot (networking), laskenta (compute) ja käyttöoikeuksien hallinta (identity) [62].

Luvusta 2.4 löytyvästä palvelinarkkitehtuurin laatumallista tietoturvan alle on sijoitettu seuraavat laatuattribuutit: eheys, luottamuksellisuus, kiistämättömyys, vastuullisuus ja autenttisuus. Pilvipalvelut tarjoavat hyvin ratkaisuja näihin laatuattribuutteihin. Microsoft toteaa seuraavaa Azuren tarjoamista tietoturvaominaisuuksista: ”Microsoft Azure provides confidentiality, integrity, and availability of customer data, while also enabling transparent accountability” [62]. Tämä kutakuinkin kattaa suurimman osan edellä mainituista laatuattribuuteista, joita hyvältä palvelinarkkitehtuurilta odotetaan.

Azure tarjoaa Microsoft defender for cloud -ohjelmaa tietoturvan hallintaan. Defender for cloud toimii tietoturvaominaisuuksia yhteen kokoavana ohjelmana, jonka käyttöliittymän kautta on mahdollista tarkastella ja hallinnoida applikaatioiden ja ratkaisujen tietoturvaa. Lisäksi Azure tarjoaa muun muassa haavoittuvuuksien skannausta, läpäisytestausta, palomuuria, autentikoinnin hallintaa, diagnostiikka- ja analytiikkatyökaluja, tietojen ja tiedonsiirron salausta, verkkojen hallintaa, lokitusta ja tietojen varmuuskopiointia. Kokonaisuutena tarjolla on kattava paketti tietoturvaan ja siihen läheisesti liittyviä ominaisuuksia. [62].

4.1.1 Azuren, AWSn ja GCPn tarjoamat ratkaisut tietoturvaan

Kolme suurta pilvipalveluntarjoajaa, Amazon Web Services (AWS), Google Cloud Platform (GCP) ja Microsoft Azure, ottavat tietoturvan vakavasti ilmeisistä syistä. Yksi laajaa huomiota saava tietoturvaloukkaus, joka johtuu pilvipalveluntarjoajasta, voi karkottaa lukemattomia potentiaalisia asiakkaita, maksaa miljoonien eurojen tappiot ja mahdollisesti johtaa oikeudellisiin seuraamuksiin. Mitä kolme suurta pilvipalveluntarjoajaa tarjoavat neljällä tietoturvan toiminta-alueella. [71].

Verkko- ja infrastruktuuriturvallisuus:

- AWS tarjoaa useita suojausominaisuuksia ja palveluita, jotka on suunniteltu lisäämään yksityisyyttä ja hallitsemaan verkkoon pääsyä. Näitä ovat verkkopalomuurit, joiden avulla asiakkaat voivat luoda yksityisiä verkkoja ja hallita instansseja tai sovellusten käyttöä. Yritykset voivat hallita salausta AWS-palveluiden välillä. Mukana on myös liitäntävaihtoehtoja, jotka mahdollistavat yksityiset tai erilliset yhteydet, hajautettu palveluneston lieventämisteknologia, jota voidaan soveltaa osana sovellus- ja sisällöntoimitusstrategioita ja kaiken liikenteen automaattinen salaus AWSn maailmanlaajuisissa ja alueellisissa verkoissa AWS-suojattujen laitteiden välillä. [71], [72].
- Google on suunnitellut ja toteuttanut erityisesti tietoturvaan liittyviä laitteita, kuten Titan, mukautettu suojaussiru, jota GCP käyttää luomaan laitteiston luottamuksen perustan palvelimiinsa ja oheislaitteisiinsa. Google rakentaa oman verkkolaitteistonsa parantaakseen turvallisuutta. Tämä kaikki tiivistyy sen palvelinkeskusten rakenteisiin, jotka sisältävät useita fyysisiä ja loogisia suojakerroksia. Verkkopuolella GCP on suunnitellut ja kehittää edelleen globaalia verkkoinfrastruktuuria, joka tukee sen pilvipalveluita kestämään hyökkäyksiä, kuten hajautettua palvelunestoa (DDoS) ja suojelemaan palveluitaan ja asiakkaitaan. Globaalin verkkoinfrastruktuurinsa sisäänrakennettujen ominaisuuksien lisäksi GCP tarjoaa verkon suojausominaisuuksia, joita asiakkaat voivat halutessaan ottaa käyttöön. Näitä ovat pilvikuormituksen tasapainotus ja Cloud Armor, verkkotietoturvapalvelu, joka tarjoaa suojan DDoS- ja sovellushyökkäyksiä vastaan. Google käyttää useita turvatoimia varmistaakseen siirrettävien tietojen aitouden, eheyden ja yksityisyyden. Se salaa ja todentaa siirrettävät tiedot yhdessä tai useammassa verkkokerroksessa,

kun tiedot siirtyvät Googlen hallitsemattomien fyysisten rajojen ulkopuolelle. [71], [73].

- Microsoft Azure toimii Microsoftin hallinnoimissa ja ylläpitämissä datakeskuksissa. Yrityksen mukaan nämä maantieteellisesti hajallaan sijaitsevat datakeskukset täyttävät alan keskeiset turvallisuus- ja luotettavuusstandardit. Palvelinkeskuksia valvoo ja hallinnoi Microsoftin henkilökunta. Microsoft myös suorittaa taustatarkistuksia operatiiviselle henkilöstölle ja rajoittaa pääsyä sovelluksiin, järjestelmiin ja verkkoinfrastruktuuriin käyttöoikeustason mukaan. Azure Firewall on hallittu, pilvipohjainen verkkotietoturvapalvelu, joka suojaa Azure Virtual Network -resursseja. Se on firewall as a service, jossa on sisäänrakennettu korkea käytettävyys ja rajoittamaton skaalautuvuus. Azure Firewall voi purkaa lähtevän liikenteen salauksen, suorittaa vaaditut suojaustarkastukset ja salata sitten liikenteen uudelleen ennen sen välittämistä määränpäähänsä. Järjestelmänvalvojat voivat sallia tai estää käyttäjien pääsyn verkkosivustoluokkiin, kuten uhkapeleihin, sosiaaliseen mediaan tai muihin. [71], [62].

Identiteetti ja kulunvalvonta:

- AWS tarjoaa ominaisuuksia, joilla voidaan hallita käyttäjien käyttöoikeuksia kaikissa AWS-palveluissa. Näitä ovat AWS Identity and Access Management (IAM), jonka avulla yritykset voivat määrittää yksittäisiä käyttäjätilejä, joilla on käyttöoikeudet AWS-resursseihin ja AWS Multi-Factor Authentication, joka sisältää vaihtoehdot ohjelmisto- ja laitteistopohjaisille todentamiselle. AWS IAMn avulla voidaan myöntää työntekijöille ja sovelluksille yhdistetty käyttöoikeus AWS-hallintakonsoliin ja AWS-palvelun sovellusliittymiin käyttämällä olemassa olevia identiteettijärjestelmiä, kuten Microsoft Active Directorya. AWS tarjoaa myös AWS Directory Servicen, jonka avulla organisaatiot voivat integroitua identiteettijärjestelmien kanssa hallinnollisten rasitteiden vähentämiseksi ja loppukäyttäjäkokemuksen parantamiseksi sekä AWS Single Sign-On (SSO), jonka avulla organisaatiot voivat hallita käyttäjien käyttöoikeuksia AWSssä. [71], [72].
- Googlen Cloud Identity and Access Management tarjoaa useita tapoja hallita identiteettejä ja rooleja Google Cloudissa. Ensinnäkin Cloud IAM antaa järjestelmänvalvojille valtuudet määrittää, kuka pääsee käsiksi tiettyihin resursseihin, mikä tarjoaa täyden hallinnan ja näkyvyyden GCP-resurssien

keskitettyyn hallintaan. Lisäksi yrityksille, joilla on monimutkaiset organisaatiorakenteet, satoja työryhmiä ja monia projekteja, Cloud IAM tarjoaa yhtenäisen kuvan koko organisaation tietoturvapolitiikasta sisäänrakennetulla auditoinnilla, joka helpottaa sisäistä valvontaa. Saatavilla on myös Cloud Identity, identiteetti palveluna (IDaaS), joka hallitsee keskitetysti käyttäjiä ja ryhmiä. Yritykset voivat määrittää Cloud Identityn yhdistämään käyttäjätiedot Googlen ja muiden identiteetintarjoajien välillä. GCP tarjoaa myös Titan-suojausavaimia, jotka tarjoavat kryptografisen todisteen siitä, että käyttäjät ovat vuorovaikutuksessa laillisten palvelujen kanssa (eli palvelujen kanssa, joihin he rekisteröivät suojausavaimensa) ja että heillä on suojausavain hallussaan. Lopuksi Cloud Resource Manager tarjoaa resurssikokonaisuuksia, kuten organisaatioita, kansioita ja projekteja, joiden avulla organisaatiot voivat ryhmitellä ja järjestää hierarkkisesti GCP-resursseja. [71], [73].

- Azure Active Directory (Azure AD) on yritysidentiteettipalvelu, joka tarjoaa kertakirjautumisen, monivaiheisen kirjautumisen ja ehdollisen pääsyn Azure-palveluihin sekä yritysverkkoihin, paikallisiin resursseihin ja tuhansiin SaaS-sovelluksiin. Azure ADn avulla organisaatiot voivat suojata käyttäjätietoja suojatulla adaptiivisella pääsyllä, yksinkertaistaa ja virtaviivaistaa valvontaa yhdistetyn identiteetinhallinnan avulla ja varmistaa yksinkertaistetun identiteetin hallinnan noudattaminen. Microsoft sanoo, että se voi auttaa suojaamaan käyttäjiä 99,9 prosentilta kyberturvallisuushyökkäyksistä. [71], [62].

Tietosuoja ja salaus:

- AWS tarjoaa mahdollisuuden lisätä suojauskerroksen pilvessä oleviin tietoihin. Se tarjoaa skaalattavia salausominaisuuksia, mukaan lukien lepotilassa olevien tietojen salausominaisuudet useimmissa AWS-palveluissa, mukaan lukien Amazon EBS, Amazon S3, Amazon RDS, Amazon Redshift, Amazon ElastiCache, AWS Lambda ja Amazon SageMaker. Saatavilla on myös joustavia avaintenhallintavaihtoehtoja, mukaan lukien AWS-avainhallintapalvelu, jonka avulla yritykset voivat valita, hallinnoiko AWS salausavaimia vai pitävätkö yritykset ne omassa hallinnassaan. Laitteistopohjainen salausavainten tallennus AWS CloudHSMää käyttäen ja salatut viestijonot arkaluontoisten tietojen lähettämiseksi käyttämällä palvelinpuolen salausta (SSE) Amazon SQSlle. [71], [72].

- Google tarjoaa Confidential Computingia, jota se kutsuu "läpimurtoteknologiaksi", joka salaa käytössä olevat tiedot samalla kun tietoja käsitellään. Luottamukselliset laskentaympäristöt pitävät tiedot salattuna muistissa ja muualla keskusyksikön ulkopuolella. Ensimmäinen tuote Confidential Computing -portfoliossa on Confidential VM. Google käyttää jo nyt erilaisia eristys- ja hiekkalaatikkotekniikoita osana pilvi-infrastruktuuriaan tehdäkseen usean vuokraajan arkkitehtuuristaan turvallisen ja Confidential VM vie tämän uudelle tasolle tarjoamalla muistin salauksen, jotta käyttäjät voivat eristää työkuormia paremmin pilvessä. Toinen tuote, Cloud External Key Manager (Cloud EKM), antaa organisaatioille mahdollisuuden käyttää tuetussa ulkoisessa avaintenhallintakumppanissa hallinnoimiaan avaimia tietojen suojaamiseen Google Cloud Platformissa. Yritykset voivat säilyttää avainten alkuperän kolmannen osapuolen avaimiin nähden ja hallita avainten luomista, sijaintia ja jakelua. Heillä on myös täysi määräysvalta siihen, kuka käyttää heidän avaimiaan. [71], [73].
- Azure Key Vault auttaa suojaamaan pilvisovellusten ja -palvelujen käyttämiä salausavaimia. Azure Key Vault on suunniteltu virtaviivaistamaan avainten hallintaprosessia ja antamaan yrityksille mahdollisuuden hallita avaimia, joilla pääsee käsiksi tietoihin ja jotka salaavat niitä. Kehittäjät voivat luoda avaimia kehitystä ja testausta varten muutamassa minuutissa ja sitten siirtää ne tuotantoavaimiin. Järjestelmänvalvojat voivat tarvittaessa myöntää ja peruuttaa avaimia koskevia käyttöoikeuksia. Microsoft Information Protection ja Microsoft Information Governance auttavat suojaamaan ja hallitsemaan tietoja Microsoft 365:ssä. Microsoft Information Protection laajentaa tietojen katoamisen eston kaikkiin Microsoft 365 -sovelluksiin ja -palveluihin sekä Windows 10:een ja Edgeen. Azure Purview auttaa organisaatioita ymmärtämään, missä niiden strukturoidut tiedot sijaitsevat, jotta ne voivat suojata ja hallita niitä paremmin. [71], [62].

Sovelluksen suojaus:

- AWS Shield on hallittu DDoS-suojauspalvelu, joka suojaa Amazon-pilvessä käynnissä olevia sovelluksia. AWS Shield tarjoaa aina päällä olevan tunnistuksen ja automaattiset sisäiset rajoitukset, jotka on suunniteltu minimoimaan sovelluksen seisokit ja latenssi. AWS Shieldissä on kaksi tasoa, Standard ja Advanced. Kaikilla AWS-asiakkailla on oikeus AWS Shield Standardin

automaattisiin suojauksiin, jotka yhtiön mukaan suojaavat yleisimmiltä DDoS-hyökkäyksiltä, jotka kohdistuvat verkkosivustoihin tai sovelluksiin. Kun Shield Standardia käytetään Amazon CloudFrontin ja Amazon Route 53:n kanssa, asiakkaat saavat kattavan suojan kaikkia tunnettuja infrastruktuurihyökkäyksiä vastaan. Yritykset voivat valita AWS Shield Advancedin paremman suojan hyökkäyksiä vastaan, jotka on kohdistettu Amazon EC2:ssa, Elastic Load Balancingissa, Amazon CloudFrontissa, AWS Global Acceleratorissa ja Amazon Route 53:ssa toimiviin sovelluksiin. Shield Standardin mukana tulevien suojausten lisäksi Shield Advanced tarjoaa ylimääräistä havaitsemista ja lieventämistä suuria ja kehittyneitä DDoS-hyökkäyksiä vastaan, lähes reaaliaikaisen näkyvyyden hyökkäyksille ja integroinnin AWS WAFiin, pilvipalveluntarjoajan verkkosovelluspalomuuriin. [71], [72].

- Google Cloud Web App and API Protection (WAAP) tarjoaa kattavan uhkien suojauksen verkkosovelluksille ja APIille. Cloud WAAP perustuu samaan tekniikkaan, jota Google käyttää julkisten palveluidensa suojaamiseen verkkosovellusten hyväksikäytöltä, DDoS-hyökkäyksiltä, petolliselta bot-toiminnalta ja APIin kohdistuvilta uhilta. Cloud WAAP edustaa siirtymistä yhdistettyyn sovellusten suojaukseen ja se on suunniteltu parantamaan uhkien ehkäisyä, lisäämään toiminnan tehokkuutta sekä lisäämään näkyvyyttä ja telemetriaa. Se tarjoaa myös suojan pilvissä ja paikallisissa ympäristöissä. Cloud WAAP yhdistää kolme tuotetta tarjotakseen kattavan suojan uhkia ja petoksia vastaan. Yksi niistä on Google Cloud Armor, joka on osa GCPn maailmanlaajuista kuormituksen tasapainotusinfrastruktuuria ja tarjoaa verkkosovellusten palomuuri- ja DDoS-torjuntaominaisuuksia. Toinen on Apigee API Management, joka tarjoaa API-elinkaarihallintaominaisuuksia, joissa keskitytään voimakkaasti turvallisuuteen. Kolmas on reCaptcha Enterprise, joka tarjoaa suojan petolliselta toiminnalta, roskapostilta ja väärinkäytöksiltä, kuten kirjautumistietojen automaattiselta täyttämiseltä (credential stuffing), automaattiselta tilien luomiselta ja bottien aiheuttamilta uhilta. Cloud Security Scanner etsii haavoittuvuuksia ja dataa verkkosovellusten haavoittuvuuksista ja antaa yrityksille mahdollisuuden ryhtyä toimiin ennen kuin hyökkääjät voivat hyödyntää niitä. [71], [73].
- Microsoft Cloud App Security on pilvisovellusten tietoturvavälittäjä, joka yhdistää toimintojen näkyvyyden, tiedonkulun hallinnan, käyttäjien toiminnan seurannan ja kehittyneen analytiikan, minkä ansiosta yritykset voivat tunnistaa kyberuhkia ja

torjua niitä kaikissa Microsoftin ja kolmannen osapuolen pilvipalveluissa. Tietoturva-ammattilaisille suunniteltu Cloud App Security integroituu natiivisti tietoturva- ja identiteettityökaluihin, kuten Azure Active Directory, Microsoft Intune ja Microsoft Information Protection, ja tukee erilaisia tiloja, mukaan lukien lokien kerääminen, APIt ja käänteinen välityspalvelin (reverse proxy). [71], [62].

4.2 Saatavuus

Googlella, Amazonilla ja Microsoftilla on pilvipalveluidensa osalta palvelutasosopimukset, joissa määritellään saatavuudeksi vähintään 99,9 % jokaista maksullista palvelua kohti, mutta enintään 99,99 %. Perspektiivistä katsottuna 99,9 %:n saatavuus tarkoittaa vain yhdeksän tunnin häiriöaikaa vuodessa ja 99,99 % alle tunnin häiriöaikaa vuodessa (Taulukko 1). Suuret pilvipalveluntarjoajat voivat täyttää näiden sopimusten suhteellisen korkean riman monimutkaisuudesta huolimatta vakiintuneiden prosessien ansiosta. [63].

Pilvipalveluiden osalta saatavuuden kanssa ei juuri ole ongelmia ja suurimmat saatavuuteen liittyvät ongelmat tulevat ohjelmistojen ja applikaatioiden puolelta niiden monimutkaisuudesta johtuen. Sovelluksen monimutkaisuuden kasvaessa myös riski häiriöaikaan kasvaa. Sen pienentäminen vaatii usein kustannusten kasvattamista.

Korkean saatavuuden järjestelmien rakentaminen merkitsee yrityksille suuria kustannuksia. Tällaisiin toimiin ei riitä, että palvelinkeskuksessa on vain vikasietoinen palvelimien klusteri, palvelinkeskuksessa on myös oltava useita redundantteja energialähteitä ja jopa replikointi useiden maantieteellisten sijaintien välillä katastrofien sattuessa. Hyvin suuria, monikansallisia yrityksiä lukuun ottamatta, melkein kenelläkään ei ole varaa tällaisiin järjestelyihin. [64].

Pilvipalveluiden myötä tällaisen palvelun rakentamiskustannukset ovat kuitenkin laskeneet dramaattisesti. Useimpien pilvipohjaisten palveluntarjoajien, erityisesti ohjelmistopohjaisten palvelujen, on nyt mahdollista tarjota erittäin aggressiivisia palvelutasosopimuksia saatavuuden osalta. [64].

Pilvipalveluiden asiakkaat haluavat ennen kaikkea olla varmoja, että palvelu säilyy saatavilla. Parhaimmillaan saatavuus voi yleensä olla viisi 9:ää tai 99,999 %:a. Kaikki palveluntarjoajat eivät kuitenkaan tarjoa näin kattavaa saatavuutta. Itse asiassa, kun tarkastellaan kokonaista vuotta, monien yritysten tarjonta jää selkeästi tämän alle. Moni

palveluntarjoaja saattaa tarjota 99 %:n saatavuutta SLA-sopimuksessaan. Tämä saattaa kuulostaa hyvältä, mutta 99 %:n saatavuus voi jättää palvelun alas useiksi tunneiksi kerrallaan, kaikki sopimusrajojen sisällä. Oheisesta taulukosta selviää, että 99 %:n saatavuudella palvelu voi olla poissa käytöstä lähes 88 tuntia vuodessa. Tämä osoittaa, kuinka tärkeä todellinen luku on. [74].

Saatavuusaika prosentteina	Keskimääräinen vuotuinen katkosaika
99 %	87 tuntia 40 minuuttia
99,9 %	8 tuntia 46 minuuttia
99,99 %	52 minuuttia 36 sekuntia
99,999 %	5 minuuttia 16 sekuntia
99,9999 %	31,6 sekuntia

Taulukko 1. Palvelun saatavuus [74].

Luvuista käy ilmi, että markkinoilla tarjottavien palveluiden tasoissa on suuria eroja. Palvelutaso 99,9 %:n saatavuudella tarjoaa itse asiassa huomattavasti vähemmän katkoksia kuin 99 %:n taso. Tämä ei välttämättä välity pelkkiä 9:ä tuijottamalla.

Vertaillaan kolmen suuren pilvipalvelutarjoajan SLA-sopimuksia. Kaikilta pilvipalveluilta on otettu vertailuun SLA, joka koskee yksittäisen sovelluksen saatavuutta yhdestä palvelinkeskuksesta käsin yhden kuukauden tarkastelujaksolla. Amazon Web Services toteaa SLA-sopimuksessa, että jos kuukausittainen käyttöaikaprosentti laskee alle 99,5 %:n, käyttäjällä on oikeus 10 %:n palveluhyvitykseen. Jos se putoaa alle 99,0 %, käyttäjä saa takaisin 30 %. Jos saatavuus on alle 95 % niin korvaus on 100 %. [75].

Microsoft Azuren pilvipalvelujen SLA tarjoaa vastaavia korvauksia: 10 %:n palveluhyvitys alle 99,95 %:n käyttöajasta ja 25 %:n hyvitys alle 99 %:n käyttöajasta ja 100 %:n hyvitys alle 95 %:n käyttöajasta [76]. Google Cloudin SLA tarjoaa 10 %:n hyvityksen alle 99,95 %:n käyttöajasta, 25 %:n hyvityksen alla 99 %:n käyttöajasta ja 50 %:n hyvityksen alle 95 %:n käyttöajasta [77]. AWS ja Azure tarjoavat hyvinkin yhtenevät SLA:t, mutta Googlen SLA tarjoaa hieman heikommat korvaukset kuin edellä mainitut.

Kaikissa näissä laskelmissa oletetaan, että on tiedossa, mitä katkosajalla tarkoitetaan. Katkosaikana pidetään yleensä aikaa, jolloin järjestelmä tai palvelu ei toimi. Yritykset tekevät kaikkensa maksimoidakseen käytettävyyden. Mutta se, mitä asiakas näkee ja mitä palveluntarjoaja näkee, voivat olla kaksi eri asiaa. [74].

Valitettavasti on erittäin helppoa joutua väärään turvallisuuden tunteeseen, kun on kyse palvelun saatavuudesta pilvessä. Saatavuuteen voi vaikuttaa seuraavat seikat:

- Palvelutasosopimuksia ei aina täytetä
- Katkosaika lisääntyy useiden palveluiden myötä
- Kolmannen osapuolen palveluiden katkokset
- Pilvi on saatavilla, mutta palvelu on poissa käytöstä
- Huono palvelun laatu

Ilmeisin ongelma on se, että pilvi on saatavilla, mutta palvelu taikka sovellus on poissa käytöstä. Joskus palvelu ei toimi, vaikka kaikki infrastruktuurissa on toiminnassa. Käyttäjä voi huomata olevansa eri mieltä pilvipalveluntarjoajansa kanssa, kun pilvipalveluntarjoaja sanoo pilven olevan toiminnassa, mutta palvelu on kuitenkin alhaalla. Kuka on oikeassa ja kuka on vastuussa? Mitä hyötyä pilvipalvelusta on, jos palvelun taikka sovelluksen tekninen laatu on niin huono, että se on käyttökelvoton? Palvelun laatuongelmat eivät välttämättä koskaan näy käyttöajan seurantatilastoissa. Katkosaikojen hallinta on tärkeä osa SLA-hallintaa. Jos käyttäjä ei pysty selvittämään katkoksia, mistä tietää, mikä käyttöaika on? [74].

Jokainen palvelu/sovellus on erilainen ja palveluvaatimukset vaihtelevat liiketoiminnan tarpeiden mukaan. Pilveen siirtyminen on saanut ihmiset ajattelemaan palvelujen tai sovellusten saatavuutta verkon saatavuuden sijaan. SLA:n periaatteet ovat samat ja kysymys kuuluukin, vastaako tarjottu palvelun taso odotuksia? Kumpikin osapuoli taistelee omien etujensa puolesta. Käyttäjät haluavat hyvin toimivan palvelun ja palveluntarjoajat haluavat perustella tarjoamiensa palveluiden laskutuksen. SLA:han on syytä tutustua hyvin. [74].

4.3 Vikasietoisuus

Vikasietoisuus viittaa järjestelmän (tietokone, verkko, pilviklusteri jne.) kykyyn jatkaa toimintaansa keskeytyksettä, kun yksi tai useampi sen komponenteista hajoaa. Vikasietoisen järjestelmän luomisen tavoitteena on estää yhdestä vikapisteestä syntyviä häiriöitä ja varmistaa kriittisten sovellusten tai järjestelmien korkea käytettävyys ja liiketoiminnan jatkuvuus. [65].

Vikasietoisissa järjestelmissä käytetään varmuuskopiokomponentteja, jotka automaattisesti korvaavat vialliset komponentit ja varmistavat, että palvelun toimintakykyä ei menetetä. Nämä sisältävät:

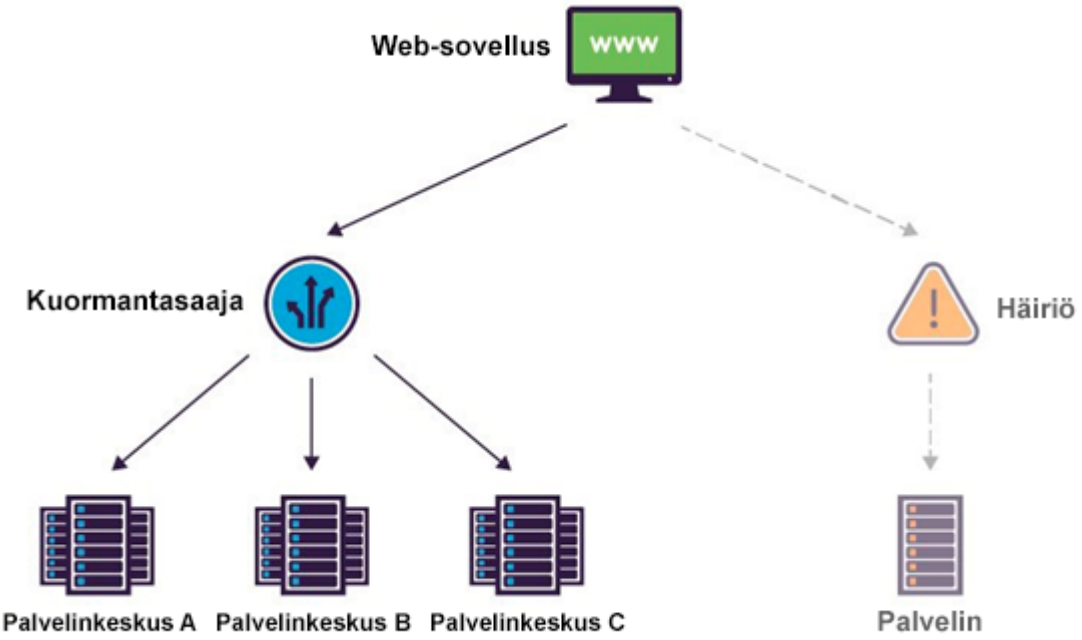
- Laitteistojärjestelmät, jotka on varmuuskopioitu identtisillä tai vastaavilla järjestelmillä. Esimerkiksi palvelimesta voidaan tehdä vikasietoinen käyttämällä identtistä palvelinta, joka toimii rinnakkain ja kaikki toiminnot peilataan varapalvelimelle.
- Ohjelmistojärjestelmät, jotka on varmuuskopioitu muilla ohjelmistoinstansseilla. Esimerkiksi asiakastietoja sisältävää tietokantaa voidaan jatkuvasti replikoida toiselle koneelle. Jos ensisijainen tietokanta kaatuu, toiminnot voidaan ohjata automaattisesti toiseen tietokantaan.
- Virtalähteet, jotka on tehty vikasietoisiksi vaihtoehtoisilla lähteillä. Esimerkiksi monilla organisaatioilla on sähkögeneraattoreita, jotka voivat tuottaa virtaa, jos päävirta katkeaa.

Samalla tavalla mikä tahansa järjestelmä tai komponentti voidaan tehdä vikasietoiseksi redundanssin avulla. [65].

Vikasietoisuus voi olla tärkeä osa katastrofipalautusstrategiaa. Esimerkiksi vikasietoiset järjestelmät, joissa on varmuuskopiokomponentteja pilvessä, voivat palauttaa kriittiset järjestelmät nopeasti, vaikka luonnon tai ihmisen aiheuttama katastrofi tuhoaisi paikan päällä olevan IT-infrastruktuurin. [65].

Vikasietoisuus liittyy läheisesti saatavuuteen, mutta menee yhden askeleen pidemmälle takaamalla häiriöajattomuuden. Azuren tapauksessa järjestelmä voidaan siirtää kokonaan toiseen tietokeskukseen, jos kokonainen tietokeskus on poissa käytöstä. Näin mahdollistetaan häiriöajattomuus ja 100% saatavuus. [66].

Verkkosovellusten yhteydessä vikasietoisuus liittyy kuormituksen tasaus- ja vikasietoratkaisujen käyttöön käytettävyyden varmistamiseksi redundanssin ja nopean katastrofipalautuksen kautta [65].



Kuva 10. Kuormantasaus on tärkeä osa vikasietoisuutta [65].

Kuormantasausratkaisut mahdollistavat sovelluksen toimimisen useissa verkkosolmuissa, mikä poistaa huolta yhdestä vikapisteestä syntyvästä häiriöstä. Useimmat kuormantasaajat myös optimoivat työtaakan jakautumisen useiden laskentaresurssien kesken, mikä tekee niistä yksittäin kestävämpiä aktiivisuuspiikkejä vastaan, jotka muutoin aiheuttaisivat hitautta ja muita häiriöitä. Lisäksi kuormantasaus auttaa selviytymään osittaisista verkkovioista. Esimerkiksi järjestelmä, jossa on kaksi tuotantopalvelinta, voi käyttää kuormantasaajaa siirtämään automaattisesti työkuormia yksittäisen palvelimen vian sattuessa. [65].

Kuten todettua, vikasietoisuuteen liittyy keskeisesti katastrofipalautusstrategiat. Nopea katastrofipalautus rakentuu kahden konseptin varaan. Ensinnäkin varmuuskopiointi ja sitä kautta nopea palautus ovat keskiössä. Tämä tarkoittaa yleensä paikallisten työkuormien varmuuskopiointia ja pilven käyttöä tietovarastona, josta palautus tapahtuu suoraan pilvipalveluntarjoajan alustalle. [78].

Tätä mallia tukevia ratkaisuja on laaja valikoima AWS:ssä ja Azuressa aina lukuisiin kolmannen osapuolen palveluntarjoajiin asti, jotka varmuuskopioivat tiedot julkiseen pilveen ja suorittavat sitten tarvittavat muutokset lähtöinfrastruktuuriin, jotta ne voivat toimia natiivisti julkisessa pilvessä. [78].

Toisena vaihtoehtona on tukeutua replikointiin ja automaatioon. Tässä lähestymistavassa on ideana replikoida ympäristö julkiseen pilveen ja tuotantohäiriön sattuessa nopeasti siirtyä pilvireplikoituun ympäristöön. Nämä palvelut on usein rakennettu tietosuojatuiksi niin, että tietoja replikoidaan pienissä erissä tiukkojen tietojen häviämistavoitteiden saavuttamiseksi. Tämä voidaan tehdä natiivisti Azure Site Recoveryn ja AWS Cloudenduren kautta sekä useiden ulkopuolisten palveluntarjoajien kautta. [78].

AWS ja Azure tarjoavat kattavia ratkaisuja varmuuskopiointiin ja palautukseen sekä replikointipalveluihin. Google puolestaan luottaa kolmannen osapuolen tarjoajaekosysteemiin. AWS:llä ja Azurella on lukuisia työkaluja ja palveluita, jotka tukevat useita katastrofipalautusmenetelmiä. Ne kaikki toimivat samalla tavalla ja tukevat sekä pilvi- että paikallisia palvelimia, jotka voidaan palauttaa suoraan pilvialustalle huolehtien samalla tarvittavista muutoksista. Palveluista veloitetaan yleensä datamäärän tai työmäärän mukaan, palvelusta ja palveluntarjoajasta riippuen. [78].

Liiketoiminnan jatkuvuussuunnitelman kannalta palautusajaksi luvataan Azuren taholta korkeintaan kaksi tuntia kuukausitasolla, mutta kuitenkin useimmissa yksittäisissä tapauksissa vain muutama minuutti [81]. AWS tarjoaa eritasoisia ja hintaisia vaihtoehtoaja lähtien halvimmasta vaihtoehdosta, jossa palautusaika lasketaan tunneissa aina kalliiseen reaaliaikaista palautusta lupaavaan vaihtoehtoon [82].

On monia kolmannen osapuolen palveluntarjoajia, jotka tarjoavat hallittuja katastrofipalautuspalveluita, jotka voivat usein olla sopivampia kuin pilvipalveluntarjoajien omat ratkaisut. Katastrofipalautus voi olla monimutkaista ja kallista rakentaa ja testata ja organisaation on varmistettava, että sillä on sisäiset taidot ja resurssit, joita tarvitaan katastrofipalautussuunnitelman tehokkaaseen käynnistämiseen ja toimittamiseen tarvittaessa. [78].

Palvelujen ottaminen palveluntarjoajalta poistaa riskit ja helpottaa sisäisten tiimien vaatimuksia. Jos yritys kuitenkin harkitsee DRaaS:ää (Disaster Recovery as a Service), yrityksen on tärkeää muistaa säilyttää kontrolli ja valvonta. On tärkeää, että yritys ymmärtää, testaa ja pysyy mukana toimituksessa alusta loppuun. [78].

Pilvipohjainen katastrofinpalautusjärjestelmä on houkutteleva joustavuutensa puolesta. Sen avulla yritys voi rakentaa ympäristöjä nopeasti ja maksaa vain siitä mitä se tarvitsee sen sijaan, että se rahoittaisi käyttöä odottavaa lepäävää omaisuutta. Kuitenkin, jos

vaatimuksena on jatkuvasti toimiva katastrofipalautusinfrastruktuuri, siitä voi tulla kallista ja kiinteähintainen palvelu voisi olla sopivampi. [78].

Monimutkaisuutta ei myöskään voida jättää huomiotta. Pilven avulla IT-tiimin vastuulla on suuri osa asianmukaisen alustan toimittamisesta, jota ylläpidetään, tuetaan ja joka on käytettävissä tarvittaessa. Mitä tulee katastrofipalautukseen, on tärkeää, että organisaatiot ymmärtävät täysin, mitä he tarvitsevat ympäristöltä ja mitä taitoja tarvitaan sen käyttämiseen. Kun ne on määritelty, organisaatioilla on hyvät edellytykset ymmärtää, pitäisikö julkinen pilvi olla osa heidän katastrofipalautussuunnitelmaansa. [78].

4.4 Skaalautuvuus

Yksi pilvipalveluiden tärkeimmistä ominaisuuksista on skaalautuvuus. Yritykset voivat lisätä tai vähentää IT-resursseja, kuten prosessointitehoa ja tiedontallennuskapasiteettia, ilman toimintahäiriöitä. Yritysten ei tarvitse käyttää viikkoja tai kuukausia infrastruktuurin uudistamiseen, kuten jouduttaisiin tekemään paikallisten konesalien yhteydessä. Sen sijaan pilvipalveluntarjoajilla on jo infrastruktuuri olemassa ja yritykset voivat helposti lisätä palvelimia tarpeen mukaan saavuttaakseen tavoitteensa. Kun lisävaatimusten kysyntä on kadonnut, voidaan palata alkuperäiseen kokoonpanoon. [67].

Skaalautuvuuden kaltainen käsite on pilven elastisuus, joka on järjestelmän kyky laajentua ja supistua työkuormitustarpeiden perusteella. Vaikka nämä kaksi käsitettä kuulostavat samalta, tärkein ero pilven skaalautuvuuden ja pilven elastisuuden välillä on aika. Pilven elastisuutta vaaditaan lyhytaikaisissa kuormissa, kuten käyttäjäpiikkien yhteydessä. Pilven skaalautuvuus puolestaan on strategisesti suunniteltua pitkän aikavälin kasvua. [67], [69].

Skaalaustyypit pilvipalveluissa:

- Vertikaalinen skaalaus: Vertikaalinen skaalaus tarkoittaa resurssien, kuten keskusmuistin tai prosessointitehon, lisäämistä olemassa olevaan palvelimeen, kun työmäärä lisääntyy. Tämän tyyppinen skaalaus ei vaadi koodin muutoksia, koska palvelimelle lisätään vain lisälaajennusyksiköitä. Palvelimen koko ja kapasiteetti rajoittavat kasvun kokonaismäärää ja voi näin ollen vaikuttaa suorituskykyyn. [67], [68], [69].

- Vaakasuora skaalaus: Kun tarvitaan suurempaa kapasiteettia, suorituskykyä, tallennustilaa, muistia ja ominaisuuksia, voidaan lisätä palvelimia alkuperäiseen pilvi-infrastruktuuriin niin, että ne toimivat yhtenä järjestelmänä. Tällainen skaalaus on monimutkaisempaa kuin yhden palvelimen vertikaalinen skaalaus, koska mukana on muita palvelimia. Jokaisen palvelimen on oltava itsenäinen, jotta niitä voidaan kutsua erikseen skaalattaessa. Vaakasuoralla skaalauksella organisaatiot voivat kasvaa loputtomasti, koska rajoituksia ei ole. [67], [68], [69].
- Diagonaalinen skaalaus: Diagonaalinen skaalaus on yhdistelmä vertikaalista ja vaakasuoraa skaalausta. Organisaatiot voivat kasvaa pystysuunnassa, kunnes ne saavuttavat palvelimen rajan, ja sitten kloonata palvelimen lisätäkseen resursseja tarpeen mukaan. Tämä on hyvä ratkaisu organisaatioille, jotka kohtaavat odottamattomia tehopiikkejä, koska sen avulla ne voivat olla ketteriä ja joustavia skaalautuakseen tai pienentääkseen. [67], [68].

Tärkeimmät pilven skaalautuvuuden edut:

- Käytännöllisyys: IT-järjestelmänvalvojat voivat helposti lisätä vain muutamalla napsautuksella lisää virtuaalikoneita, jotka ovat saatavilla viipymättä ja jotka on räätälöity juuri organisaation tarpeisiin. Tämä säästää IT-henkilöstön arvokasta aikaa. Sen sijaan, että käytettäisiin tunteja ja päiviä fyysisen laitteiston asentamiseen, tiimit voivat keskittyä muihin tehtäviin. [69], [67], [68].
- Joustavuus ja nopeus: Kun liiketoiminnan tarpeet muuttuvat ja kasvavat, mukaan lukien odottamattomat kysynnän piikit, pilven skaalautuvuus mahdollistaa reagoinnin nopeasti. Pienilläkin yrityksillä on käytettävissään tehokkaita resursseja, jotka ennen olivat kalliita. Yrityksiä eivät enää sido vanhentuneet laitteet ja ne voivat päivittää järjestelmiä ja lisätä tehoa ja tallennustilaa helposti. [69], [67], [68].
- Kustannussäästöt: Pilven skaalautuvuuden ansiosta yritykset voivat välttää kalliiden laitteiden hankinnan alkukustannukset, jotka saattavat vanhentua muutamassa vuodessa. Pilvipalveluntarjoajien kautta he maksavat vain käyttämästään ja minimoivat jätteen. [69], [67], [68].
- Palautus: Skaalautuvan pilvipalvelun avulla voidaan vähentää palautuskustannuksia poistamalla tarpeen rakentaa ja ylläpitää toissijaisia datakeskuksia. [69].

- Luotettavuus: Organisaatiot voivat luottaa korkeaan suorituskykyyn, koska skaalautuva arkkitehtuuri pystyy vastaamaan kysynnän äkilliseen kasvuun tai laskuun [67].

Azure, AWS ja Google Cloud tarjoavat kaikki samantapaiset automaattisesti skaalautuvat palvelut vaakasuoraa skaalausta hyödyntäen. Azurella on Azure Autoscale palvelu sovellusten dynaamiseen skaalaamiseen. Autoscale on sisäänrakennettu ominaisuus pilvi- sekä mobiiliominaisuuksissa kuin myös virtuaalikoneissa sekä verkkosivuissa, jotka toimivat Azuressa. Autoscale auttaa sovelluksia toimimaan kysynnän muutoksissa. Suorituskyky tarkoittaa eri asioita eri sovelluksissa. Jotkut sovellukset ovat laskentatehoa vaativia, toiset vaativat suureen määrän muistia. Autoscale voi skaalata palvelun minkä tahansa metriikan mukaan. [83].

Esimerkiksi verkkosovellus, joka käsittelee miljoonia pyyntöjä päivällä ja ei yhtään yöllä, voidaan skaalata kysyntää vastaavaksi. Ajastuksen avulla voidaan reagoida jo ennalta mahdollisiin kysyntäpiikkeihin kuten black friday ruuhkaan verkkokaupassa. Jos palvelun kysyntäpiikki osuu aina tietylle kellonajalle, voidaan kyseiselle ajanjaksolle suunnitella korkeammat skaalaustavoitteet. Jos palvelua käytetään lähtökohtaisesti virka-aikana, voidaan arki-illoiksi ja viikonlopuksi vähentää kapasiteettia. [83].

Autoscale voi myös seurata keskeisiä suorituskykymittareita ja varoittaa, jos jokin näistä muuttuu. Hälytyksiä voidaan asettaa melkein minkä tahansa metriikan, kuten suorittimen tilan tai vasteajan, perusteella. Myös tapahtumista voidaan luoda hälytyksiä, esim. silloin, kun itse automaattinen skaalautuminen laukeaa. [83].

AWS Auto Scaling tarkkailee sovelluksia ja säätää kapasiteettia automaattisesti ylläpitääkseen tasaisen ja ennustettavan suorituskyvyn mahdollisimman alhaisin kustannuksin. Auto Scaling toiminnon avulla voidaan määrittää sovellusten skaalaus useille palveluille. Toiminnolla voidaan asettaa tavoitekäyttötasoja useille resursseille yhdessä käyttöliittymässä. Kaikkien skaalautuvien resurssien keskimääräistä käyttöä voidaan tarkkailla ilman, että tarvitsee siirtyä muihin sovelluksiin. [84].

Auto Scalingin avulla voidaan rakentaa skaalaussuunnitelmia, joiden avulla voidaan automatisoida eri sovellusten resurssien käyttöä vastaamaan kysynnän muutoksiin. Auto Scaling luo automaattisesti kaikki skaalauskäytännöt ja asettaa tavoitteet tarpeiden perusteella. Auto Scaling tarkkailee sovellusta ja lisää tai poistaa automaattisesti kapasiteettia reaaliajassa tarpeiden muuttuessa. [84].

Google Cloudissa Managed instance groupit (MIG) tarjoavat automaattisen skaalauksen ominaisuuksia, joiden avulla voidaan automaattisesti lisätä tai poistaa virtuaalikoneita MIG:sta kuormituksen lisääntymisen tai vähenemisen perusteella. Automaattinen skaalaus auttaa sovelluksia käsittelemään sulavasti lisääntyvää liikennettä ja vähentämään kustannuksia, kun resurssien tarve on pienempi. Skaalauskäytäntö voidaan määrittää ja skaalaus suorittaa mitatun kuormituksen ja määriteltyjen asetusten perusteella. [85].

Automaattinen skaalaus toimii lisäämällä virtuaalikoneita MIG:hin, kun kuormitusta on enemmän, ja poistamalla niitä, kun virtuaalikoneiden tarve vähenee. Automaattinen skaalaus toimii ainoastaan MIG:ssa. MIG on kokoelma virtuaalikoneita, jotka on luotu yhteisestä instanssista. Automaattinen skaalaaja lisää tai poistaa virtuaalikoneita hallitusti automaattisen skaalauksen asetusten perusteella. [85].

Kun MIG:lle määritellään automaattisen skaalauksen asetukset, määritetään yksi tai useampi metriikka, joita automaattinen skaalaus käyttää MIG:n skaalaamiseen. Kun asetetaan useita metriikoita, automaattinen skaalaaja laskee virtuaalikoneiden suositellun määrän kullekin metriikalle ja asettaa MIG:lle virtuaalikoneiden määräksi suurimman lasketun arvon. [85].

4.5 Ylläpidettävyys

Ylläpidettävyys liittyy saatavuuteen kuvaamalla, miten häiriöajat alkavat ja miten ne ratkaistaan. Katkosaikoja aiheuttavan tapahtuman sattuessa huollettavat palvelut voidaan korjata nopeasti. Mitä nopeammin tapaus korjataan, sitä nopeammin palvelu tulee jälleen saataville. [70].

Ylläpidettävyydessä on kaksi pääkomponenttia: ennakoiva ja reaktiivinen.

- Ennakoiva ylläpito edellyttää koodikannan rakentamista, jota voidaan helposti ymmärtää ja muuttaa. Kehityksen edetessä ongelmia syntyy yhteensopimattomuudesta olemassa olevan koodin kanssa. Jos ohjelmoijat kirjoittavat "spagettikoodia" ylläpidettävyyden priorisoinnin sijaan, ongelmia ilmenee todennäköisesti ja niitä on vaikea löytää ja ratkaista. Ennakoivaan ylläpitoon kuuluu myös toimenpiteitä, kuten laadunvarmistus ja testaus. [70].
- Reaktiivinen ylläpidettävyys kuvaa palvelun korjattavuutta häiriötilanteiden jälkeen. Tähän vaikuttavat palvelun reagointimenettelyt häiriötilanteissa. Koska

> häiriötilanteet ovat väistämättömiä, hyvä reagointi tapauksiin ja riittävät suojamekanismit ovat välttämättömyys. Jos tapausten reagointimenetelmät ovat luotettavia, ongelmat voidaan ratkaista nopeasti. Asianmukainen reagointi edistävää myös oppimista ja vähentämään vian toistumisen todennäköisyyttä. Hyvin ylläpidettävä palvelu antaa ohjelmoijille mahdollisuuden toteuttaa reagointimenetelmiä tehokkaasti. [70].

Ylläpidettävyys näkyy käytettävyysmittareissa. Häiriöiden pituuden tai taajuuden lyhentäminen johtaa parempaan käytettävyyteen. Ylläpidettävyys ei kuitenkaan ole vain käytettävyyden päämäärä. Tällainen lähestymistapa voi johtaa huonosti kohdennettuihin kehitysresursseihin. Ylläpidettävyyteen investoiminen ei välttämättä heti johda parempaan käytettävyyteen. Kun vanhaa koodia korjataan teknisen velan kuittaamiseksi, palvelu toimii samalla tavalla kuin ennenkin säilyttäen käytettävyytensä. Vasta häiriöiden sattuessa paremman ylläpidettävyyden edut konkretisoituvat. Ylläpidettävyyttä tulee ajatella investointina luotettavuuteen, ei pelkkänä käytettävyyden osana. [70].

Laajamittaiset hajautetut järjestelmät, kuten pilvi-infrastruktuurit, ovat luonnostaan epäluotettavia. Azure Service Fabricin avulla kehittäjät voivat luoda luotettavia hajautettuja palveluita epäluotettavan infrastruktuurin päälle. Luodakseen vankkoja hajautettuja palveluita epäluotettavan infrastruktuurin päälle kehittäjien on voitava testata palveluiden vakautta samalla kun taustalla oleva epäluotettava infrastruktuuri käy läpi monimutkaisia tilasiirtymiä vikojen vuoksi. [79].

Azuren Fault Analysis Service antaa kehittäjille mahdollisuuden luoda vikoja ja testata palveluita. Nämä kohdistetut simuloidut viat, kuten jonkin osa-alueen uudelleenkäynnistys, voivat auttaa löytämään yleisimpiä vikoja. Kohdistetut simuloidut viat ovat kuitenkin määritelmän mukaan vääristyneitä eli keinotekoisia, joten ne voivat jättää huomiotta vikoja, jotka näkyvät vain vaikeasti ennustettavissa, pitkissä ja monimutkaisissa tilasiirtymien sarjoissa. Puolueettomaan testaukseen voidaan käyttää Chaosta. Chaos simuloi jaksoittaisia, lomitettuja vikoja koko klusterissa pitkien ajanjaksojen aikana. Simulointi koostuu joukosta Service Fabric API -kutsuja. [79].

Kun Chaosiin on määritetty vikojen määrä ja tyyppi, voidaan se käynnistää C#-, PowerShell- tai REST-sovellusliittymän kautta vikojen luomisen aloittamiseksi klusterissa ja palveluissa. Chaos voidaan määrittää toimimaan tietyn ajan, jonka jälkeen

se pysähtyy automaattisesti tai vaihtoehtoisesti voidaan kutsua StopChaos API:a (C#, PowerShell tai REST) Chaosin pysäyttämiseksi milloin tahansa. [79].

Chaoksen ollessa käynnissä se tuottaa erilaisia tapahtumia, jotka tallentavat ajon tilan kyseisellä hetkellä. Esimerkiksi ExecutingFaultsEvent sisältää kaikki viat, jotka Chaos on päättänyt suorittaa kyseisessä iteraatiossa. ValidationFailedEvent sisältää tiedot virheistä, joka löydettiin klusterin validoinnin aikana. GetChaosReport API:n (C#, PowerShell tai REST) kautta voidaan saada raportti Chaos-ajoista. Nämä tapahtumat säilytetään luotettavassa sanakirjassa, jonka typistyskäytäntö riippuu kahdesta asetuksesta: MaxStoredChaosEventCount (oletusarvo on 25000) ja StoredActionCleanupIntervalInSeconds (oletusarvo on 3600). Jokaisen StoredActionCleanupIntervalInSeconds-muuttujassa olevan sekuntimääreen jälkeen kaikki paitsi viimeisimmät MaxStoredChaosEventCount-tapahtumat poistetaan luotettavasta sanakirjasta. [79].

Chaos luo vikoja koko Service Fabric -klusteriin ja pakkaa kuukausien tai vuosien aikana havaitut viat muutamaan tuntiin. Limitettyjen vikojen yhdistäminen korkeaan vikatiheyteen löytää rajatapauksia, jotka muuten saattavat jäädä huomaamatta. Tämä johtaa palvelun koodin laadun merkittävään parantumiseen. [79].

AWS Cloud Development Kitin (CDK) avulla voidaan kuvata sovelluksen infrastruktuuria käyttämällä yleiskäyttöistä ohjelmointikieltä kuten TypeScriptiä, JavaScriptiä tai Pythonia. Näin ollen infrastruktuurille voidaan kirjoittaa testejä samalla tavalla kuten testejä kirjoitettaisiin sovelluksille. [80].

Infrastruktuuritestien kirjoittamismalli on hyvin samanlainen kuin testien luominen sovelluskoodille. Ainoa asia, joka eroaa normaaleista testeistä, ovat väitteet (assertions), jotka kirjoitetaan koodiin. TypeScript CDK:n mukana toimitetaan väitekirjasto, jonka avulla on helppo tehdä väitteitä infrastruktuurista. Väitekirjasto on tällä hetkellä vain TypeScript- ja JavaScript-käyttäjien saatavilla, mutta se on tulossa myöhemmin myös muiden kielten käyttäjien saataville. AWS käyttää myös sisäisesti kyseistä kirjastoa testien kirjoittamiseen. [80].

5 JOHTOPÄÄTÖKSET

Tässä työssä tutkittiin ja löydettiin palvelinarkkitehtuurin keskeiset laatuattribuutit ja luotiin niiden pohjalta palvelinarkkitehtuurin laatumalli tukemaan palvelinarkkitehtuurin suunnittelua erityisesti pilvipalveluissa. Työssä perehdyttiin myös pilvipalveluihin ja siihen, miten pilvipalvelut tukevat palvelinarkkitehtuurin laatuattribuutteja. Tämän lisäksi tutustuttiin myös palvelinarkkitehtuurimalleihin ja pilvipalveluiden tarjoamiin mahdollisuuksiin niin palvelinarkkitehtuurin saralla kuin yleisestikin.

Pilvipalveluiden käytön yleistyessä ja niiden ominaisuusvalikoiman kasvaessa tarve hyvälle suunnittelulle kasvaa. Pilvipalvelut mahdollistavat monimutkaisten kokonaisuuksien rakentamisen. Monimutkaisuuden ja laajuuden kasvaessa myös mahdolliset vikatilanteet ja käyttöön vaikuttavat ohjelmointivirheet kasvavat.

Ohjelmistokehitys on nykyisellään kallista ja kehityksen hinta tulee tulevaisuudessa kasvamaan entisestään edellä mainituista syistä monimutkaisuuden ja laajuuden kasvaessa. Laajat ohjelmistojärjestelmät vaativat paitsi rahaa myös aikaa. Samalla asiakkaat ja käyttäjät asettavat ohjelmistojen käytölle korkeita vaatimustasoja, kuten korkean saatavuuden, käytettävyyden, käytön nopeuden ja tietoturvan. Tämä nostaa pilviarkkitehtuurin keskiöön.

Ilman toimivaa ja perusteltua pilviarkkitehtuuria yritykset menettävät paljon aikaa ja rahaa yrittäessään saada ohjelmistoja käyttäjien vaatimalle tasolle. Laadukas pilviarkkitehtuuri on yritykselle selkeä kilpailuetu ja tämän alan osaaminen työmarkkinoilla haluttua.

Perinteiset konesalit ovat viime vuosien aikana joutuneet enenevissä määrin pilvipalveluiden jalkoihin. Vaikka niillekin on edelleen kysyntää ja paikkansa palvelinratkaisuissa, niin pilvipalveluiden käytön helppous ja kustannustehokkuus varsinkin pk-yrityksille on kiistaton.

Palvelinarkkitehtuurin laatua on toistaiseksi tutkittu kovin niukasti verrattuna ohjelmistoarkkitehtuurin laatuun. Tämä on selkeä kehityskohde tulevaisuudessa pilvipalveluiden käytön lisääntyessä. Tarve osaamiselle ja ymmärrykselle ohjelmistokehityksessä on ilmeinen. Moni taho onkin viimevuosina alkanut kiinnittää enemmän huomiota asiaan ja tutkittua tietoa löytyy kaiken aikaa enemmän.

Työn tutkimuskysymykset olivat

1. Mitkä ovat olennaiset palvelinarkkitehtuurin laatuattribuutit?
2. Miten pilvipalvelut tukevat palvelinarkkitehtuurin laatuattribuuttien mukaista suunnittelua?

Vastauksena ensimmäiseen tutkimuskysymykseen kehitettiin palvelinarkkitehtuurin laatumalli, joka sisältää seuraavat olennaiset laatuattribuuttikategoriat: tietoturva, saatavuus, vikasietoisuus, skaalautuvuus ja ylläpidettävyys. Työssä käytetty palvelinarkkitehtuurin laatumalli johdettiin ohjelmistoarkkitehtuurin laatumalleista, palvelinarkkitehtuurin laatuattribuutit ohjelmistoarkkitehtuurin vastaavista. Tähän ratkaisuun päädyttiin pääosin siksi, että vastaavia palvelinarkkitehtuurin laatumalleja ei ollut tarjolla, kun puolestaan ohjelmistokehityksen saralta laatumalleja oli tarjolla useita ja tutkimustietoa vuosikymmenten edestä. Kuten todettua, palvelinarkkitehtuurin laatua on viimevuosina alettu tutkimaan enemmän, mutta vielä tätä työtä tehtäessä akateeminen tutkimus aiheesta oli vähäistä. Monet pilvipalveluita tarjoavista yrityksistä ovat tehneet laadukasta sisältöä aiheeseen liittyen, mutta se osoittautui kuitenkin liian hajanaiseksi ja pintapuoliseksi akateemiseen tutkimukseen verrattuna.

Vastauksena tutkimuskysymykseen kaksi tarkasteltiin kolmea suurta pilvipalvelujen tarjoajaa. Microsoft, Amazon ja Google ovat luoneet ominaisuuksiltaan laajat pilvipalvelut, jotka täyttävät jo nykyisellään palvelinarkkitehtuuriin kohdistuvat kovat vaatimukset. Tulevaisuudessa nämä palvelut eittämättä monipuolistuvat ja kehittyvät entisestään, joten laadukkaan palvelinarkkitehtuurin rakentaminen ja palvelinarkkitehtuurin laatumallin noudattaminen ei ole mahdoton tehtävä. Ohjelmistojen monimutkaistuessa tämä vaatii kuitenkin entistä syvempää osaamista.

Jatkotutkimusaiheena voisi toimia palvelinarkkitehtuurin laatumallin validointi käytännössä ja syvällisempi tutkimus aiheesta taikka laajempi otanta eri pilvipalveluista ja niiden tarjoamista mahdollisuuksista kuin myös pilvipalveluiden tarpeellisen kehityssuunnan tutkiminen, jotta palvelinarkkitehtuurin laatuattribuuttien kaltaista rakennetta pystyttäisiin tukemaan entistä paremmin.

LÄHTEET

[1] RedHat, “Understanding virtualization”, RedHat, 2021. [Online]. Saatavilla: https://www.redhat.com/en/topics/virtualization?intcmp=7016000000127cYAAQ. [Viitattu: 16.8.2021].

[2] S. Edwards, “Modern Server Technology Guide,” Medium, 5.11.2018. [Online], Saatavilla: https://medium.com/@sedwardscode/modern-server-technology-guide-8daabc217dfa. [Viitattu: 16.8.2021].

[3] J. A. McCall, P. K. Richards, & G. F. Walters, “Factors in Software Quality”. Nat'l Tech.Information Service, no. Vol. 1, 2 and 3, 1977.

[4] P. Berander, L. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö, & P. Tomaszewski, Software quality attributes and trade-offs. Blekinge Institute of Technology, 2005. Saatavilla: https://www.researchgate.net/profile/Jeanette-Eriksson/publication/238700270_Software_quality_attributes_and_trade-offs_Authors/links/543fa4550cf2f3e82851eef5/Software-quality-attributes-and-trade-offs-Authors.pdf.

[5] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod, & M. Merritt, Characteristics of Software Quality. North Holland, 1978.

[6] B. Boehm, W. Barry, J. R. Brown, & M. Lipow, “Quantitative evaluation of software quality”. International Conference on Software Engineering. Proceedings of the 2nd international conference on Software engineering. 1976.

[7] ISO/IEC 25010, ISO/IEC Standardi, 2011.

[8] American Society for Quality, “Software Quality,” American Society for Quality, 2021. [Online]. Saatavilla: https://asq.org/quality-resources/software-quality. [Viitattu: 16.8.2021].

[9] A. Montazerolghaem, M. H. Yaghmaee, & A. Leon-Garcia, "Green Cloud Multimedia Networking: NFV/SDN Based Energy-Efficient Resource Allocation", IEEE Transactions on Green Communications and Networking, vol. 4, issue 3, pp. 873-889, 2020.

[10] Amazon Web Services, “What is cloud computing?”, Amazon Web Services, 2021. [Online]. Saatavilla: https://aws.amazon.com/what-is-cloud-computing/. [Viitattu: 30.8.2021].

[11] S. Brown, “Cloud vs. Data Center: Which Is Right for Your Business?”, Rutter Networking Technologies, 25.2.2020. [Online], Saatavilla: https://www.rutter-net.com/blog/cloud-computing-vs.-data-center-which-is-right-for-your-business. [Viitattu: 30.8.2021].

[12] Salesforce.com, inc, “12 Benefits of Cloud Computing”, Salesforce.com, inc, 2021. [Online]. Saatavilla: https://www.salesforce.com/products/platform/best-practices/benefits-of-cloud-computing/. [Viitattu: 30.8.2021].

[13] IBM, “Benefits of Cloud Computing”, IBM, 2021. [Online]. Saatavilla: https://www.ibm.com/cloud/learn/benefits-of-cloud-computing. [Viitattu: 30.8.2021].

[14] Guru99, “Advantages and Disadvantages Of Cloud Computing”, Guru99, 2021. [Online]. Saatavilla: https://www.guru99.com/advantages-disadvantages-cloud-computing.html. [Viitattu: 30.8.2021].

[15] GlobalDots, “13 Key Cloud Computing Benefits for Your Business”, GlobalDots, 5.7.2018. [Online]. Saatavilla: https://www.globaldots.com/resources/blog/cloud-computing-benefits-7-key-advantages-for-your-business/. [Viitattu: 30.8.2021].

[16] JavaTpoint, “Advantages and Disadvantages of Cloud Computing”, JavaTpoint, 2021. [Online]. Saatavilla: https://www.javatpoint.com/advantages-and-disadvantages-of-cloud-computing. [Viitattu: 30.8.2021].

[17] The State of Queensland, “Benefits of cloud computing”, The State of Queensland, 21.7.2017. [Online]. Saatavilla: https://www.business.qld.gov.au/running-business/it/cloud-computing/benefits. [Viitattu: 30.8.2021].

[18] Sify Technologies Limited, “How Cloud Data Centers Differ from Traditional Data Centers”, Sify Technologies Limited, 2021. [Online]. Saatavilla: https://www.sifytechnologies.com/blog/cloud-datacenters-differ-traditional-datacenters/. [Viitattu: 30.8.2021].

[19] K. Robinson, “Data Center or Public Cloud—Which Is Right for You?”, Colocation America, 4.12.2019. [Online], Saatavilla: https://www.colocationamerica.com/blog/datacenter-or-public-cloud. [Viitattu: 30.8.2021].

[20] RapidScale, “Cloud Computing Stats - Security and Recovery”, RapidScale, 30.1.2015. [Online]. Saatavilla: https://www.slideshare.net/rapidscale/cloud-computing-stats-security-and-recovery. [Viitattu: 30.8.2021].

[21] M. Biddick, “Time To Think About Cloud Computing,” InformationWeek, 22.10.2008. [Online], Saatavilla: https://www.informationweek.com/cloud/software-as-a-service/time-to-think-about-cloud-computing/d/d-id/1073198. [Viitattu: 30.8.2021].

[22] Statista, “Number of smartphone users from 2016 to 2021”, Statista, 6.2021. [Online]. Saatavilla: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Viitattu: 30.8.2021].

[23] C. Boulton, “How Sunny Delight juices up sales with cloud-based analytics,” CIO, 14.9.2015. [Online], Saatavilla: https://www.cio.com/article/2983624/how-sunny-delight-juices-up-sales-with-cloud-based-analytics.html. [Viitattu: 30.8.2021].

[24] VMware, “MIT Executive Study Uncovers Top 10 Trends Shaping IT Resilience,” VMware, 11.2020. [Online], Saatavilla: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/microsites/cio-vantage/vmware-global-mit-executive-study.pdf. [Viitattu: 30.8.2021].

[25] Verizon, “State of the Market: Internet of Things 2017”, Verizon, 2017. [Online]. Saatavilla: https://www.verizon.com/about/sites/default/files/Verizon-2017-State-of-the-Market-IoT-Report.pdf. [Viitattu: 30.8.2021].

[26] Business wire, “Cloud Computing Could Cut Data Center Energy Consumption by Nearly One-Third by 2020, According to Pike Research”, Business wire, 20.9.2011. [Online]. Saatavilla: https://www.businesswire.com/news/home/20110920005075/en/Cloud-Computing-Could-Cut-Data-Center-Energy-Consumption-by-Nearly-One-Third-by-2020-According-to-Pike-Research. [Viitattu: 30.8.2021].

[27] M. Richman, “Cloud Computing’s Value Proposition | Maximizing Business Value with AWS,” A cloud guru, 7.3.2019. [Online], Saatavilla: https://acloudguru.com/blog/engineering/cloud-computings-value-proposition-maximizing-business-value-with-aws. [Viitattu: 30.8.2021].

[28] A. Larkin, “Disadvantages of Cloud Computing,” Cloud academy, 7.8.2019. [Online], Saatavilla: https://cloudacademy.com/blog/disadvantages-of-cloud-computing/. [Viitattu: 13.9.2021].

[29] R. Hersher, “Amazon And The $150 Million Typo,” NPR, 3.3.2017. [Online], Saatavilla: https://www.npr.org/sections/thetwo-way/2017/03/03/518322734/amazon-and-the-150-million-typo?t=1529930367722&t=1631547716941. [Viitattu: 13.9.2021].

[30] P. Venezia, “Murder in the Amazon cloud,” InfoWorld, 23.6.2014. [Online], Saatavilla: https://www.infoworld.com/article/2608076/murder-in-the-amazon-cloud.html. [Viitattu: 13.9.2021].

[31] S. Lavinski, “10 Disadvantages & Risks of Cloud Computing,” FAUN, 4.4.2019. [Online], Saatavilla: https://faun.pub/10-disadvantages-risks-of-cloud-computing-35111de75611. [Viitattu: 13.9.2021].

[32] RedHat, “What is cloud architecture?”, RedHat, 24.6.2019. [Online]. Saatavilla: https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-architecture. [Viitattu: 27.9.2021].

[33] VMware, “What is Cloud Architecture?,” VMware, 2021. [Online], Saatavilla: https://www.vmware.com/topics/glossary/content/cloud-architecture. [Viitattu: 27.9.2021].

[34] S. Watts, M. Raza, “SaaS vs PaaS vs IaaS: What’s The Difference & How To Choose,” bmc, 15.6.2019. [Online], Saatavilla: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/. [Viitattu: 27.9.2021].

[35] IBM Cloud Education, “IaaS vs. PaaS vs. SaaS”, IBM, 2.9.2021. [Online]. Saatavilla: https://www.ibm.com/cloud/learn/iaas-paas-saas. [Viitattu: 27.9.2021].

[36] StorageCraft, “Private, Public, Hybrid, or Multi-Cloud: What’s the Difference Between Them?,” StorageCraft, 25.2. [Online], Saatavilla: https://blog.storagecraft.com/private-public-hybrid-multi-cloud/. [Viitattu: 27.9.2021].

[37] R. Kilstad, “Which one should you choose? - Private, Public, Hybrid or Multi-Cloud?,” TietoEvry, 26.5.2021. [Online], Saatavilla: https://www.tietoevry.com/en/blog/2021/04/Which-one-should-you-choose-Private-Public-Hybrid-or-Multi-Cloud/. [Viitattu: 27.9.2021].

[38] J. Lewis, M. Fowler, “Microservices,” Martin Fowler, 25.3.2014. [Online], Saatavilla: https://martinfowler.com/articles/microservices.html. [Viitattu: 11.10.2021].

[39] IBM Cloud Education, “Microservices”, IBM, 30.3.2021. [Online]. Saatavilla: https://www.ibm.com/cloud/learn/microservices. [Viitattu: 11.10.2021].

[40] Microsoft, “Microservices architecture style”, Microsoft, 30.10.2019. [Online]. Saatavilla: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices. [Viitattu: 11.10.2021].

[41] T. Huston, “What is Microservices?,” SmartBear, 2021. [Online], Saatavilla: https://smartbear.com/solutions/microservices/. [Viitattu: 11.10.2021].

[42] M. Richards, Software Architecture Patterns. Sebastopol, CA: O’Reilly Media, Inc., 2015.

[43] RedHat, “What is event-driven architecture?”, RedHat, 27.9.2019. [Online]. Saatavilla: https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture. [Viitattu: 25.10.2021].

[44] Amazon Web Services, “What is an Event-Driven Architecture?”, Amazon Web Services, 2021. [Online]. Saatavilla: https://aws.amazon.com/event-driven-architecture/. [Viitattu: 25.10.2021].

[45] G. Jansen, J. Saladas, “Advantages of the event-driven architecture pattern,” IBM, 12.5.2021. [Online], Saatavilla: https://developer.ibm.com/articles/advantages-of-an-event-driven-architecture/. [Viitattu: 25.10.2021].

[46] RedHat, “What is Function-as-a-Service (FaaS)?”, RedHat, 3.1.2020. [Online]. Saatavilla: https://www.redhat.com/en/topics/cloud-native-apps/what-is-faas. [Viitattu: 25.10.2021].

[47] IBM Cloud Education, “FaaS (Function-as-a-Service)”, IBM, 30.7.2019. [Online]. Saatavilla: https://www.ibm.com/cloud/learn/faas. [Viitattu: 25.10.2021].

[48] Cloudflare, “What is BaaS? | Backend-as-a-Service vs. serverless”, Cloudflare, 2021. [Online]. Saatavilla: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/. [Viitattu: 25.10.2021].

[49] R. Saraswathi, “Four Architecture Choices for Application Development in the Digital Age,” IBM, 6.1.2020. [Online], Saatavilla: https://www.ibm.com/cloud/blog/four-architecture-choices-for-application-development. [Viitattu: 25.10.2021].

[50] Microsoft, “Saga distributed transactions”, Microsoft, 2021. [Online]. Saatavilla: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/saga/saga. [Viitattu: 25.10.2021].

[51] C. Richardson, “Pattern: Saga,” Microservices.io, 2021. [Online], Saatavilla: https://microservices.io/patterns/data/saga.html. [Viitattu: 25.10.2021].

[52] S. Dashora, “Microservices Using the Saga Pattern,” DZone, 2.5.2019. [Online], Saatavilla: https://dzone.com/articles/microservices-using-saga-pattern. [Viitattu: 8.11.2021].

[53] Cloud Native Computing Foundation: Cloud Native Computing Foundation (“CNCF”) Charter, The Linux Foundation, 6.11.2015. Saatavilla: https://github.com/cncf/foundation/blob/master/charter.md. [Viitattu: 8.11.2021].

[54] Microsoft, “What is Cloud Native?”, Microsoft, 27.10.2021. [Online]. Saatavilla: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/definition. [Viitattu: 8.11.2021].

[55] S. Patnaik, “Cloud Native Application Architecture,” Medium, 12.2.2019. [Online], Saatavilla: https://medium.com/walmartglobaltech/cloud-native-application-architecture-a84ddf378f82. [Viitattu: 8.11.2021].

[56] Weaveworks, “6 Essential Things You Need to Know About Cloud Native Applications”, Weaveworks, 2021. [Online]. Saatavilla: https://www.weave.works/technologies/going-cloud-native-6-essential-things-you-need-to-know/. [Viitattu: 8.11.2021].

[57] AppDynamics, “What is Cloud Native Architecture?”, AppDynamics, 2020. [Online]. Saatavilla: https://www.appdynamics.com/topics/what-is-cloud-native-architecture. [Viitattu: 8.11.2021].

[58] Information Technology Services, “ITS Technical Glossary”, Information Technology Services, 20.4.2001. [Online]. Saatavilla: https://web.archive.org/web/20070902151937/http://www.its.state.nc.us/Information/Glossary/Glossm.asp. [Viitattu: 22.11.2021].

[59] Sun Microsystem, “Distributed Application Architecture”, Sun Microsystem. [Online]. Saatavilla: https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf. [Viitattu: 22.11.2021].

[60] Techopedia, “Client/Server Architecture”, Techopedia, 25.8.2020. [Online]. Saatavilla: https://www.techopedia.com/definition/438/clientserver-architecture. [Viitattu: 22.11.2021].

[61] AO Kaspersky Lab, “What is Cloud Security?”, AO Kaspersky Lab, 2021. [Online]. Saatavilla: https://www.kaspersky.com/resource-center/definitions/what-is-cloud-security. [Viitattu: 6.12.2021].

[62] Azure, “Introduction to Azure security”, Azure, 2.12.2021. [Online]. Saatavilla: https://docs.microsoft.com/en-us/azure/security/fundamentals/overview. [Viitattu: 6.12.2021].

[63] Z. Flower, “Why is high availability important in cloud computing?,” TechTarget, 22.10.2020. [Online], Saatavilla: https://searchcloudcomputing.techtarget.com/answer/How-much-cloud-uptime-do-you-need. [Viitattu: 10.1.2022].

[64] T. Rodrigues, “What high availability for cloud services means in the real world,” TechRepublic, 20.11.2011. [Online], Saatavilla: https://www.techrepublic.com/blog/the-enterprise-cloud/what-high-availability-for-cloud-services-means-in-the-real-world/. [Viitattu: 10.1.2022].

[65] Imperva, “Fault Tolerance”, Imperva. [Online]. Saatavilla: https://www.imperva.com/learn/availability/fault-tolerance/. [Viitattu: 24.1.2022].

[66] J. Kozlowicz, “High Availability vs. Fault Tolerance vs. Disaster Recovery,” Lunavi, 10.1.2020. [Online], Saatavilla: https://www.lunavi.com/blog/high-availability-vs-fault-tolerance-vs-disaster-recovery. [Viitattu: 24.1.2022].

[67] Cloudzero, “What Is Cloud Scalability? 4 Benefits For Every Organization”, Cloudzero, 27.8.2021. [Online]. Saatavilla: https://www.cloudzero.com/blog/cloud-scalability. [Viitattu: 24.1.2022].

[68] A. Domichen, “Cloud Scalability Explained: Types, Benefits, and More,” VEXXHOST, 6.11.2020. [Online], Saatavilla: https://vexxhost.com/blog/cloud-scalability/. [Viitattu: 24.1.2022].

[69] VMware, “What is cloud scalability?”, VMware, 2022. [Online]. Saatavilla: https://www.vmware.com/topics/glossary/content/cloud-scalability.html. [Viitattu: 24.1.2022].

[70] H. Culver, “Availability, Maintainability, Reliability: What's the Difference?,” Dzone, 24.9.2020. [Online], Saatavilla: https://dzone.com/articles/availability-maintainability-reliability-whats-the. [Viitattu: 6.2.2022].

[71] B. Violino, “Cyber security in the public cloud,” InfoWorld, 27.9.2021. [Online], Saatavilla: https://www.infoworld.com/article/3634111/cyber-security-in-the-public-cloud.html. [Viitattu: 7.2.2022].

[72] AWS, “AWS Cloud Security”, AWS, 2022. [Online]. Saatavilla: https://aws.amazon.com/security/. [Viitattu: 7.2.2022].

[73] Google, “Trust and security”, Google, 2022. [Online]. Saatavilla: https://cloud.google.com/security. [Viitattu: 7.2.2022].

[74] Total Uptime, “Cloud Service Level Agreement Expectations”, Total Uptime, 2022. [Online]. Saatavilla: https://totaluptime.com/cloud-service-level-agreement-expectations/. [Viitattu: 21.2.2022].

[75] AWS, “Amazon Compute Service Level Agreement”, AWS, 24.8.2021. [Online]. Saatavilla: https://aws.amazon.com/compute/sla/. [Viitattu: 21.2.2022].

[76] Azure, “SLA for App Service”, Azure, 12.2021. [Online]. Saatavilla: https://azure.microsoft.com/en-us/support/legal/sla/app-service/v1_5/. [Viitattu: 21.2.2022].

[77] Google, “App Engine Service Level Agreement (SLA)”, Google, 30.1.2020. [Online]. Saatavilla: https://cloud.google.com/appengine/sla. [Viitattu: 21.2.2022].

[78] P. Stringfellow, “Public cloud: A key component in a disaster recovery plan,” ComputerWeekly, 3.7.2019. [Online], Saatavilla: https://www.computerweekly.com/feature/Public-cloud-A-key-component-in-a-disaster-recovery-plan. [Viitattu: 7.3.2022].

[79] Azure, “Induce controlled Chaos in Service Fabric clusters”, Azure, 24.2.2022. [Online]. Saatavilla: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-controlled-chaos. [Viitattu: 21.3.2022].

[80] R. Huijbers, “Testing infrastructure with the AWS Cloud Development Kit (CDK)”, AWS, 23.9.2019. [Online]. Saatavilla: https://aws.amazon.com/blogs/developer/testing-infrastructure-with-the-aws-cloud-development-kit-cdk/. [Viitattu: 21.3.2022].

[81] Azure, “SLA for Azure Site Recovery”, Azure, 5.2018. [Online]. Saatavilla: https://azure.microsoft.com/en-us/support/legal/sla/site-recovery/v1_2/. [Viitattu: 19.4.2022].

[82] AWS, “Disaster recovery options in the cloud”, AWS, 2022. [Online]. Saatavilla: https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-options-in-the-cloud.html. [Viitattu: 19.4.2022].

[83] Azure, “Azure Autoscale”, Azure, 2022. [Online]. Saatavilla: https://azure.microsoft.com/en-gb/features/autoscale/. [Viitattu: 27.4.2022].

[84] AWS, “AWS Auto Scaling”, AWS, 2022. [Online]. Saatavilla: https://aws.amazon.com/autoscaling/. [Viitattu: 27.4.2022].

[85] Google, “Autoscaling groups of instances”, Google, 22.4.2022. [Online]. Saatavilla: https://cloud.google.com/compute/docs/autoscaler. [Viitattu: 27.4.2022].