ArchiMate käsikirja

Malleja ja esimerkkejä

Dokumentin tiedot	
Tila	Valmis
Versio	1.0
Luontipvm	19.5.2019
Muutettu	28.5.2022
Tekijä(t)	Eero Hosiaisluoma (EHo)

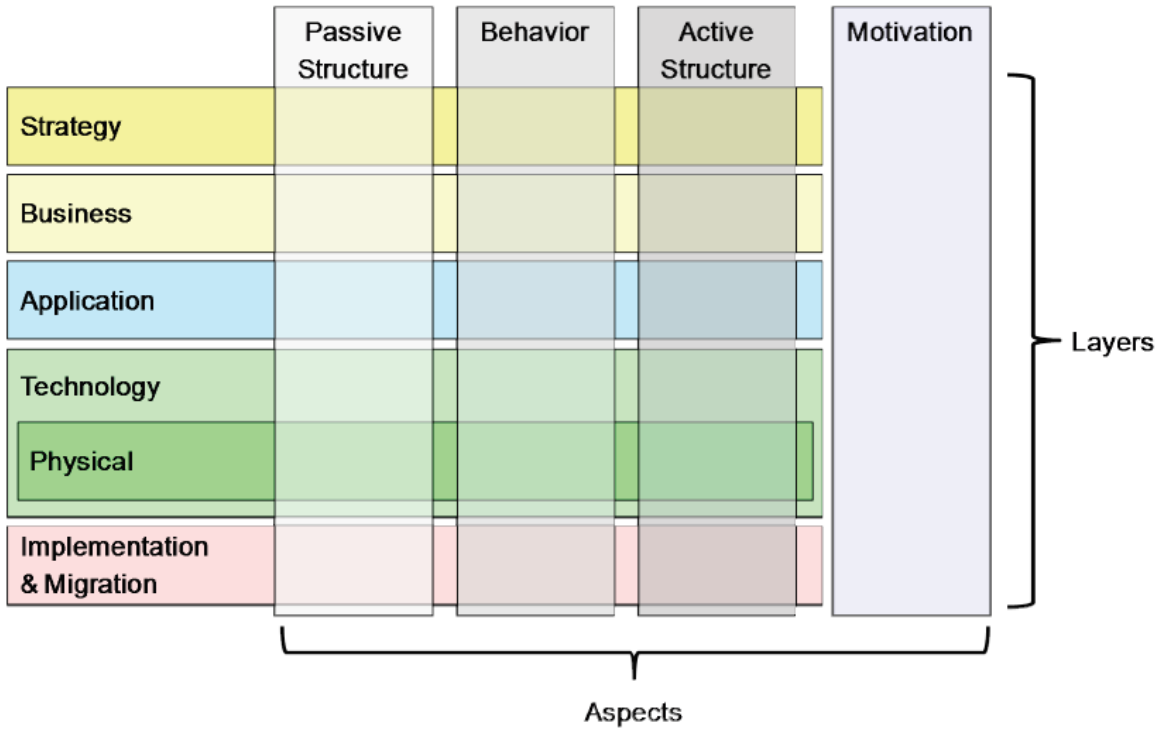
Sisällys

Johdanto

1.1 Dokumentin tavoite ja kohde

Tämä dokumentti sisältää ArchiMate-malleja ja esimerkkikaavioita. Osa malleista on ns. suunnittelumalleja. Kyseessä on itse asiassa ”Keittokirja”, ja esitetyt kaaviot ovat kuin ”reseptejä”, sillä on monta eri tapaa mallintaa asioita. Ei ole yhtä oikeata tapaa, vaan kaikki mallinnustavat ovat yhtä oikeita.

ArchiMate on laaja ja ilmaisuvoimainen notaatio, jossa on iso joukko elementtejä. Useimpiin mallinnustarpeisiin riittää kuitenkin osajoukko ArchiMate-elementeistä, ja vain muutama kaaviotyyppi. Tässä dokumentissa esitellään keskeiset kaaviotyypit, sekä ArchiMate-elementtien osajoukko. Elementit on ryhmitelty kerroksiin ArchiMate -viitekehyksen mukaisesti (kuva alla).



Kuva 1: ArchiMate Framework.

Tässä dokumentissa olevat kaaviot on mallinnettu ArchiMate 3.0.1 -standardia [1] noudattaen. Kaaviot on tehty QPR Enterprise Architect -välineellä (https://www.qpr.com/fi/ohjelmistot/qpr-enterprisearchitect). Lisää ArchiMate -esimerkkejä blogissa [5]. Dokumenttia saa vapaasti hyödyntää, kunhan lähde on mainittu. ☺

1.2 Viitteet

[1] ArchiMate 3.1, Open Group, 2019. https://pubs.opengroup.org/architecture/archimate3-doc/

[2] *Enterprise Architecture at Work*, 4th Edition, Marc Lankhorst et al., Springer, 2017.

[3] *Mastering ArchiMate*, Edition III, Gerben Wierda, 2017.

[4] *Lean Enterprise Architecture Method For Value Chain Based Development In Public Sector*, Hosiaisluoma et al, 2018. https://www.hosiaisluoma.fi/blog/lean-enterprise-architecture-method-for-value-chain-based-development-in-public-sector/

[5] ArchiMate Examples blog, Eero Hosiaisluoma. https://www.hosiaisluoma.fi/blog/archimate-examples/

[6] *ArchiMate Cookbook*, Eero Hosiaisluoma. https://www.hosiaisluoma.fi/blog/archimate-cookbook/

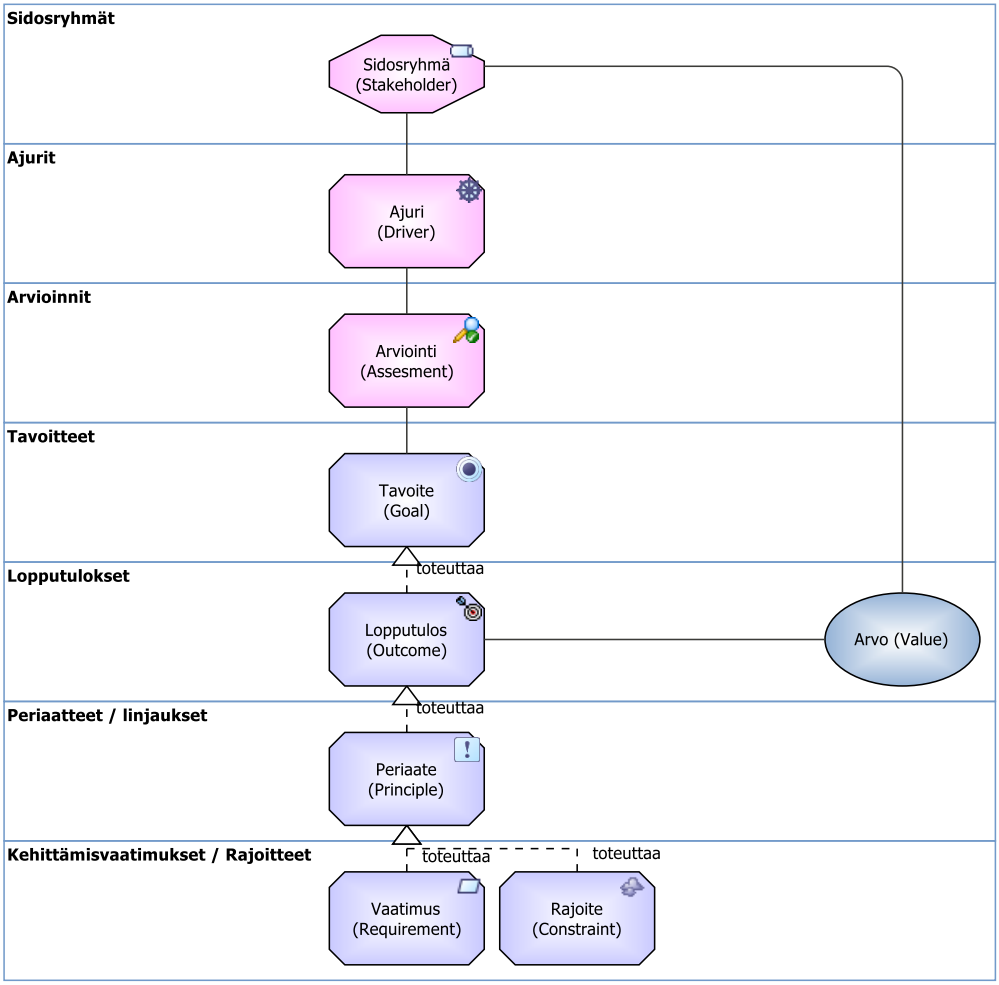
2. ArchiMate kaaviotyypit

2.1 Tavoitteet -näkymä

KENELLE

MIKSI

MITÄ



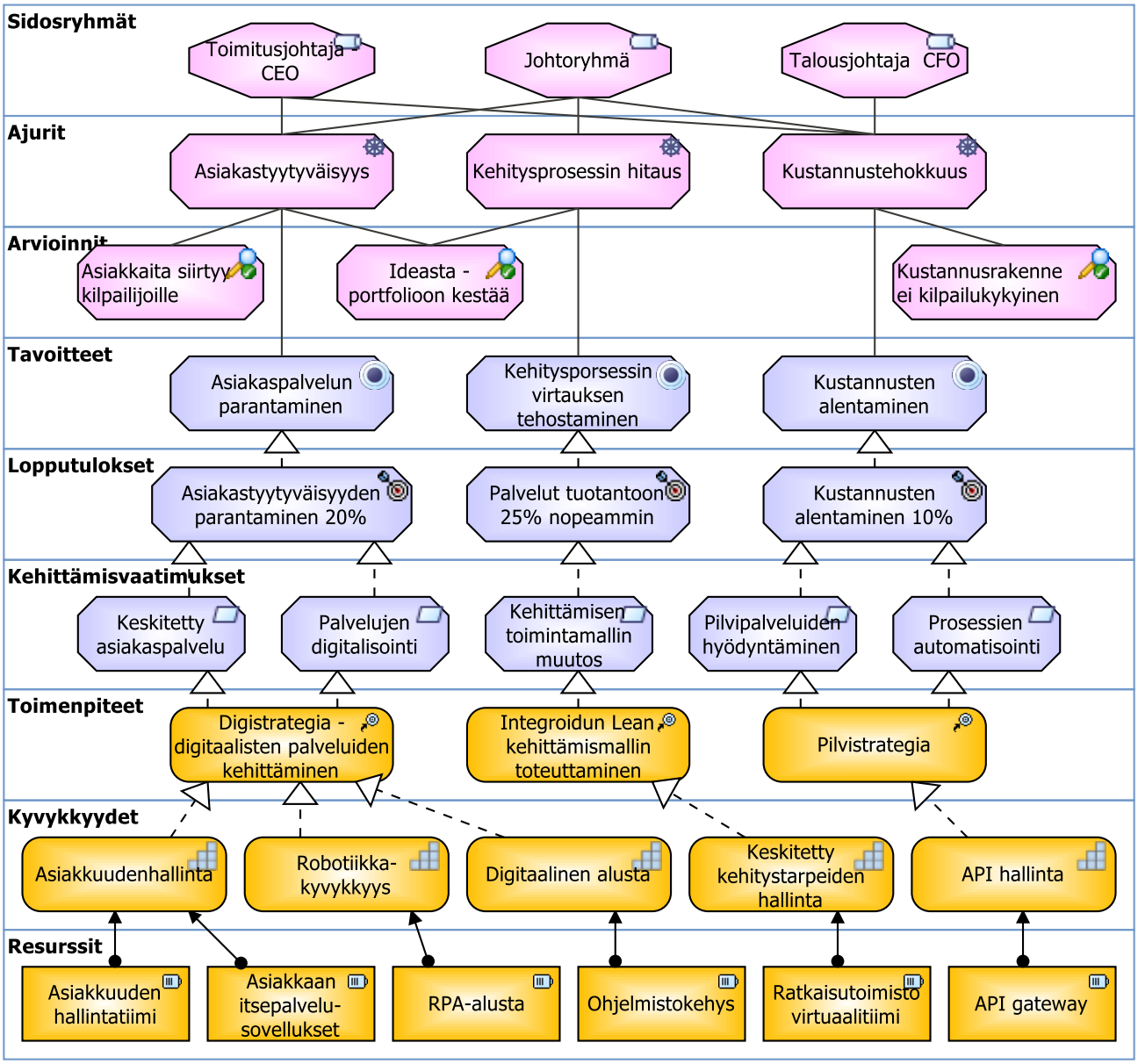
Kuva 2: Tavoitteet -näkymä suunnittelumalli.

Tavoitteet -näkymän avulla kuvataan, MIKSI kehittämiskohde on merkityksellinen, miksi muutos tarvitaan. Tavoitteet -näkymässä kuvataan keskeisimmät kehittämiskohteeseen liittyvät taustavaikuttimet (ajurit) ja tavoitteet. Tämän näkymän avulla pyritään vastaamaan kehittämiskohteen osalta kysymyksiin: KENELLE, MIKSI, MITÄ? Tavoitteet -näkymä toimii johdantona tarkempien rakenneosien tunnistamiseen ja kuvaamiseen.

Tavoitteet näkymässä käytetään ArchiMate Motivation- sekä Strategy-elementtejä.

Arvo (Value) voidaan kytkeä lopputulokseen osoittamaan sen konkreettista hyötyä tai tärkeyttä.

2.1.1 Tavoitteet -näkymä - esimerkki

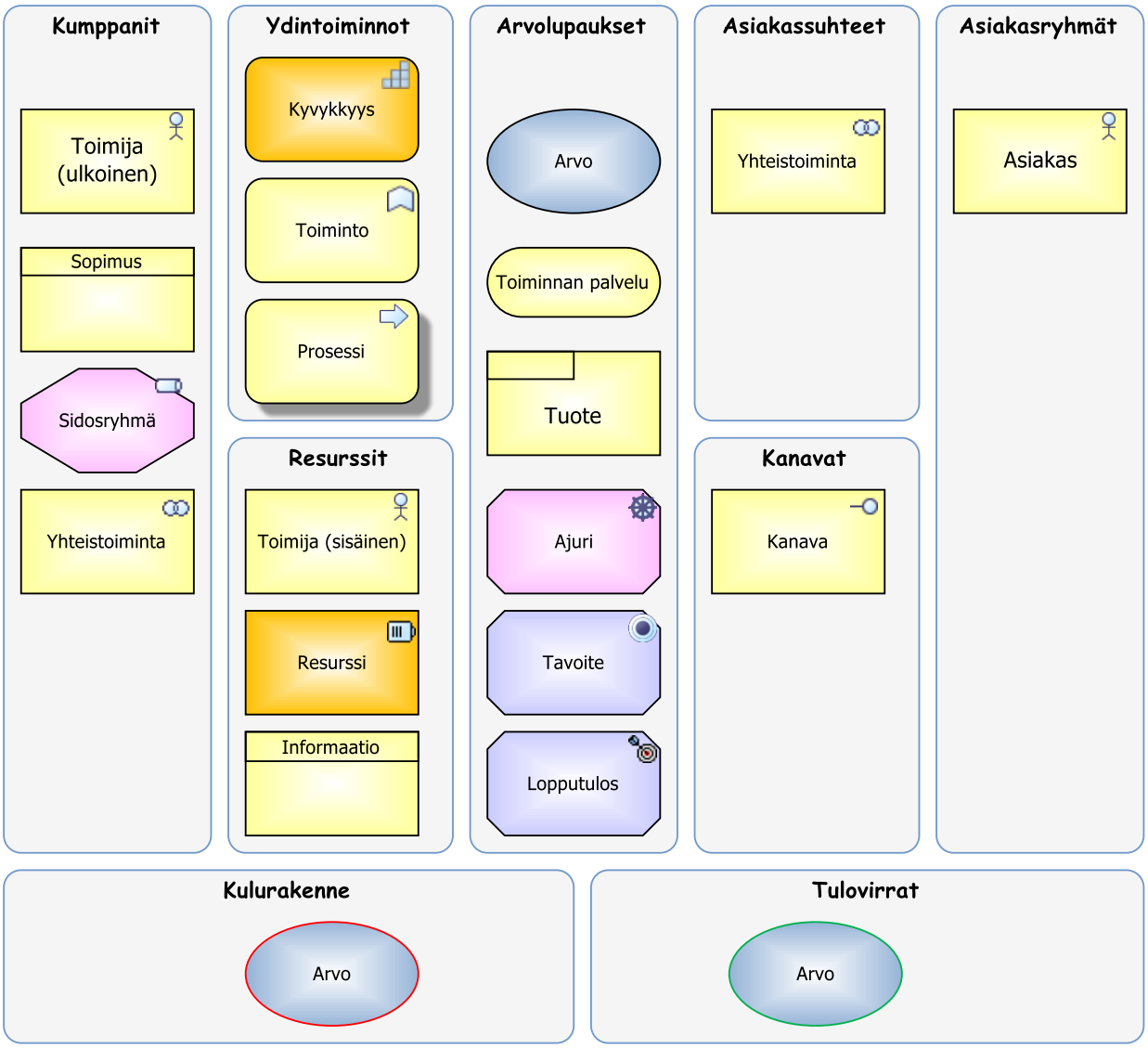


Kuva 3: Tavoitteet-näkymä -esimerkki.

Tavoitteet -näkymää voidaan soveltaa eri laajuisiin käyttötarkoituksiin: esimerkiksi koko organisaation strategian tai yksittäisen kehittämiskohteen kuvaamiseen. ArchiMate Motivation- ja Strategy -elementit ovat otsikkotasolla (kuva yllä) ymmärrettäviä, jolloin eri sidosryhmien (johto, toiminnan kehittäjät jne.) ei tarvitse tuntea ArchiMate-elementtejä tai niiden symboleita (ruori/Driver, maalitaulu/Goal jne.). Tässä mielessä Tavoitteet -näkymä (ja siinä käytetyt ArchiMate Motivation- ja Strategy -elementit) on yleiskäyttöinen, ja tällainen näkymä olisi hyödyllistä laatia kaikkiin muutostarpeisiin ja kehittämiskohteisiin.

2.2 Liiketoimintamalli -näkymä

2.2.1 Business Model Canvas (BMC)

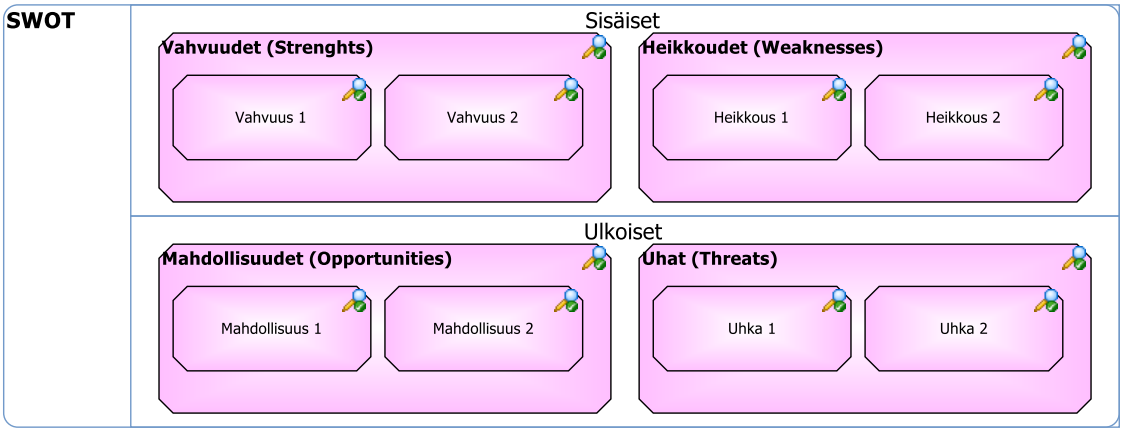


Kuva 4: Liiketoimintamalli - *Business Model Canvas* (BMC).

Liiketoimintamallin (Business Model) kuvaamisessa voidaan hyödyntää *Business Model Canvas (BMC)* menetelmää ja kuvaustapaa. ”Liiketoimintamalli kuvaa lyhyesti ja selkeästi, mitä organisaatio tekee ja millaisia palveluita se tuottaa antamansa arvolupauksen mukaisesti. Liiketoimintamallit vaativat toimiakseen useita organisaation kyvykkyyksiä, kuten henkilökunnan osaamista, toimivia prosesseja ja riittäviä resursseja.”

BMC-kaaviossa käytetään ensisijaisesti ArchiMate:n liiketoimintakerroksen (Business Layer) elementtejä, sekä myös Motivation ja Strategy -elementtjä.

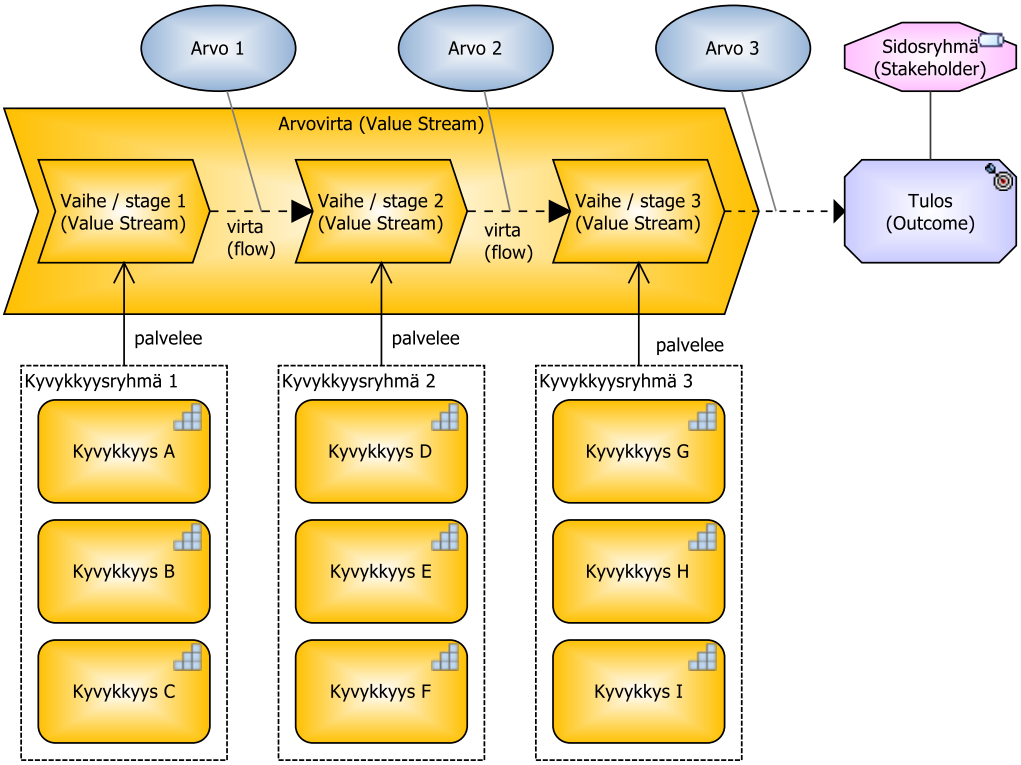
2.2.2 SWOT analyysi -näkymä



Kuva 5: Kehitettävän kohteen SWOT-analyysi.

Kehittämiskohdetta voidaan arvioida SWOT-analyysin (Strengths-Weaknesses-Opportunities-Threats) avulla. Tässä kaaviotyypissä käytetään ArchiMate:n *arviointi* (Assesment) -elementtiä.

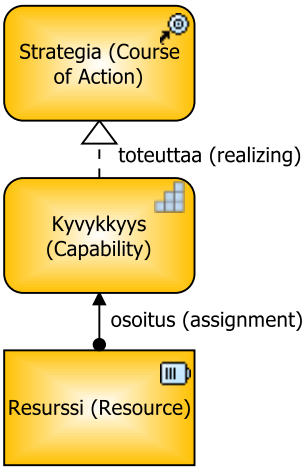
2.2.3 Arvovirta -näkymä



Kuva 6: Arvovirta (Value Stream).

Arvovirran (Value Stream) avulla voidaan määritellä liiketoimintamallin (Business Model) mukainen kuvaus siitä, MITEN esimerkiksi asiakkaalle tuotetaan arvoa. Arvoketjumallinnus havainnollistaa myös, kuinka kyvykkyydet kytkeytyvät arvovirtaan. Ts. mikä on kyvykkyyksien rooli ja merkitys kokonaisuudessa, mitä lisäarvoa ne tuottavat. Kaikkia kyvykkyyksiä ja niihin liittyviä resursseja ei aina voida kohdistaa arvovirtaan selkeästi. Tässä mielessä arvoketju ”paljastaa” arvoa tuottavat ja tuottamattomat kyvykkyydet. Arvovirta voidaan mallintaa tarkemmin prosessina, joka kuvaa toimintamallia (Operating Model). Tällöin arvovirta ja prosessi kuvaavat samaa asiaa, mutta eri abstraktiotasoilla. Arvovirtaa mallinnetaan ArchiMate:n *arvovirta* (Value Stream) -elementillä (löytyy ArchiMate 3.1 -versiosta [1]).

2.2.4 Strategia- ja kyvykkyys -näkymä



Kuva 7: Strategia suunnittelumalli.

Organisaation kehittämisen kannalta on oleellista, että strategia ja strategiset tavoitteet saadaan kytkettyä liiketoimintamalliin (Business Model), operatiivisen toiminnan toimintamalliin (Operating Model), sekä linkitettyä mahdollisuuksien mukaan myös kaikkiin kehittämiskohteisiin.

Strategia voidaan mallintaa ArchiMate Strategy- elementeillä: Course of Action (toimenpide), Capability (kyvykkyys) ja Resource (resurssi). Näiden elementtien avulla voidaan tarkastella organisaatiota *Resource Based View (RBV)* -tyyppisen jäsentelyn mukaisesti.

Kyvykkyyskartta (Capability Model) kannattaa laatia, jotta voidaan tunnistaa:

1) organisaation strategiset **ydinkyvykkyydet**, jotka muodostavat organisaation olemassaolon ja toiminnan perustan, tai jotka tuottavat kilpailuetua, sekä
2) organisaation päivittäisen toiminnan kannalta oleelliset **peruskyvykkyydet**.

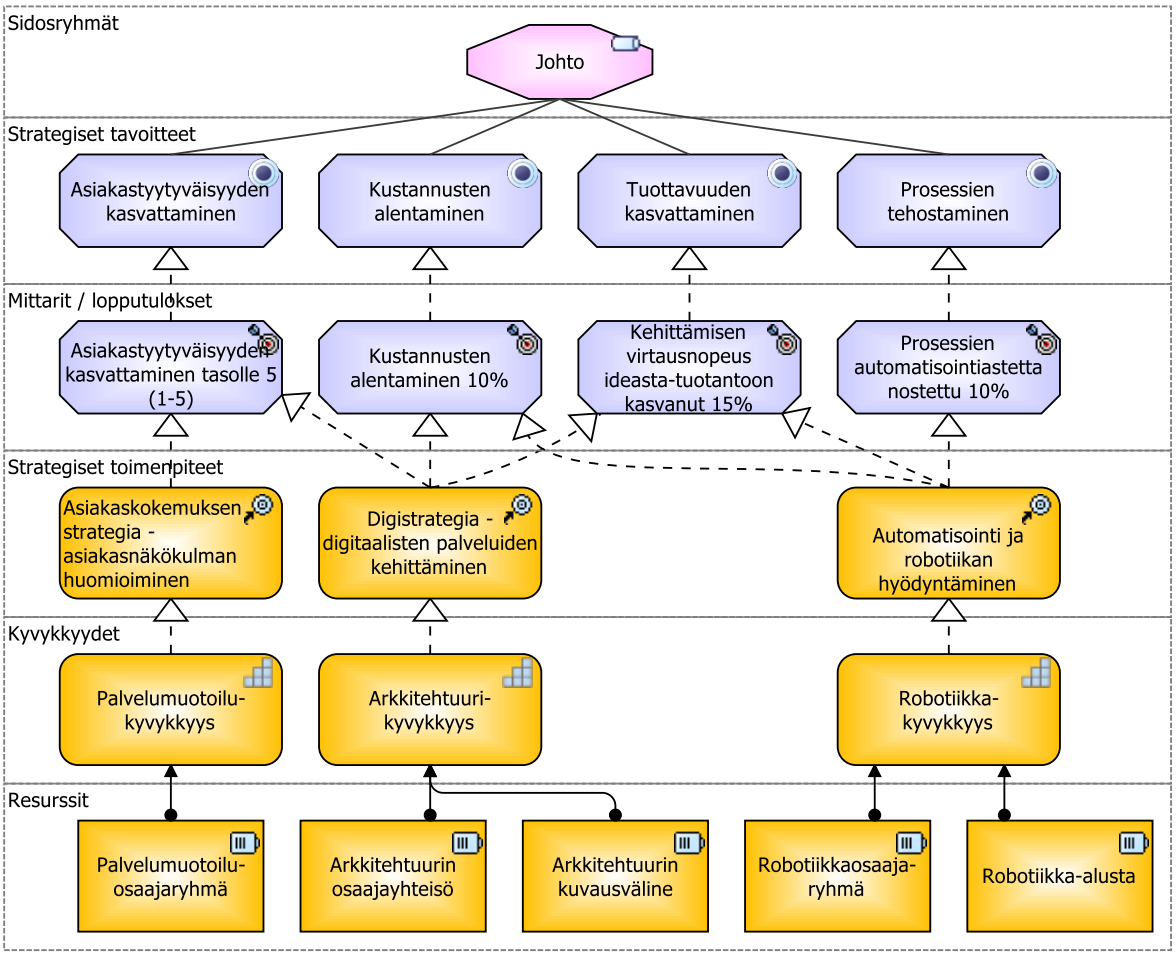
Kyvykkyyksien tunnistamiseksi voidaan huomioida seuraavaa:

- Kyvykkyys määrittelee MITÄ organisaatio tekee (kun taas resurssit määrittelevät MITEN ja MILLÄ)
- Kyvykkyys on yksikäsitteinen (ei ole päällekkäisyyksiä), ja verrattain pysyvä luonteeltaan
- Kyvykkyys voi jakaantua tarkempiin, alemman tason kyvykkyyksiin
- Kyvykkyys voi olla
    a. *Organisationaalinen* (aineeton, organisaation ylätason merkitykseen, arvonluontiin ja olemassaoloon liittyvä, strategiaan tai liiketoimintamalliin kytkettävissä oleva) tai
    b. *Operatiivinen* (aineellisten tai aineettomien resurssien tuottama, toiminnan pyörittämiseen eli toimintamalliin liittyvä).

Kuva 8: Kyvykkyyskartta.

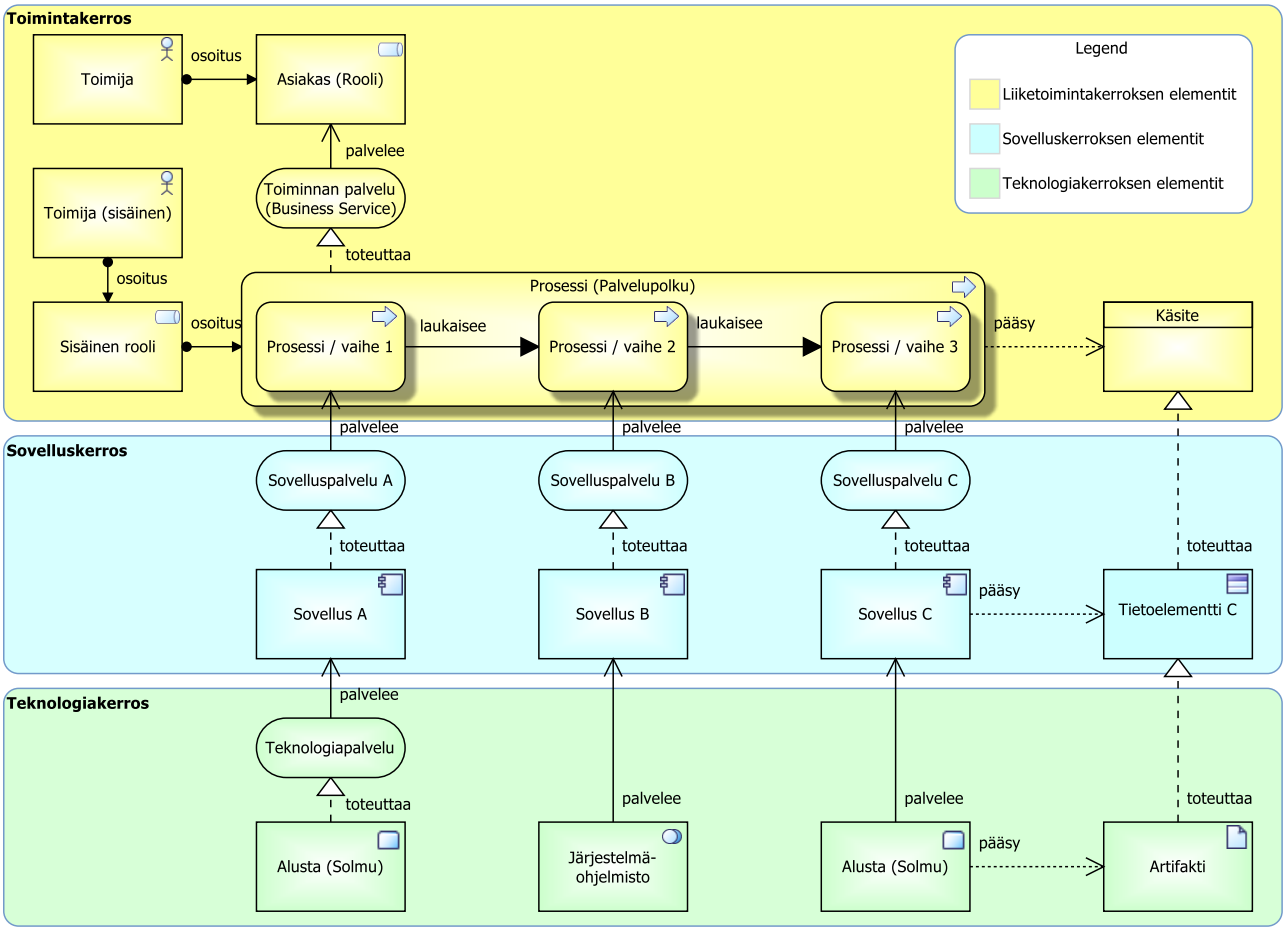
Kyvykkyyskartta voidaan mallintaa ArchiMate Capability- ja Grouping -elementeillä. Kuvassa yllä on käytetty Capability -elementtiä myös kyvykkyyksien ryhmittelyyn.

Strategian mallintamisessa yhdistetään ArchiMate Motivation- ja -Strategy-elementtejä (alla esimerkki). Näitä elementtejä voidaan käyttää myös ns. kyvykkyyspohjaisessa suunnittelussa (*Capability-Based Planning, CBP*).



Kuva 9: Strategia -esimerkki (ref. "Strategy to Capability" Value Stream).

2.3 Kerrosnäkymä



Kuva 10: Kerrosnäkymä (Layered View) - perusmalli.

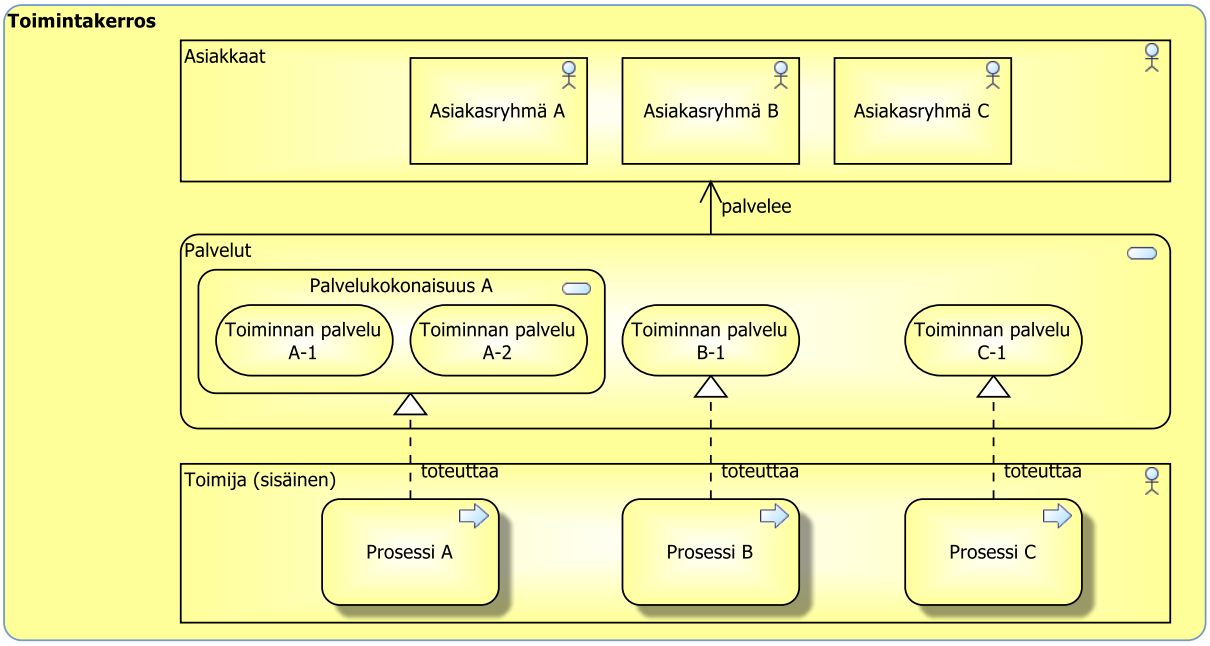
Kerrosnäkymän avulla kehittämiskohdetta voidaan tarkastella kerroksellisesti: ylimpänä esitetään liiketoimintaa, sen alapuolella loogisia sovellustason elementtejä, ja alimpana teknologiatason elementtejä. Kerrosnäkymällä voidaan esittää kokonaiskuva, joka on kehittämiskohteen kannalta kiinnostava ja merkityksellinen. Kerrosnäkymän avulla voidaan tunnistaa ja kuvata tärkeimmät osatekijät, jotka liittyvät kehittämiskohteen toimintaan ja rakenteeseen. Nämä toiminnalliset ja rakenteelliset osatekijät ovat esimerkiksi seuraavia: asiakkaat, joille tarjotaan liiketoiminnan palveluita, joita tuotetaan määrätyillä liiketoimintaprosesseilla, joita tuetaan sovelluspalveluilla, joita tarjoavat määrätyt sovellukset (eli järjestelmät), jotka on toteutettu määrätyillä teknologiapalveluilla ja alustoilla. Näistä muodostuvat kerrokset: toimintakerros, sovelluskerros ja teknologiakerros. Nämä kerrokset ovat peräisin ArchiMate-standardin kerroksista: Business Layer, Application Layer ja Technology Layer. Kerroksia kytketään yhteen palveluiden avulla. Tässä mielessä kyseessä on ”kerroksellinen ja palvelulähtöinen lähestymistapa” kehittämiskohteen tarkasteluun.

Kerrosnäkymän avulla voidaan havainnollistaa kehittämiskohteen toiminnallisten ja rakenteellisten osien väliset riippuvuudet. Voidaan esimerkiksi jäljittää, millä teknologia-alustalla tuotannossa on asennettuna järjestelmä, joka tarjoaa määrättyjä sovelluspalveluita prosessien tueksi. Edelleen, mitkä sisäiset toimijat on kiinnitetty ko. prosessiin, jolla tuotetaan määrättyjä toiminnan palveluita asiakkaille.

Kerrosnäkymissä voidaan hyödyntää kaikkien ArchiMate kerrosten elementtejä, ja kerrosnäkymän avulla näitä eri kerrosten elementtejä kytketään toisiinsa. Tämä kaaviotyyppi onkin kenties hyödyllisin ja informatiivisin kaaviotyyppi, sillä kerrosnäkymän avulla visualisoidaan eri kerrosten elementtien väliset suhteet, ns. alhaalta-ylös ja ylhäältä alas. Näin voidaan esittää samassa kaaviossa elementtien välisiä rakenteellisia- ja dynaamisia suhteita, sekä riippuvuussuhteita. Näiden osoittaminen onkin eräs arkkitehtuurin tärkeimmistä ominaispiirteistä.

Kerrosnäkymästä voidaan laatia erilaisia versioita, esimerkiksi kuvaamalla vain toiminta- ja sovelluskerrosten elementtien välisiä riippuvuuksia. Alla erilaisia variaatioita kerrosnäkymän soveltamisesta.

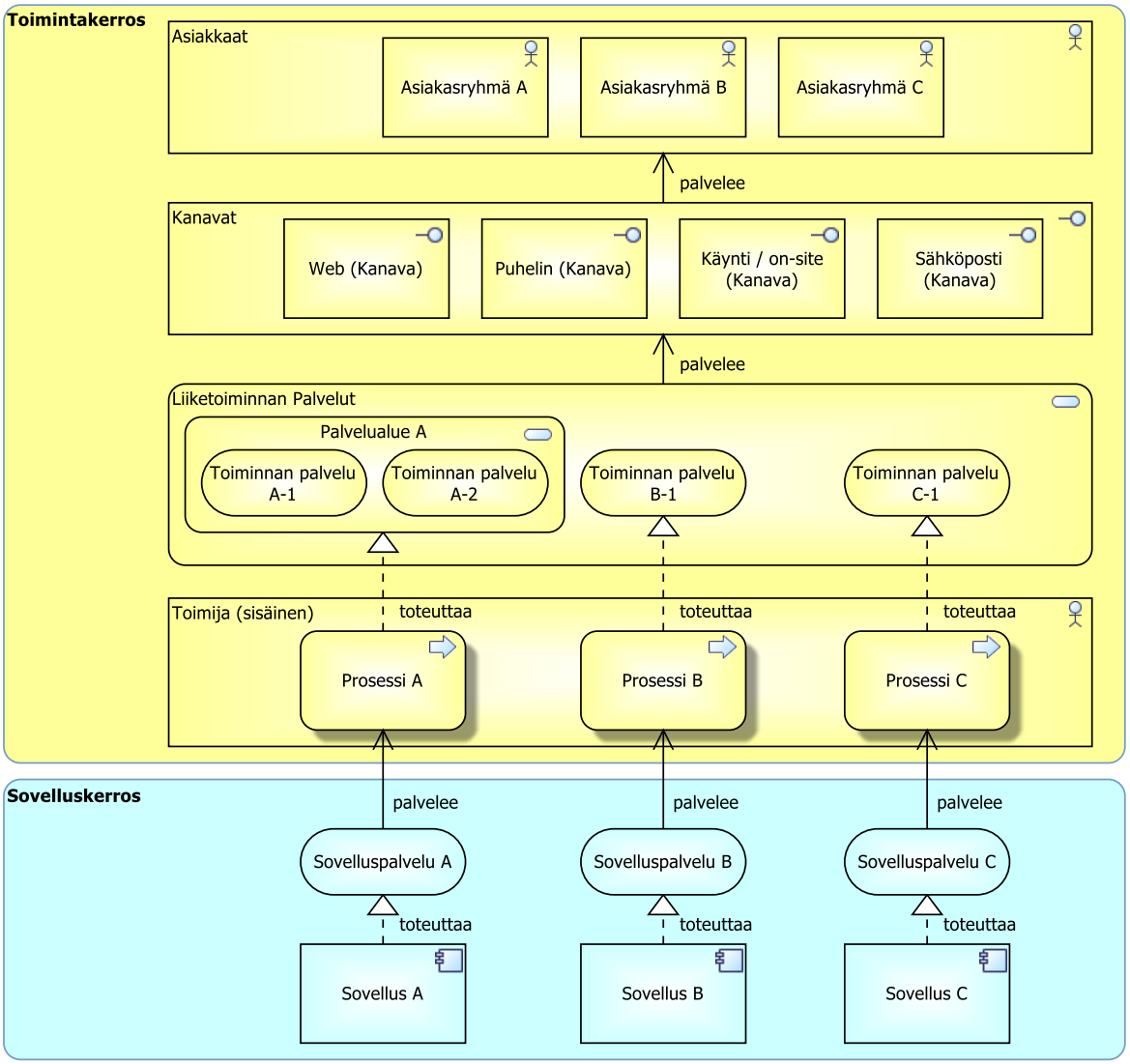
2.3.1 Kerrosnäkymä - Liiketoimintakerros



Kuva 11: Kerrosnäkymä - Liiketoimintakerros (kerrosnäkymän sovellutus).

Kerrosnäkymää voidaan soveltaa, esim. jättämällä kerroksia pois, ja tarkastella kehittämiskohdetta määrätystä näkökulmasta. Esimerkiksi toiminnan näkökulmasta, kuten tässä (kuva 11 yllä). Tämä kaavio noudattaa silti edelleen kerrosnäkymän lähestymistapaa, jossa asiakkaat ovat aina ylimpänä, sitten niille tarjottavat toiminnan palvelut, sitten prosessit ja sisäiset toimijat jne. Tässä mallissa määrätty yksikkö tuottaa prosesseillaan liiketoiminnan palveluita määrätyille asiakasryhmille.

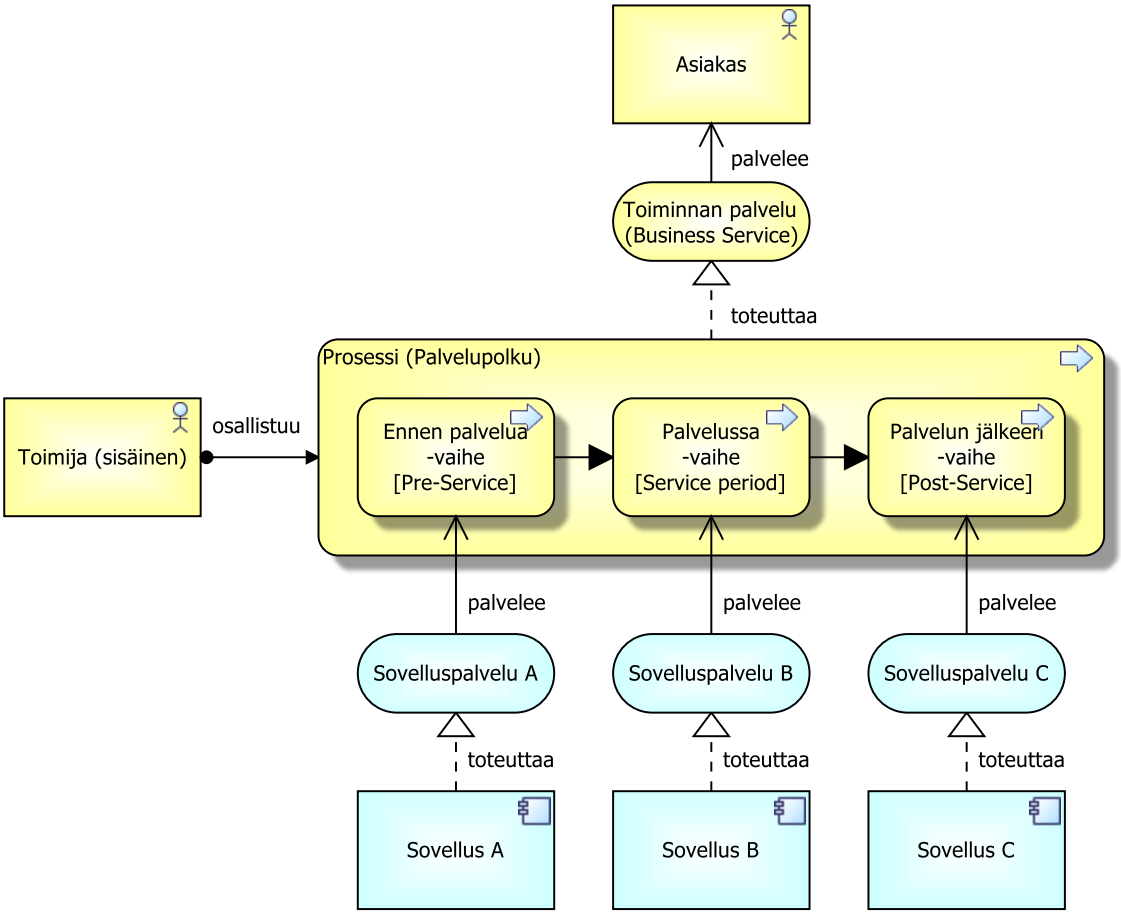
2.3.2 Kerrosnäkymä - Liiketoiminta- ja sovelluskerros



Kuva 12: Kerrosnäkymä - toiminta- ja sovelluskerrokset. Kanavat toimintakerroksessa.

Tässä versiossa edelliseen näkymään on lisätty kanavat, joilla voidaan havainnollistaa, minkä kanavien kautta näitä liiketoiminnan palveluita tarjotaan asiakkaille. Lisäksi tähän versioon on kytketty myös sovelluskerros, jolloin voidaan osoittaa mitkä järjestelmät tukevat toimintaa, ja minkä sovelluspalveluidensa kautta.

2.3.3 Kerrosnäkymä - Palvelupolku



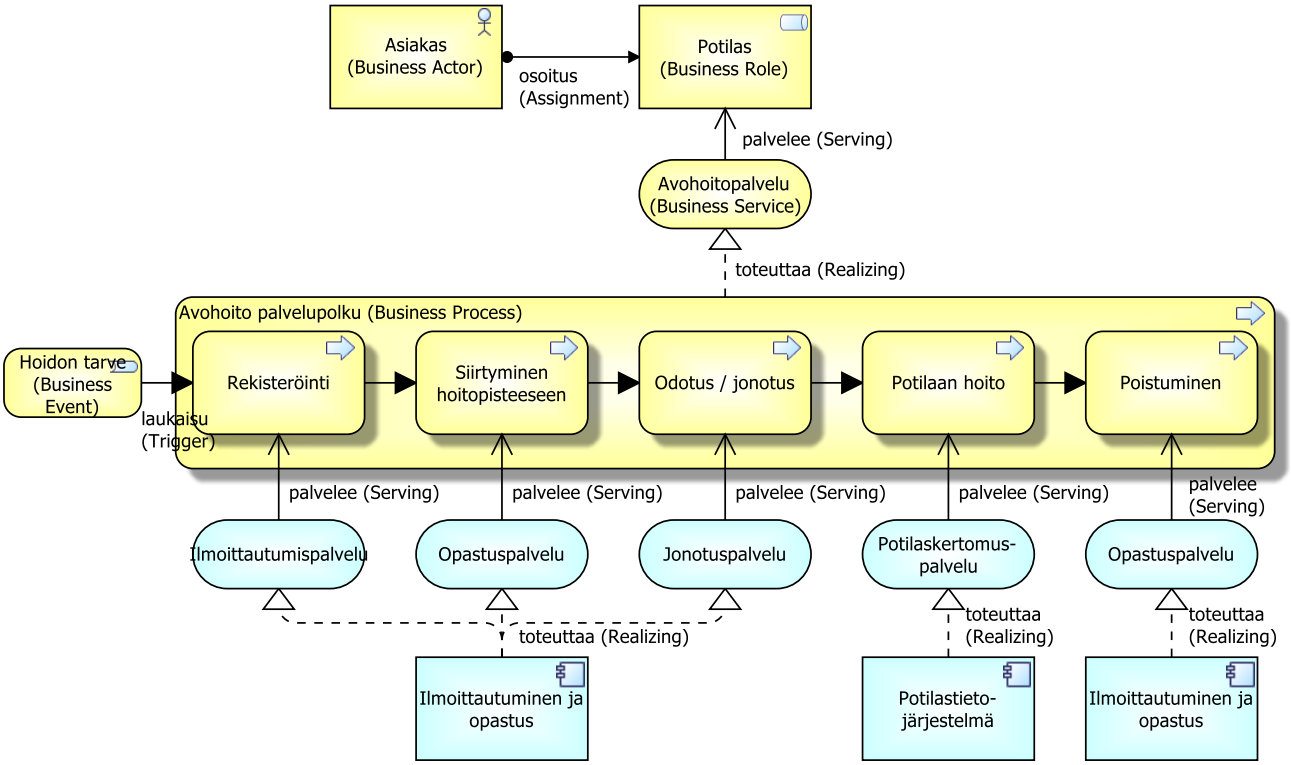
Kuva 13: Palvelupolku - suunnittelumalli.

Kerrosnäkymän avulla voidaan kuvata kehittämiskohdetta kerroksellisesti myös esimerkiksi asiakkaan näkökulmasta tarkasteltuna. Tällöin kerrosnäkymästä voidaan laatia esimerkiksi seuraavia näkymiä:

- Palvelupolku tai asiakaspolku (Customer Journey) ja
- Service Blueprint.

Nämä näkymät kytkevät asiakasnäkökulman (outside-in) organisaationäkökulmaan (inside-out).

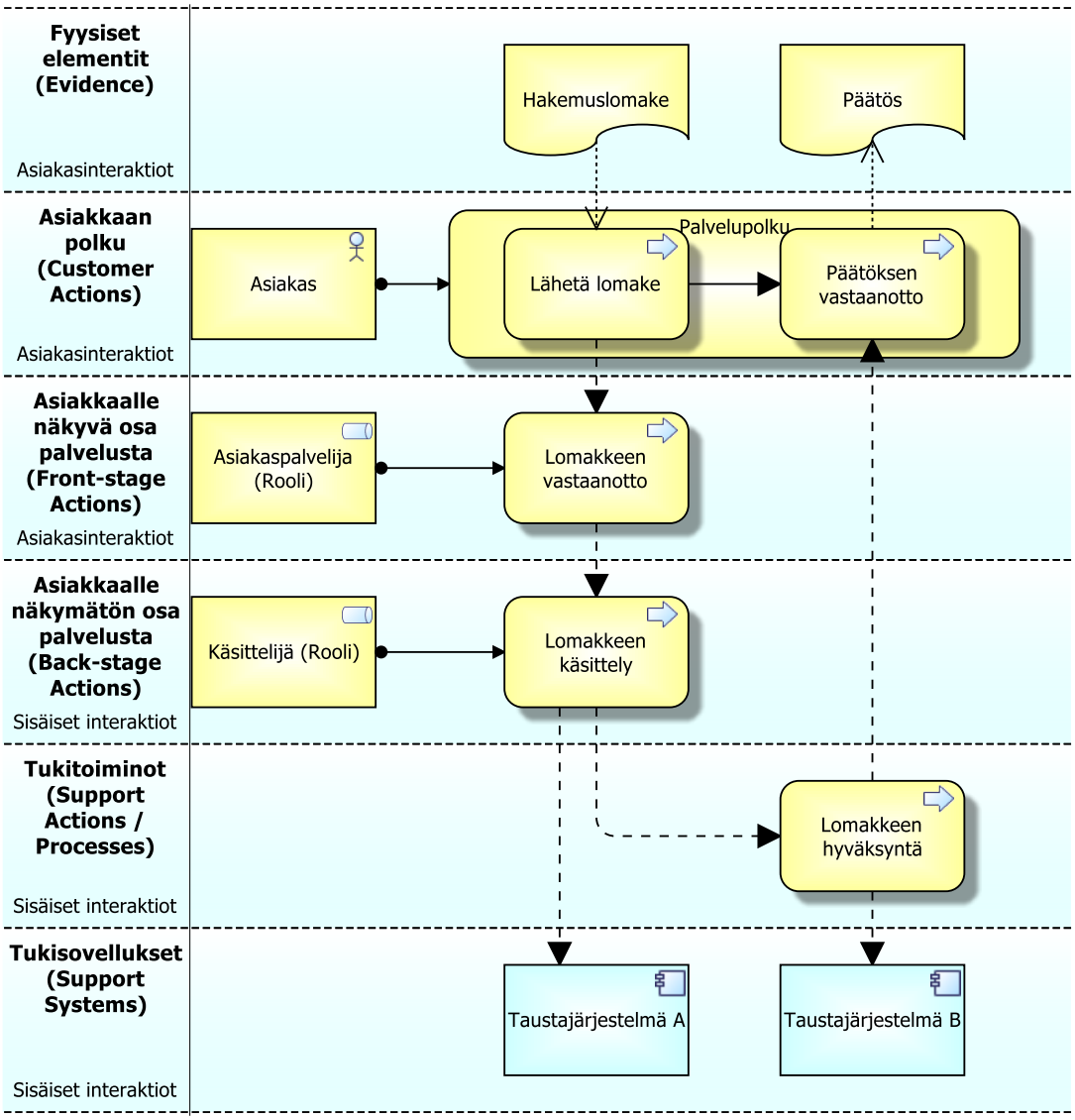
2.3.3.1 Kerrosnäkymä - Palvelupolku - esimerkki



Kuva 14: Palvelupolku - esimerkki.

Palvelupolku -näkymä on kerrosnäkymän erikoistapaus, joka yhdistelee toiminta- ja sovelluskerroksia.

2.3.4 Kerrosnäkymä - Service Blueprint - esimerkki



Kuva 15: Service Blueprint - esimerkki.

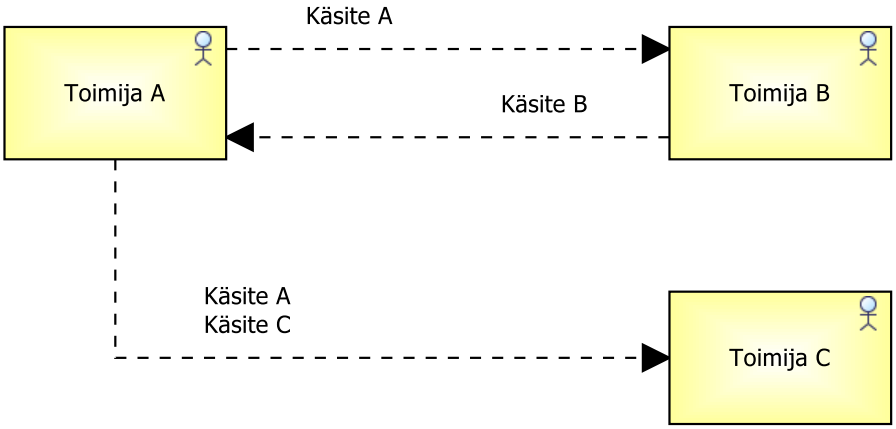
2.4 Vuorovaikutus -näkymä

Vuorovaikutusnäkymästä on kolme varianttia:

1) Toimijoiden vuorovaikutus,
2) Prosessien vuorovaikutus ja
3) Sovellusten vuorovaikutus.

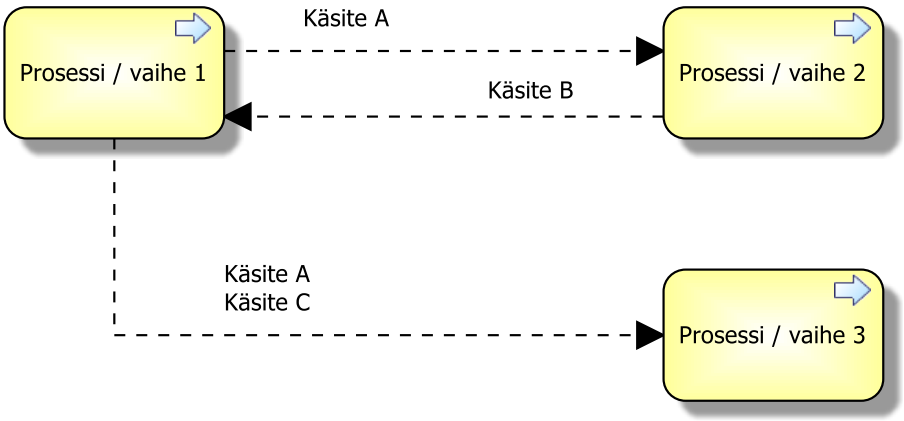
Vuorovaikutusnäkymässä elementtien välillä on *tietovirta* (Flow), jossa liikkuu informaatiota. Informaatio on liiketoimintakerroksessa esimerkiksi *käsite* (Business Object), ja sovelluskerroksessa *tietoelementti* (Data Object). Tietovirrassa siirtyvä informaatio voidaan kirjoittaa tietovirtaa kuvaavan yhteystyyppinuolen nimeksi (label), tai mikäli väline mahdollistaa, informaatio-objekti voidaan valita välineen repositoriosta.

2.4.1 Toimijoiden vuorovaikutus



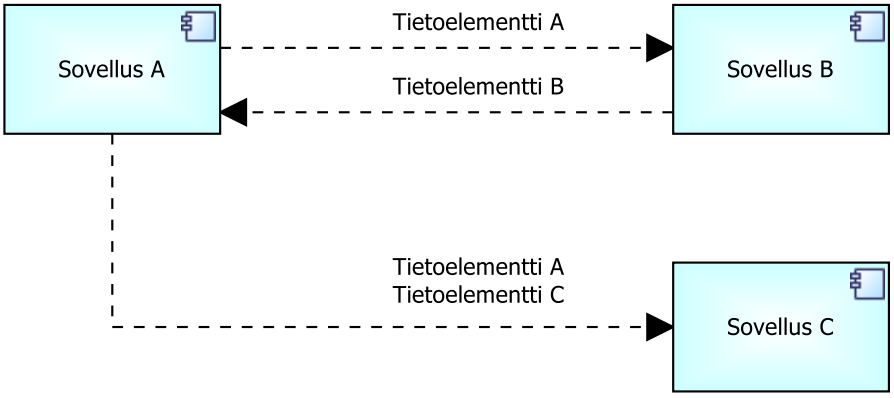
Kuva 16: Toimijoiden vuorovaikutus -näkymä.

2.4.2 Prosessien vuorovaikutus



Kuva 17: Prosessien vuorovaikutus -näkymä.

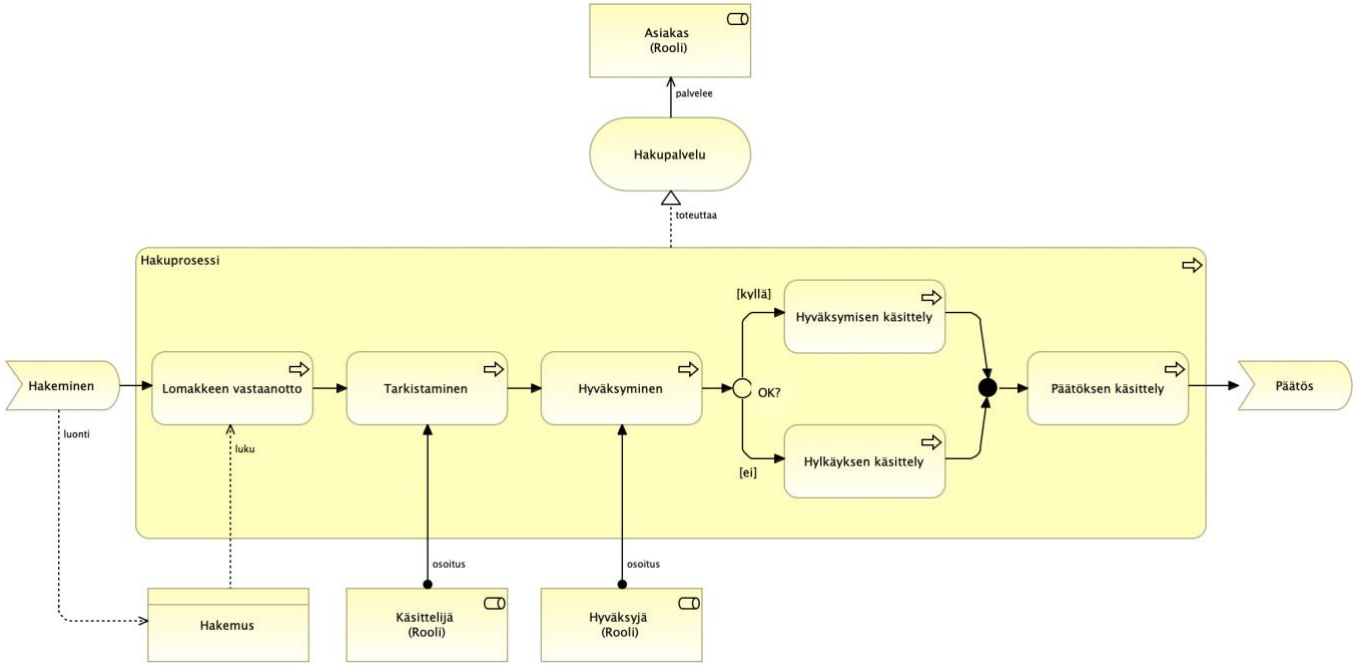
2.4.3 Sovellusten välinen vuorovaikutus



Kuva 18: Sovellusten vuorovaikutus -näkymä.

Tässä kuvataan sovellusten väliset tietovirrat: mitä tietoa siirtyy mistäkin sovelluksesta mihinkin. Vuorovaikutuskaavio ei kuvaa tiedonvaihdon dynamiikkaa: mikä sovellus aloittaa tiedonvaihdon, tai mitä rajapintoja käytetään. Näitä voidaan kuvata tarvittaessa erillisillä, vuorovaikutuksen dynamiikkaa kuvaavilla kaavioilla.

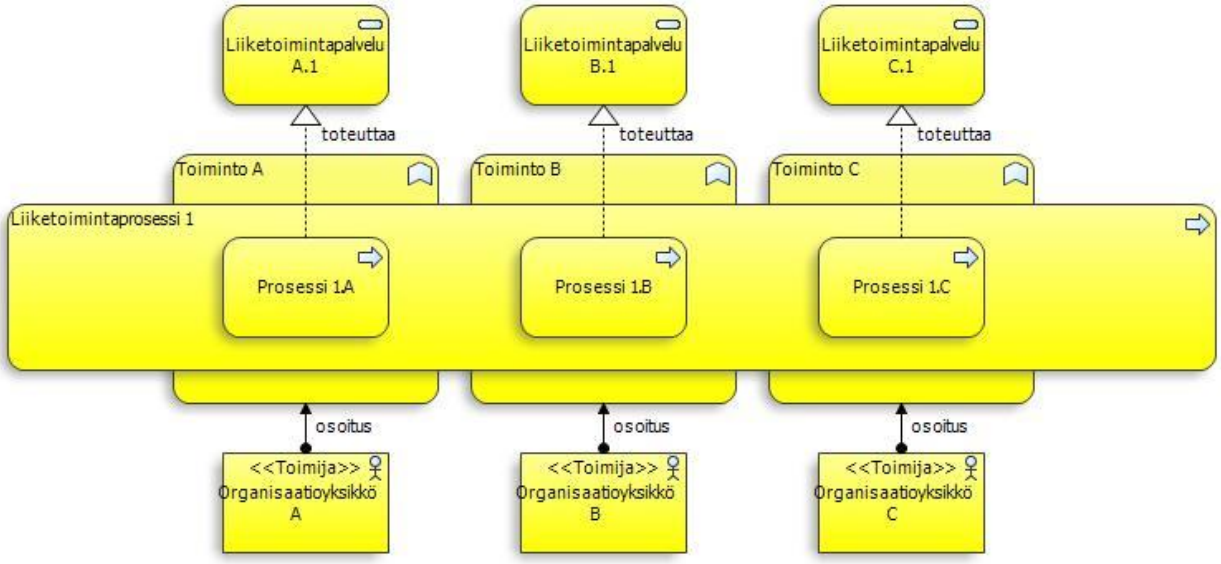
2.5 Prosessinäkymä



Kuva 19: Prosessinäkymä - esimerkki.

Prosessinäkymässä käytetään mm. ArchiMate liiketoimintakerroksen elementtejä *prosessi* (Business Process), *toimija* (Business Actor), *rooli* (Business Role), *käsite* (Business Object) ja *tapahtuma* (Business Event). Yhteystyyppeinä *laukaisu* (Trigger) ja *pääsy* (Access).

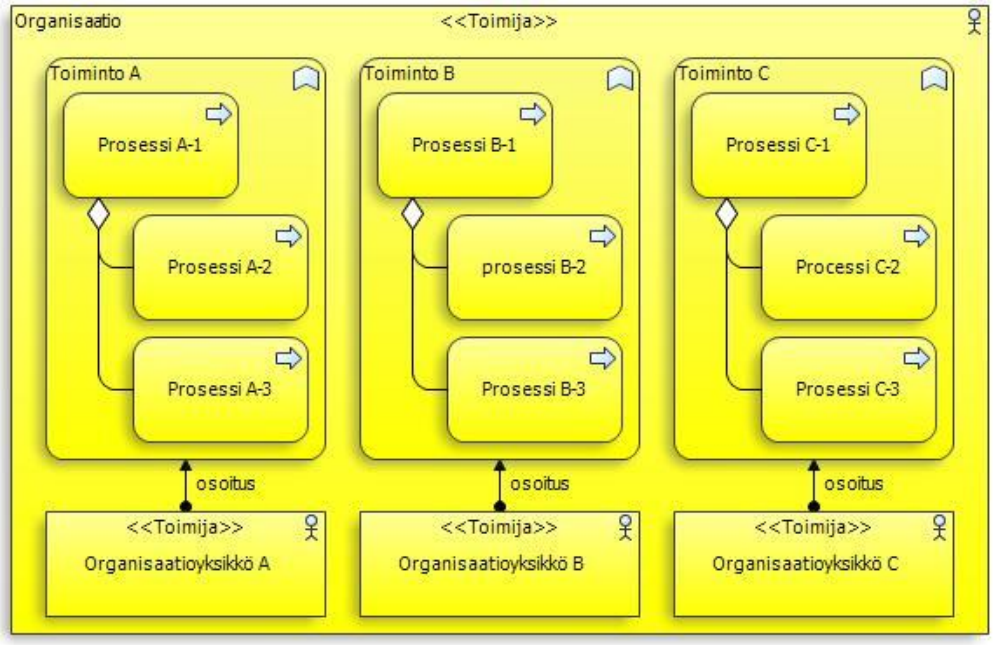
2.5.1 Prosessi toimintokohtaisesti



Kuva 20: Prosessin toimintokohtainen jaottelu.

Prosessi voi ylittää organisaation toimintoja (Business Function). Toiminto on kooste organisatorisesta toiminnasta / käyttäytymisestä (behavior) (kuten prosesseista), joita yhdistää jokin määrätty kriteeri.

2.5.2 Prosessikartta toiminnoittain

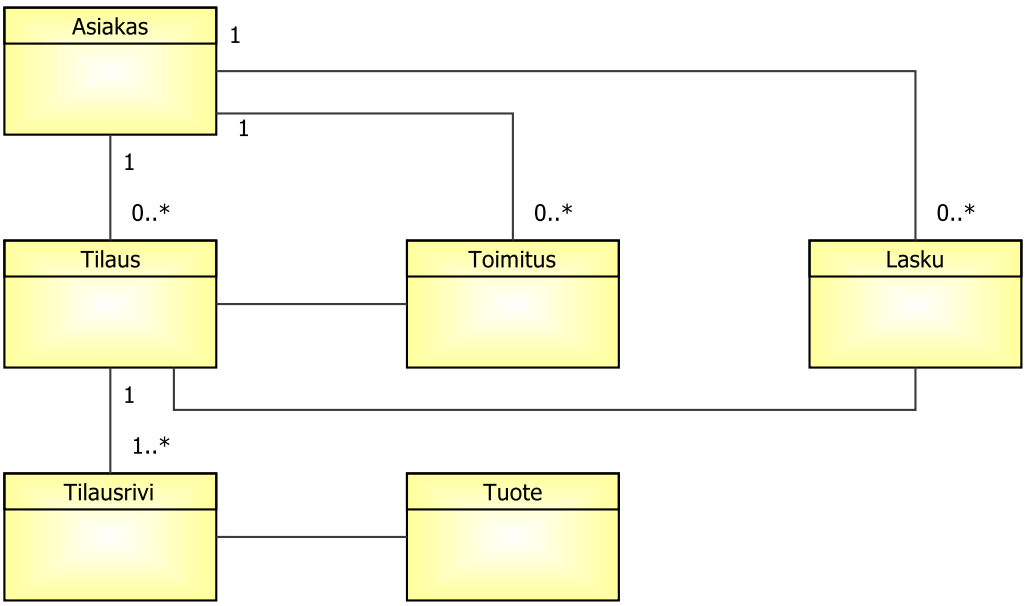


Kuva 21: Prosessikartta - toiminnoittain.

Prosessikartassa voidaan esittää organisaation pääprosessit, ja niihin liittyvät osaprosessit, toiminnoittain jäsenneltynä. Toimintoihin (Business Function) tyypillisesti liittyy jokin organisatorinen toimija (Business Actor), jotka on esitetty yllä olevassa prosessikartassa vain havainnollistamaan toimintojen ja toimijoiden yhteyttä. Sinällään prosessikarttaan eivät kuulu rakenteelliset toimijat.

2.6 Käsitemalli

Käsitemalli-näkymän avulla kuvataan kehitettävän kohteen keskeiset käsitteet ja niiden väliset suhteet.

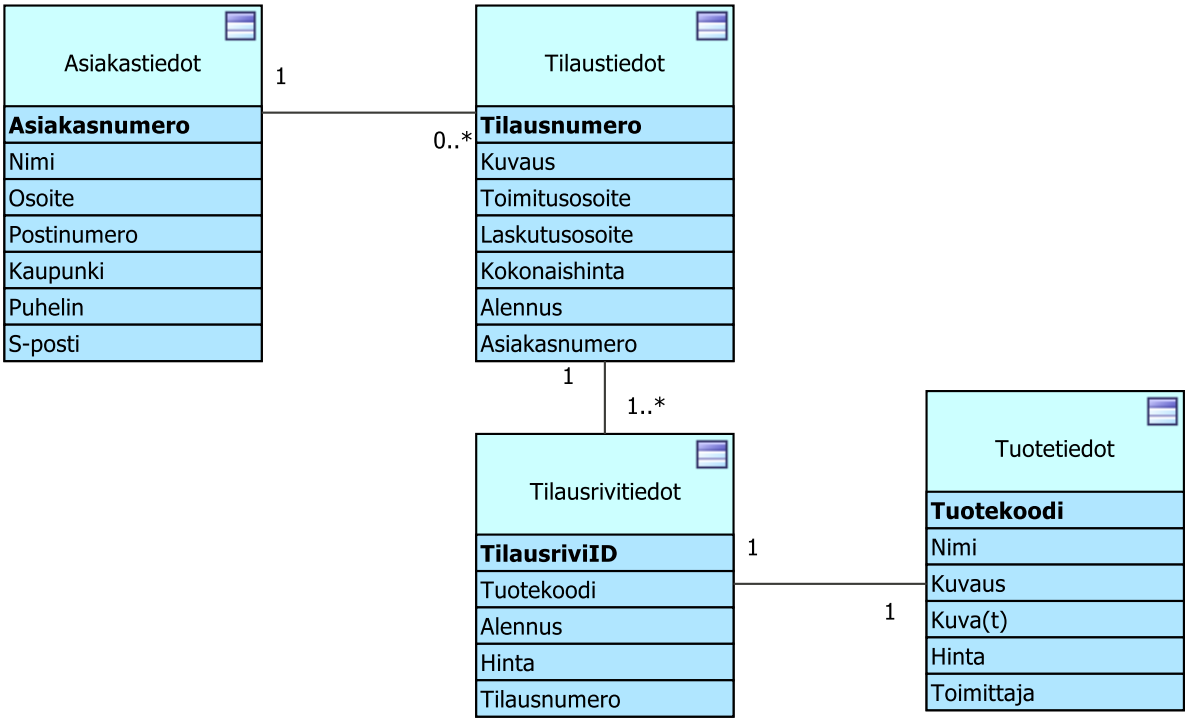


Kuva 22: Käsitemallinäkymä - esimerkki.

Tässä versiossa käsitteiden välisissä yhteyksissä on mukana kardinaliteetit, vaikka ne eivät kuulu ArchiMate-standardiin. Käsitemallissa käytetään ArchiMate *käsite* (Business Object) -elementtiä.

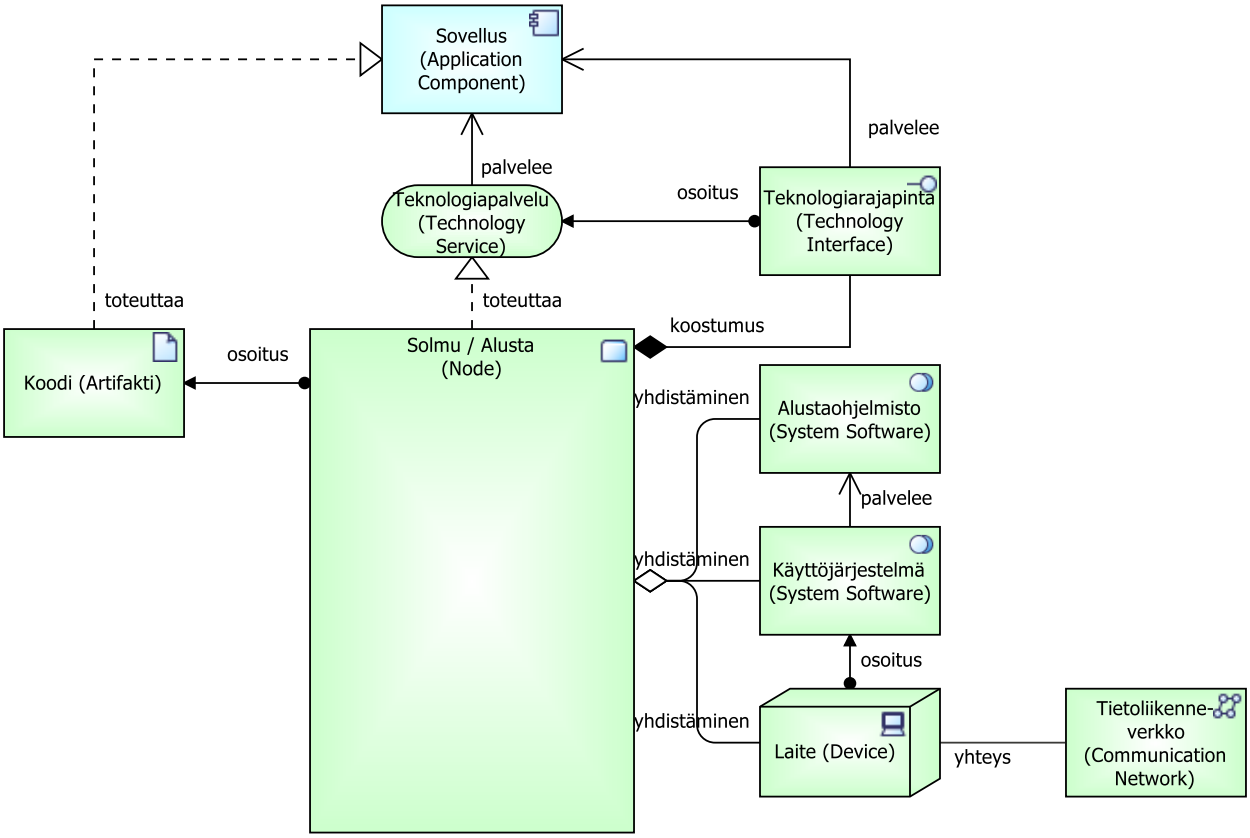
2.7 Tietomalli

Tietomallinäkymän avulla kuvataan kehitettävän kohteen tarkemman tason tietomalli: tietoelementit (Data Object) ja attribuutit, sekä tietoelementtien väliset suhteet (kardinaliteetteineen, kuten käsitemalli-näkymässä).



Kuva 23: Tietomallinäkymä - esimerkki.

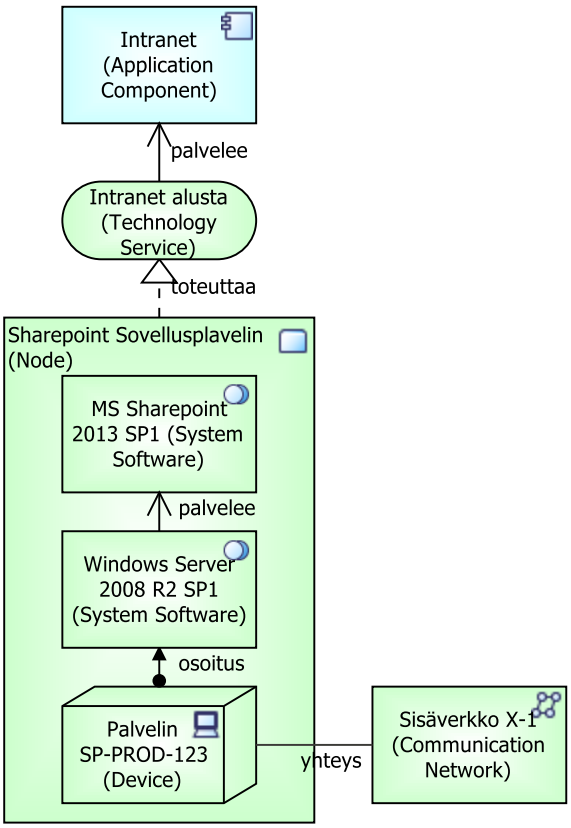
2.8 Teknologia-alusta -näkymä (Technology Platform View)



Kuva 24: Teknologia-alusta -näkymä - suunnittelumalli.

Teknologia-alustan eli infrastruktuurinäkymän avulla kuvataan kehittämiskohteen (kuten sovelluksen) alustaratkaisu: ohjelmistot, laitteistot jne. Tällä näkymällä voidaan kuvata myös sijoittelua (depolyment), sekä esim. kuormanjakoa, klusterointia, verkkotopolgiaa yms.

Teknologa-alusta -näkymä mallinnetaan ArchiMate Teknologiakerroksen elementeillä kuten: *solmu / alusta* (Node), *teknologiapalvelu* (Technology Service), *artifakti* (Artifact), *laite* (Device), *järjestelmä-/alustaohjelmisto* (System Software), *teknologiarajapinta* (Technology Interface) ja *tietoliikenneverkko* (Communication Network).



Kuva 25: Teknologia-alusta – esimerkki.

2.9 Sovellusarkkitehtuuri -näkymä

Sovellusarkkitehtuuri eli ratkaisuarkkitehtuuri (Solution Architecture) on kokonaisarkkitehtuuria (Enterprise Architecture) tarkempi taso. Kokonaisarkkitehtuurissa sovellus (Application Component) eli järjestelmä on pienin tarkasteltava yksikkö, ja tässä mielessä sovellusten sisäinen rakenne ei ole kokonaisuuden kannalta kiinnostava. Sovellusarkkitehtuurissa sen sijaan tarkastellaan sovelluksen sisäistä rakennetta, osakomponentteja eli moduulijakoa. Sovellusarkkitehtuurissa voidaan kuvata, mikä osakomponentti (Application Component) tarjoaa minkäkin *sovelluspalvelun* (Application Service) tai *sovellusrajapinnan* (Application Interface), ja mikä osakomponentti toteuttaa minkäkin *sovellusprosessin* (Application Process) tai *sovellustoiminnon* (Application Function).

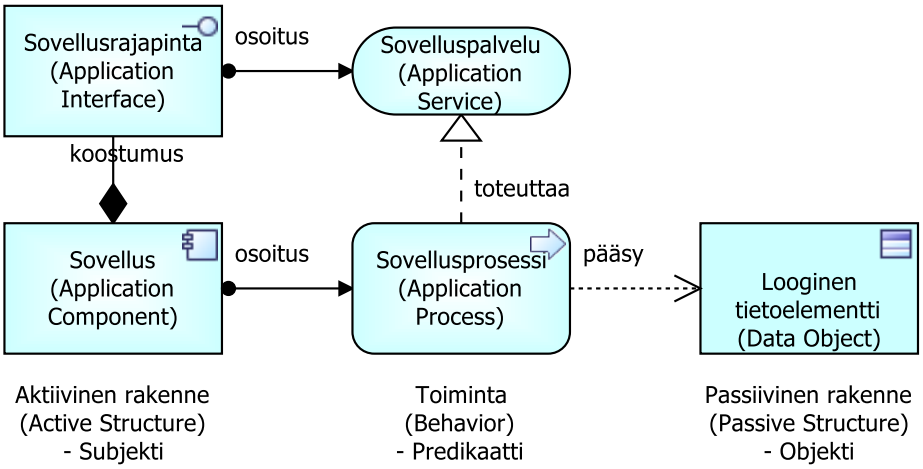
Sovellus. Järjestelmästä (myös tietojärjestelmä) käytetään kokonaisarkkitehtuurissa ja sen mallintamisessa nimitystä *sovellus* (Application). Määritys pohjautuu ArchiMate-notaatioon, jossa sovellusta mallinnetaan *sovelluskomponentti* (Application Component) -elementillä. Sovellus on rakenteellinen elementti (entiteetti), joka kapseloi sisäänsä määrätyn toiminnallisuuden. Sovellus on organisaation liiketoiminnan kannalta merkityksellinen yksikkö, jota voidaan itsenäisesti kehittää, asentaa tai se voidaan korvata toisella sovelluksella, joka tarjoaa vastaavan toiminnallisuuden.

Sovellus on aktiivinen rakenne-elementti, joka suorittaa määrättyä toiminnallisuutta (*behavior*) ja hallinnoi määrättyä tietoa (data). Sovelluksen sisältämää toiminnallisuutta mallinnetaan elementeillä *sovellustoiminto* (Application Function), ja sovellusprosessi (*Application Process*). Sovelluksen käsittelemää tietoa mallinnetaan *tietoelementillä* (Data Object). Sovelluksen ulospäin tarjoamaa toiminnallisuutta, *sovelluspalveluita* (Application Services), muut sovellukset käyttävät *sovellusrajapintojen* (Application Interfaces) kautta. Vastaavasti sovellus voi käyttää muiden sovellusten tarjoamia sovelluspalveluita sovellusrajapintojen kautta. Liiketoiminnan kannalta merkityksellistä on tuoda esiin, mitä sovelluspalveluita sovellus tarjoaa: sovelluspalvelut ovat syy sovelluksen olemassaololle.

Sovelluskomponentti (Application Component) -elementillä voidaan mallintaa mitä tahansa sovelluskerroksen (Application Layer) rakenteellista elementtiä. Esimerkiksi koko sovellusta (järjestelmää), tai sen yksittäistä osaa eli osajärjestelmää (moduulia). Sovelluskomponentti-elementillä voidaan mallintaa myös kokonaista ryhmää sovelluksia, jos ollaan kiinnostuneita vain määrätyn tyyppisistä sovelluksista, jotka käyttäytyvät samalla tavalla kehittämiskohteeseen nähden (kuten ”Asiakasjärjestelmät”, ”Lähdejärjestelmät”, ”Taloushallinnon järjestelmät” jne.). ArchiMate-notaatiossa kaikkia mallinnuselementtejä voidaan käyttää eri abstraktiotason asioiden mallintamiseen.

LeanEA-viitekehyksen tasolla kaksi (2) on Sovellusnäkymät –kerros sovelluksille. Siellä on sisältöalueet mm. sovelluspalveluille ja sovelluksille sekä sovellusintegraatioille. Sovelluksiin liittyy määrätyt kaaviotyypit (sovellus- eli järjestelmäkartta, sovellusintegraatiot, sovellusrakenne), jotka on kuvattu tarkemmin erikseen.

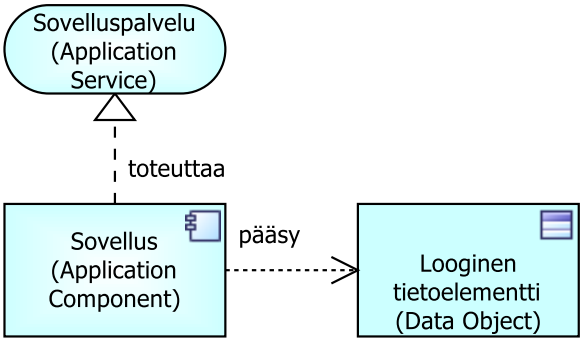
2.9.1 Sovelluksen suunnittelumalli (perusmalli)



Kuva 26: Sovelluksen suunnittelumalli (”perusmalli”).

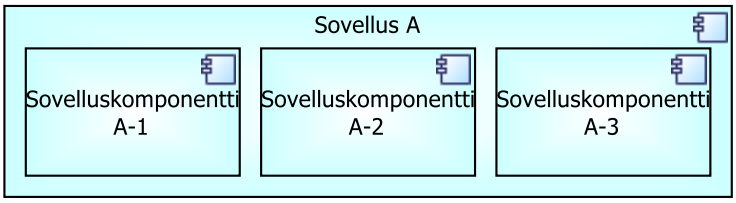
Sovellusta (tai järjestelmää) mallinnetaan *sovelluskomponentilla* (Application Component). Sovelluksen sisäisiä toiminnallisuuksia mallinnetaan *sovellusprosessi* (Application Process) tai *sovellustoiminto* (Application Function) -elementeillä. Sovelluksen ulospäin tarjoamaa toiminnallisuutta mallinnetaan *sovelluspalvelu* (Application Service) -elementillä (esimerkiksi käyttötapauksia). Sovelluksen tarjoamia käyttöliittymiä (GUI) tai sovellusrajapintoja (API) mallinnetaan *sovellusrajapinta* (Application Interface) -elementillä.

Derivointisääntöjen mukaan yllä oleva voidaan esittää yksinkertaistettuna ilman sovellusprosessia (kuva alla).



Kuva 27: Sovelluksen suunnittelumalli (yksinkertaistus perusmallista).

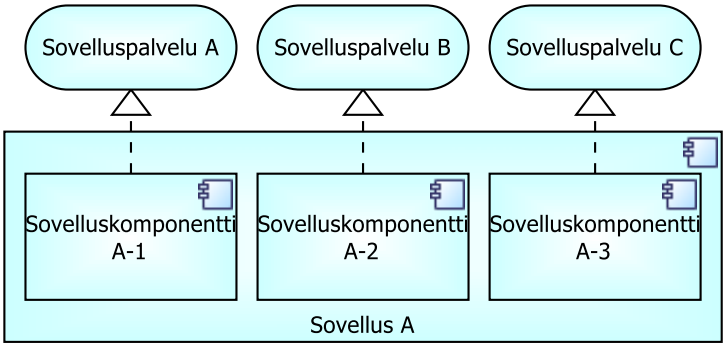
2.9.1.1 Sovelluksen looginen rakenne (sovellusrakenne) -näkymä



Kuva 28: Sovelluksen looginen rakenne (osakomponentteihin jakaminen /moduulijako).

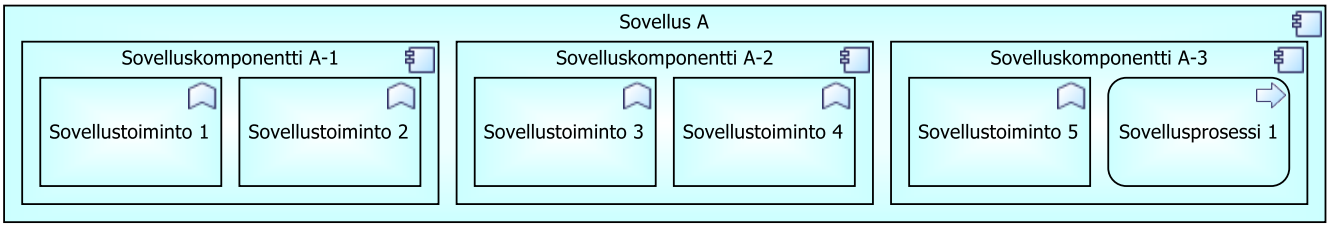
Sovellus (eli järjestelmä) voi koostua useista alemman tason sovelluskomponenteista (eli osajärjestelmistä, moduuleista). *Sovelluskomponentti* (Application Component) -elementtiä käytetään mallintamaan kaikkia eritasoisia sovelluksen osia. Sovelluksen sisäisestä rakenteesta voidaan laatia ”sovellusrakenne” -näkymä (kaavio).

Alla oleva kaavio havainnollistaa mikä osajärjestelmä/moduuli tarjoaa minkäkin sovelluspalvelun. Osakomponentit A-1, A-2 ja A-3 on kytketty Sovellukseen A *koostumus* (Composition) tai *yhdistäminen* (Aggregation) -yhteystyypeillä, jotka eivät ole näkyvissä, koska osakomponentit on sisällytetty (nesting) pääkomponenttiin.



Kuva 29: Sovelluksen looginen rakenne: osakomponentit/moduulit ja sovelluspalvelut.

Alla oleva kaavio havainnollistaa mitä toiminnallisuuksia kukin osajärjestelmä/moduuli sisältää. Tällä kaaviolla voidaan suunnitella mm. modularisointia, loogisesti yhteen kuuluvien toiminnallisuuksien liittämistä yhteen. *Sovellustoiminnot* (Application Function) on kytketty sovelluskomponentteihin *osoitus* (Assignment) -yhteystyypeillä, jotka eivät näy, koska sovellustoiminnot on sisällytetty (nesting) sovelluskomponentteihin.



Kuva 30: Sovelluksen looginen rakenne: toiminnallisuuksien kytkeminen moduuleihin.

Huom! Sovelluksen toiminnallisuuksia voidaan mallintaa sekä sovellustoiminnoilla (Application Function) että sovellusprosesseilla (Application Process). Jälkimmäistä voidaan soveltaa paremmin esimerkiksi ohjelmistorobotiikan prosessien kuvaamiseen (Robotic Process Automation, RPA).

2.9.2 Komponenttimalli

Komponenttimalli:

- Kuvaa kohdesovelluksen loogisen rakenteen eri abstraktiotasoilla (0 - n) tarpeen mukaan.
- Eri tasoilla esitetyt mallit kuvaavat miten kohdesovellus liittyy ulkoiseen ympäristöönsä ja mistä rakenneosista kohdesovellus sisäisesti koostuu.

Komponenttimalli-0 (KM-0) kuvaa:

- Sovelluksen mustana laatikkona (black-box) suhteessa ympäristöönsä, ns. kokonaisarkkitehtuuritaso
- Miten sovellus kommunikoi muiden järjestelmien kanssa, eli riippuvuussuhteet: mikä palvelee mitä Palvelurajapinnat eri sovellusten välillä.

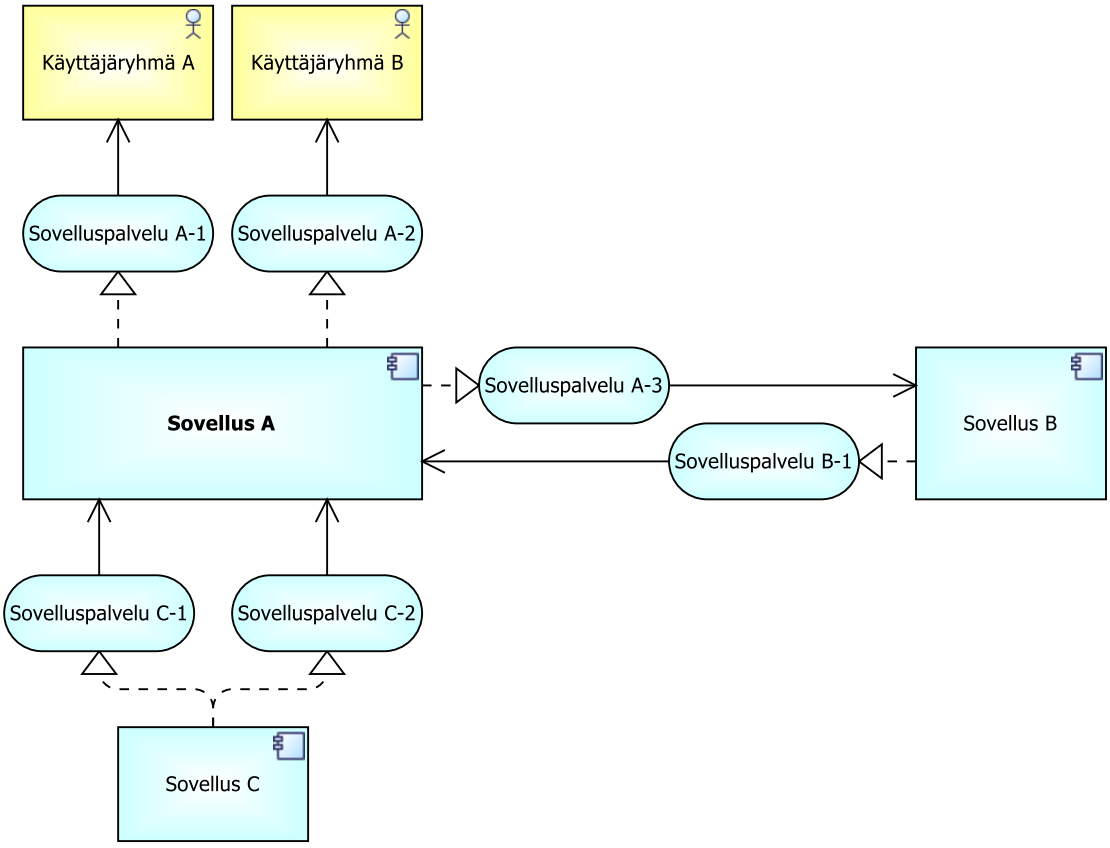
Komponenttimalli-1 (KM-1) kuvaa:

- Sovelluksen sisäisen rakenteen (white-box).
- Sovelluksen pääkomponentit eli moduulit, sekä niiden väliset suhteet ja vastuunjako.

Komponenttimalli-2 (KM-2) kuvaa:

- Sovelluksen pääkomponenttien sisäisen rakenteen (alikomponentit) ja vuorovaikutukset.

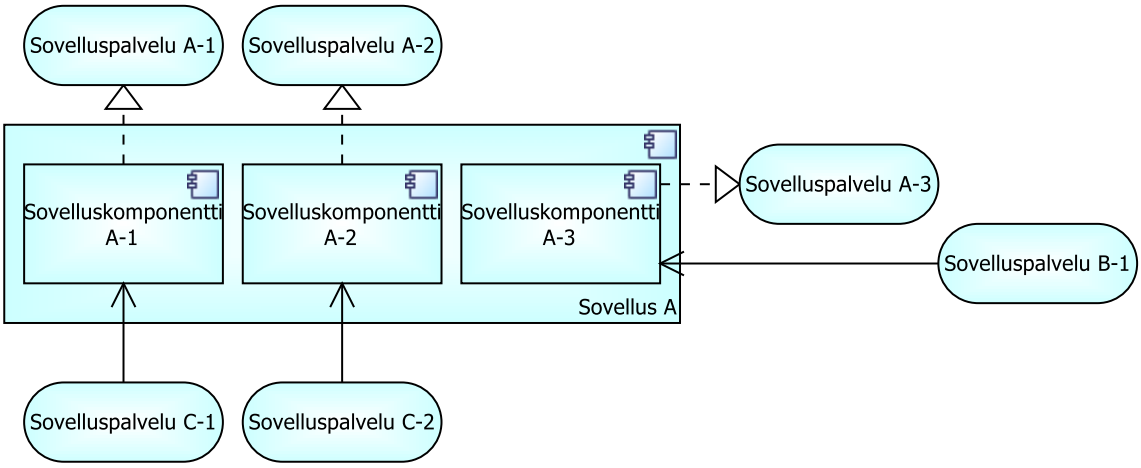
2.9.2.1 Komponenttimalli-0 (KM-0)



Kuva 31: Komponenttimalli-0 (KM-0). Sovelluksen suhde toimintaympäristöön (konteksti).

Kohdesovellus ”A” esiintyy kaavion keskellä, ”mustana laatikkona”: sen sisäinen rakenne ei ole merkityksellinen, vaan sen tarjoamat ja käyttämät sovelluspalvelut sen sijaan ovat. Kokonaisarkkitehtuurin kannalta on kiinnostavaa se, mitä sovelluksia organisaatiolla on käytössään, ja miten ne käyttävät toistensa palveluita, eli riippuvuussuhteet.

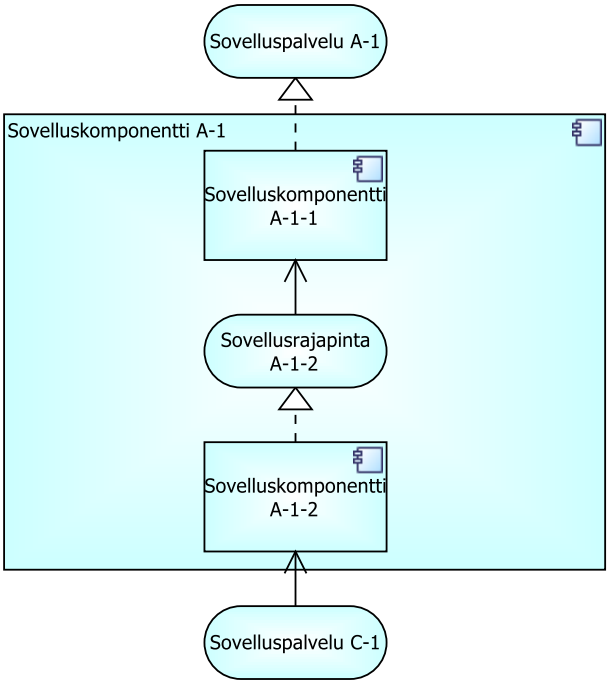
2.9.2.2 Komponenttimalli-1 (KM-1)



Kuva 32: Komponenttimalli-1 (KM-1). Sovelluksen sisäinen rakenne.

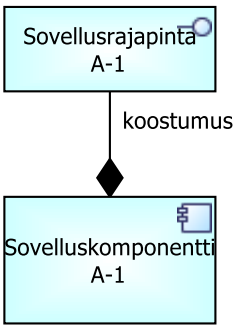
Kohdesovellus ”A:n” sisäinen rakenne on avattu, ja ulkoiset sovellukset on jätetty kaaviosta pois. Kaaviolla voidaan osoittaa, miten sovelluksen osakomponentit tarjoavat ja käyttävät sovelluspalveluita.

2.9.2.3 Komponenttimalli-2 (KM-2)



Kuva 33: Komponettimalli-2 (KM-2). Eräs sovelluksen ”A” pääkomponenteista avattuna tarkemmin.

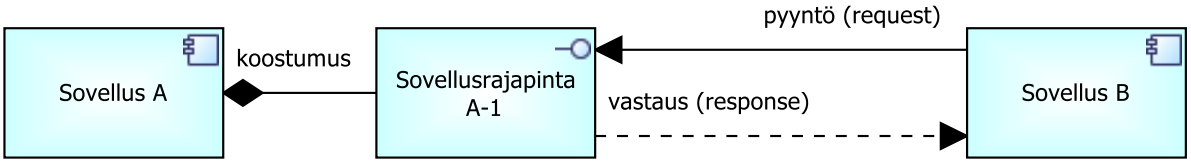
Kun sovelluspalvelun sijaan halutaan kuvata mitä sovellusrajapintaa käytetään, silloin sovellusrajapinta kytketään sovelluskomponenttiin *koostumus* (Composition) -yhteystyypillä (kuva alla).



Kuva 34: Sovellus ja sovellusrajapinta.

2.9.3 Sovellusintegraatiot

2.9.3.1 Sovellusrajapinta ja synkroninen pyyntö-vastaus (Request-Reply) suunnittelumalli

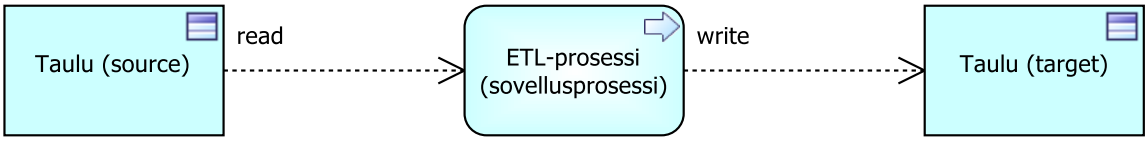


Kuva 35: Sovellusrajapinta ja synkroninen pyyntö-vastaus suunnittelumalli.

Sovellusten välinen app2app integraatioesimerkki: ”Sovellus A” tarjoaa sovellusrajapinnan ”A-1”, jota käyttää ”Sovellus B”. Aktiivinen osapuoli on ”Sovellus B”, joka aloittaa tiedonvaihdon. ”Sovellus B” kutsuu ”Sovellus A:n” toteuttamaa sovellusrajapintaa ”A-1”. ”Sovellus B” välittää kutsussa tietorakenteen/-sanoman ”request”, ja saa vastauksena tietorakenteen/sanoman ”response” ”Sovellus A:lta”.

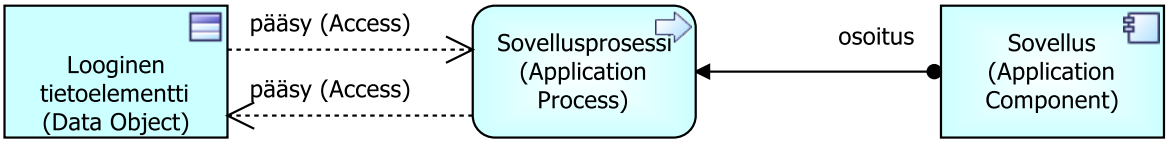
Tässä kaaviossa käytetään ArchiMate:n dynaamisia yhteystyyppejä *laukaisu* (Trigger) ja *tietovirta* (Flow).

2.9.3.2 ETL-prosessi



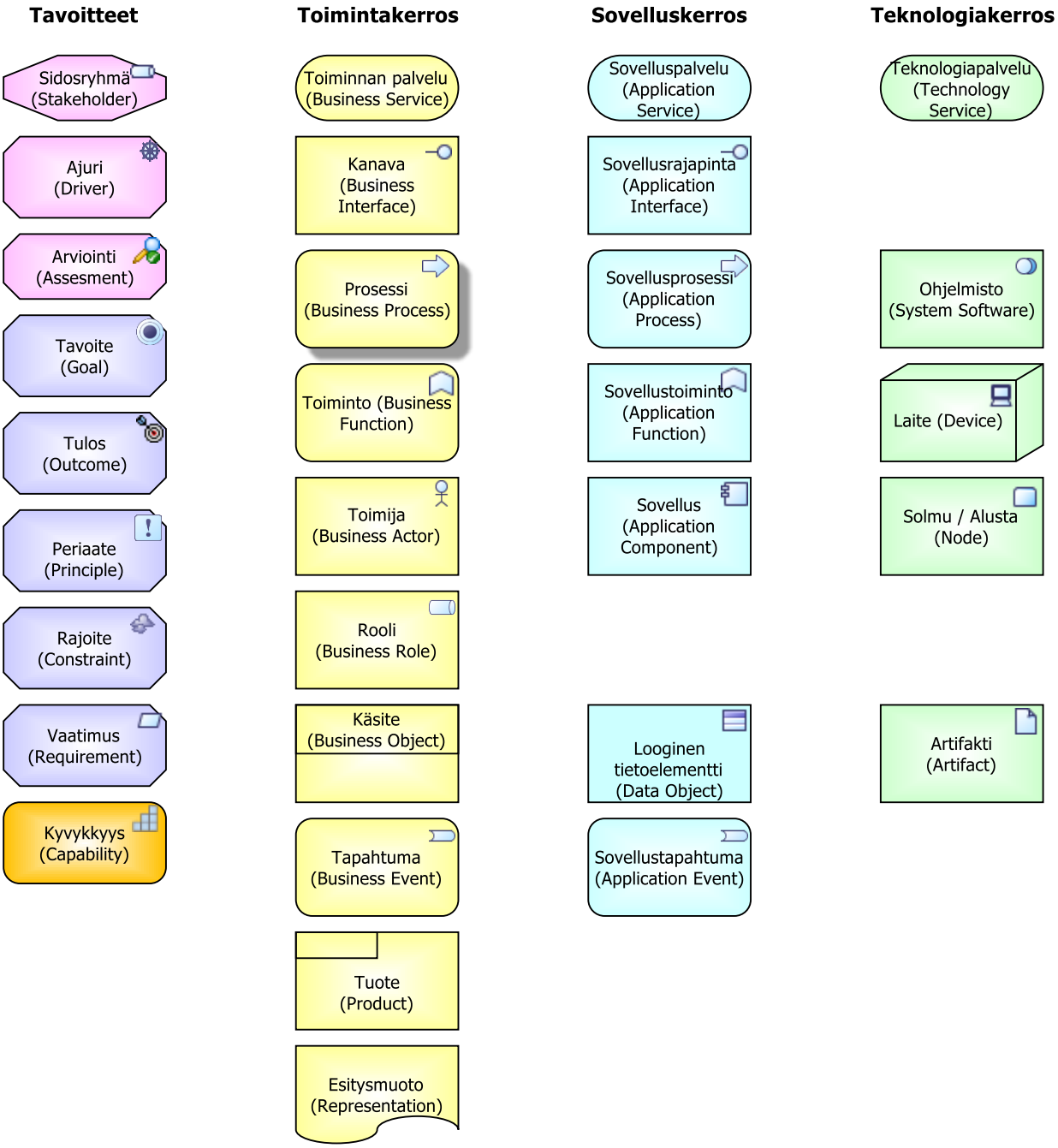
Kuva 36: ETL-prosessi.

ETL-prosessi lukee lähdetaulua ja kirjoittaa kohdetauluun (kuva yllä). ETL-prosessia mallinnetaan ArchiMate:n *Sovellusprosessi* (Application Process) -elementillä. ETL-prosessi ja suorittava sovellus (ETL-ratkaisu) kytketään toisiinsa ArchiMate:n rakenteellisella yhteystyypillä *osoitus* (Assignment) (kuva alla).



Kuva 37: ETL-prosessi ja suorittava sovellus.

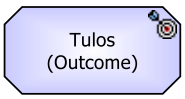
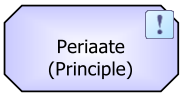
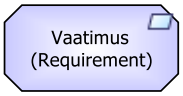
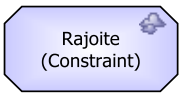
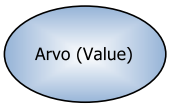
3. ArchiMate-elementtien osajoukko



Kuva 38: ArchiMate-elementtien osajoukko.

Tässä esimerkkinä ArchiMate-elementtien osajoukko, joka riittää useimpiin mallinnustarpeisiin.

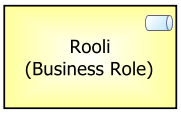
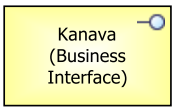
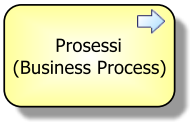
3.1 ArchiMate Motivation -elementit

ArchiMate Motivation- elementit (Tavoitteet -näkymä)		
Nimi	**Kuvaus**	**Symboli**
Sidosryhmä (Stakeholder)	Yksilö, ryhmä tai organisaatio, jolla on jokin intressi tai vaatimuksia kehittämiskohteen suhteen, tai joka on esittänyt muutostarpeen tai kiinnostunut muutoksesta ja sen vaikutuksista.	
Ajuri (Driver)	Ulkoinen tai sisäinen (organisaation suhteen) vaikutin (olosuhde, asiantila tms.), joka motivoi organisaatiota toteuttamaan muutoksia määriteltyjen tavoitteiden saavuttamiseksi. Kuvaa MIKSI jokin kehittämiskohde on tärkeä, josta syystä muutos on tarpeellinen.	
Arviointi (Assesment)	Analyysin tulos, arvio, oletus tai odotus, joka voidaan kytkeä ajuriin. Arvioinnin avulla ajuri voidaan liittää tavoitteiseen. Arviointeja voidaan käyttää esim. SWOT-analyyseissä.	
Tavoite (Goal)	Konkreettinen tahtotila, tarkoitus, haluttu asiantila, jonka organisaatio tai joku sen sidosryhmä haluaa saavuttaa. Tulisi sisältää laadullisen määreen kuten "parantaminen", "helpottaminen". Esim. asiakaspalvelun parantaminen, prosessin tehostaminen, tuottavuuden kasvattaminen, reklamaatioiden vähentäminen.	
Tulos (Outcome)	Tavoitteeseen liittyvä, mitattavissa oleva lopputulema, joka saavutetaan organisaation kyvykkyyksillä. Esim. mittari, tavoitetaso.	
Periaate (Principle)	Kehittämiskohteiden toteuttamisessa huomioitava laadullinen asia. Esim. tarkka ja ohjaava määritys tai ylätason linjaus.	
Vaatimus (Requirement)	Konkreettinen, tavoitteista johdettu, kehittämiseen kohdistuva, kehittämiskohteen (tai sen osan) rajattu tarvemääritys. Keino tavoitteen toimeenpanoon. Toiminnallinen tai ei-toiminnallinen vaatimus. Esim. sovelluksen haluttu toiminnallisuus / ominaisuus.	
Rajoite (Constraint)	Tekijä, joka vaikuttaa tai tulee huomioida kehittämiskohteen toteuttamisessa, tai joka rajoittaa sen toteutustapaa. Esim. toteutusteknologia, aika- tai budjettiraami.	
Arvo (Value)	Johonkin elementtiin liitettävissä oleva arvo. Voi ilmaista rahallista arvoa, mutta ilmaista myös jonkin asian tai tiedon käyttöarvoa, tärkeyttä. Esim. palvelun käyttöoikeutta, palvelun tuottamaa hyötyä tai lisäarvoa. Arvo voidaan kytkeä arvovirran (Value Stream) vaiheeseen (Value Stage).	

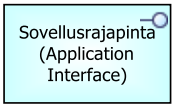
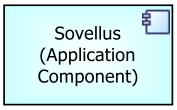
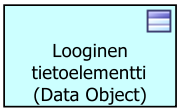
3.2 ArchiMate Strategy-elementit

ArchiMate Strategy Layer -elementit		
Nimi	**Kuvaus**	**Symboli**
Toimenpide (Course of Action)	Lähestymistapa tai suunnitelma tavoitteiden mukaisten kyvykkyyksien aikaansaamiseksi. Toimenpiteet liittyvät tavoitteisiin. Esim. strateginen tai taktinen.	
Kyvykkyys (Capability)	Kyky tai taito, jonka organisaatio, toimija tai järjestelmä omaa.	
Resurssi (Resource)	Resurssit voivat olla aineellisia (esim. taloudelliset, fyysiset) tai aineettomia (esim. teknologia, patentit, maine/brandi, kulttuuri) tai henkilöstöön liittyviä (osaaminen/tieto-taito, menetelmäosaaminen, yhteistyö, motivaatiotekijät jne.). Resurssit liittyvät kyvykkyyksiin.	
Arvovirta (Value Stream)	Tulossa ArchiMate 3.1 -versiossa.	

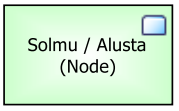
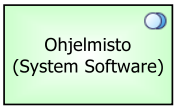
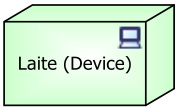
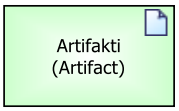
3.3 ArchiMate Business Layer -elementit

ArchiMate Business Layer -elementit - Toimintakerros		
Nimi	**Kuvaus**	**Symboli**
Toimija (Business Actor)	Kohteeseen liittyvät ulkoiset ja sisäiset toimijat. Toimija voi olla organisaatio, organisaatioyksikkö (Organization Unit) tai henkilö (Person). Esim. asiakkaat ja kumppanit.	
Rooli (Business Role)	Rooli, jossa jokin toimija voi toimia. Rooliin liittyy organisatorisia vastuita ja osaamisvaatimuksia.	
Toiminnan palvelu (Business Service)	Liiketoiminnallinen palvelu, jonka organisaatio tarjoaa. Nämä ovat organisaation substanssitoiminnan keskeisimpiä ylätason palveluita. Palvelut voidaan kytkeä asiakkaisiin tai prosesseihin. Palvelut voidaan tarjota eri kanavien kautta.	
Toiminnan rajapinta / kanava (Business Interface)	Kanava, rajapinta jonka kautta liiketoiminnallista palvelua tarjotaan ja käytetään.	
Toimintaprosessi (Business Process)	Liiketoimintaprosessi, ketju yhteenkuuluvia aktiviteetteja, jotka yhdessä suorittavat jonkin toiminnallisen kokonaisuuden. Toimintaprosessit osallistuvat ylätason toiminnallisten palveluiden tuottamiseen. Prosesseihin voidaan kytkeä toimijoita. Prosesseissa voidaan hyödyntää tietojärjestelmien tarjoamia sovelluspalveluita.	
Käsite (Business Object)	Liiketoiminnallinen käsite, joka kuvaa informaatiota jolla on merkitystä liiketoiminnan kannalta. Informaatiota siirtyy mm. toimijoiden ja prosessien välisessä vuorovaikutuksessa.	
Tapahtuma (Business Event)	Organisaatiossa tai sen ulkopuolella tapahtuva liiketoiminnallinen tilamuutos, joka laukaisee toimintaa, tai on seuraus toiminnasta.	

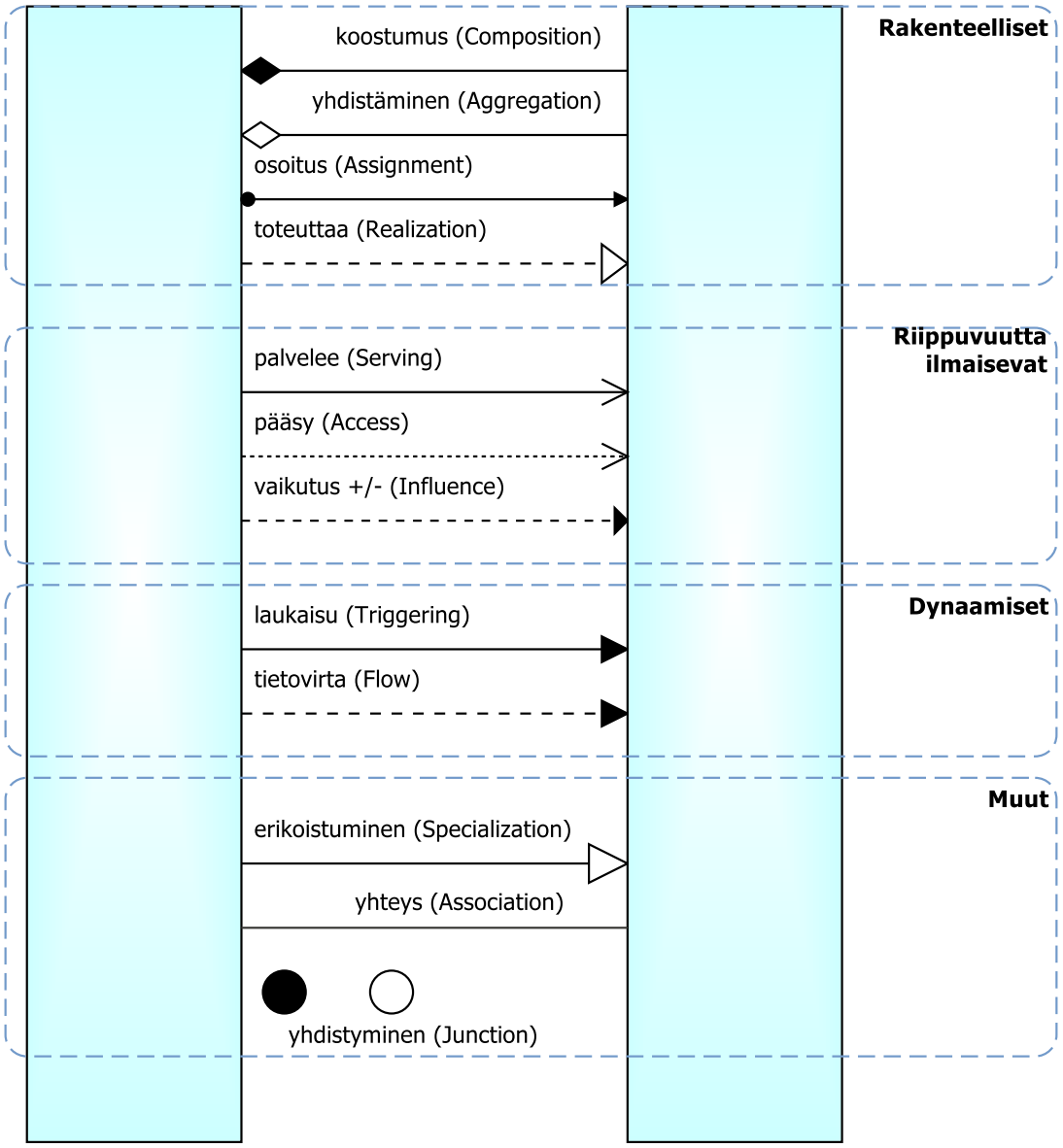
3.4 ArchiMate Application Layer -elementit

ArchiMate Application Layer -elementit - Sovelluskerros		
Nimi	**Kuvaus**	**Symboli**
Sovelluspalvelu / tietojärjestelmäpalvelu (Application Service)	Sovelluspalvelu jonka kautta sovellus tarjoaa toiminnallisuuksiaan ulospäin. Sovelluspalvelua käytetään sovellusrajapinnan kautta.	
Rajapinta (Application Interface)	Sovelluksen tarjoama rajapinta, jonka kautta sovelluksen tarjoamia sovelluspalveluita käytetään. Rajapinta voi olla a) käyttöliittymä (GUI) tai b) app2app sovellusrajapinta (API).	
Sovellus / järjestelmä (Application Component)	Sovellus / järjestelmä, joka on merkityksellinen yksikkö kokonaisarkkitehtuurin kannalta. Sovellus voi koostua osasovelluksista / osajärjestelmistä (moduuleista).	
Tietoelementti (Data Object)	Tieto, jota sovellus käsittelee ja hallitsee, ja joka siirtyy sovellusten välisessä vuorovaikutuksessa.	
Sovellusprosessi (Application Process)	Sovelluksen suorittama automaattinen toimintaketju.	
Sovellustoiminto (Application Function)	Sovelluksen suorittama automaattinen toiminto tai toimintojen muodostama kokonaisuus.	

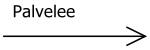
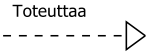
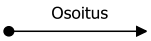
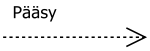
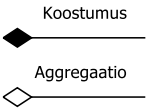
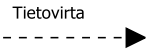
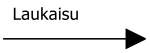
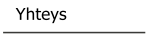
3.5 ArchiMate Technology Layer -elementit

ArchiMate Technology Layer -elementit - Teknologiakerros		
Nimi	**Kuvaus**	**Symboli**
Teknologiapalvelu (Technology Service)	Infrastruktuurin teknologia- tai alustapalvelu.	
Solmu (Node)	Looginen alustaelementti, joka kapseloi infrastruktuurin komponentteja. Esim. klusteri, alusta yms. johon voidaan liittää ohjelmistoja, laitteita yms.	
Järjestelmäohjelmisto (System Software)	Ohjelmisto joka on asennettu suoritusympäristöön (Solmuun). Esim. valmisohjelmisto, alustaohjelmisto, käyttöjärjestelmä.	
Laite (Device)	Infrastruktuurin fyysinen tai virtuaalinen teknologiakomponentti kuten palvelin tai verkkolaite (kytkin, palomuuri jne.).	
Verkko (Communication Network)	Tietoliikenneverkko	
Artefakti (Artifact)	Ohjelmistokehitysprosessin tuote. Esimerkiksi tiedosto (kuten lähdekooditiedosto), suoritettava koodi, skripti, tietokantataulu, sanoma, dokumentti.	

4. ArchiMate yhteystyypit



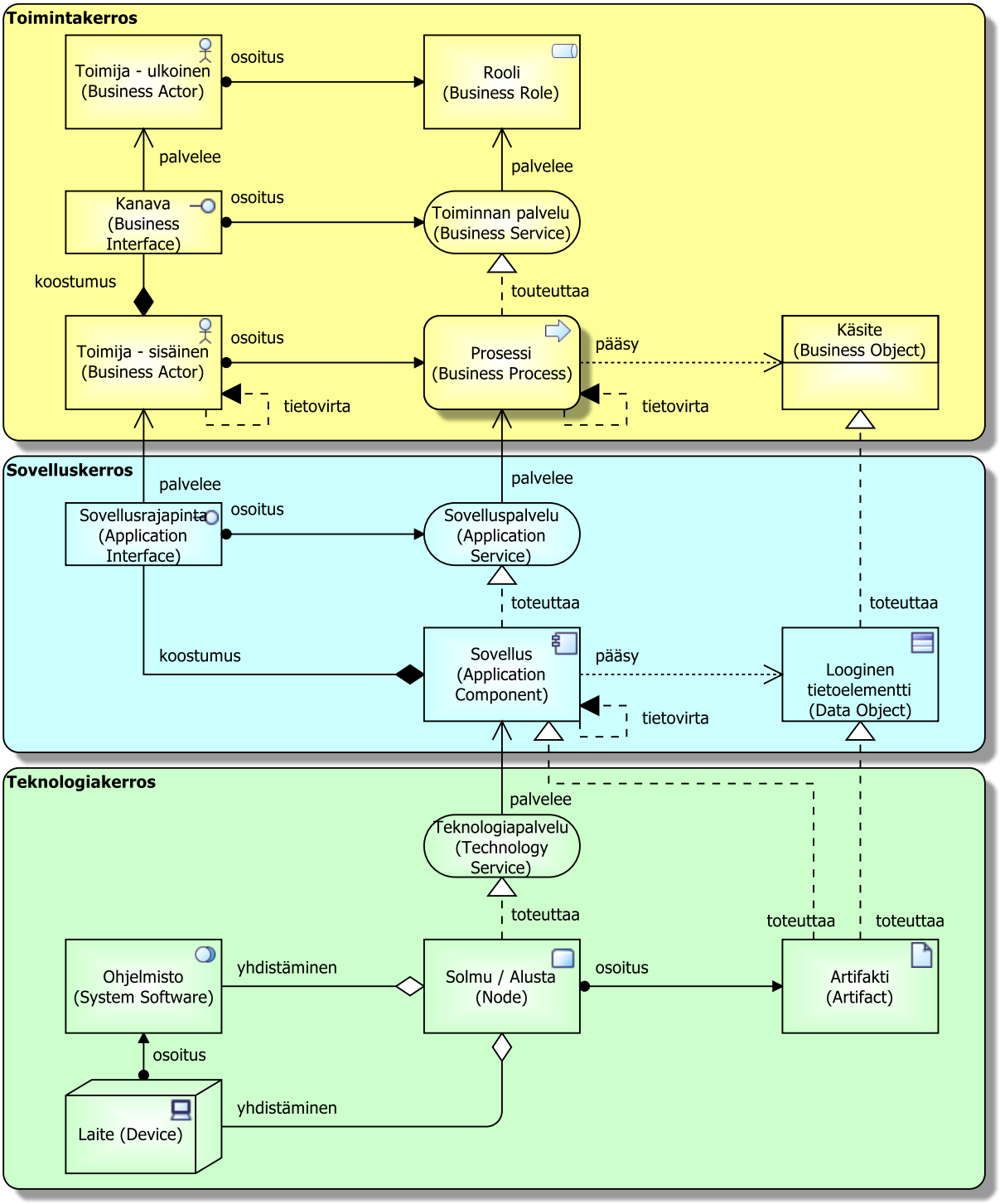
Kuva 39: ArchiMate -yhteystyypit.

ArchiMate yhteystyypit		
Nimi	**Kuvaus**	**Symboli**
Palvelee (Serving)	Riippuvuus elementtien välillä: lähde-elementti palvelee kohde-elementtiä, ts. lähde-elementtiä käyttää kohde-elementti. *Huom! Lukusuunta*.	
Realisoi / toteuttaa (Realization)	Rakenteellinen yhteys elementtien välillä: lähde-elementti realisoi/toteuttaa kohde-elementin	
Osoitus (Assignment)	Yhteys rakenteellisen elementin ja toiminnallisen elementin välillä. Esim. toimija voidaan kohdistaa prosessiin.	
Pääsy (Access)	Riippuvuus elementtien välillä: lähde-elementillä on pääsy kohde-elementtiin. Esim. sovelluksella on pääsy tieto-objektiin, sovellus käsittelee tietoa.	
Koostumus (Composition) ja (Aggregation)	Rakenteellinen yhteys elementtien välillä: juurielementti koostuu muista elementeistä. Elementtien elinkaaret on sidottu toisiinsa. Löyhemmässä koosteessa jossa symboli on valkoinen (Aggregation), koosteeseen kuuluvat elementit voivat kuulua myös muihin koosteisiin: elinkaaret eivät ole sidotut.	
Tietovirta (Flow)	Dynaaminen tietovirtayhteys lähde-elementistä kohde-elementtiin.	
Laukaisu (Trigger)	Dynaaminen yhteys joka laukaisee / käynnistää toiminnan kohde-elementissä	
Yhteys (Association)	Geneerinen yhteys kahden elementin välillä. Yleinen yhteystyyppi, joka ilmaisee yhteyden, mutta ei tarkasti sen luonnetta.	

Yhteystyypit ilmaisevat elementtien välistä a) rakenteellisuutta, b) riippuvuutta tai c) dynamiikkaa.

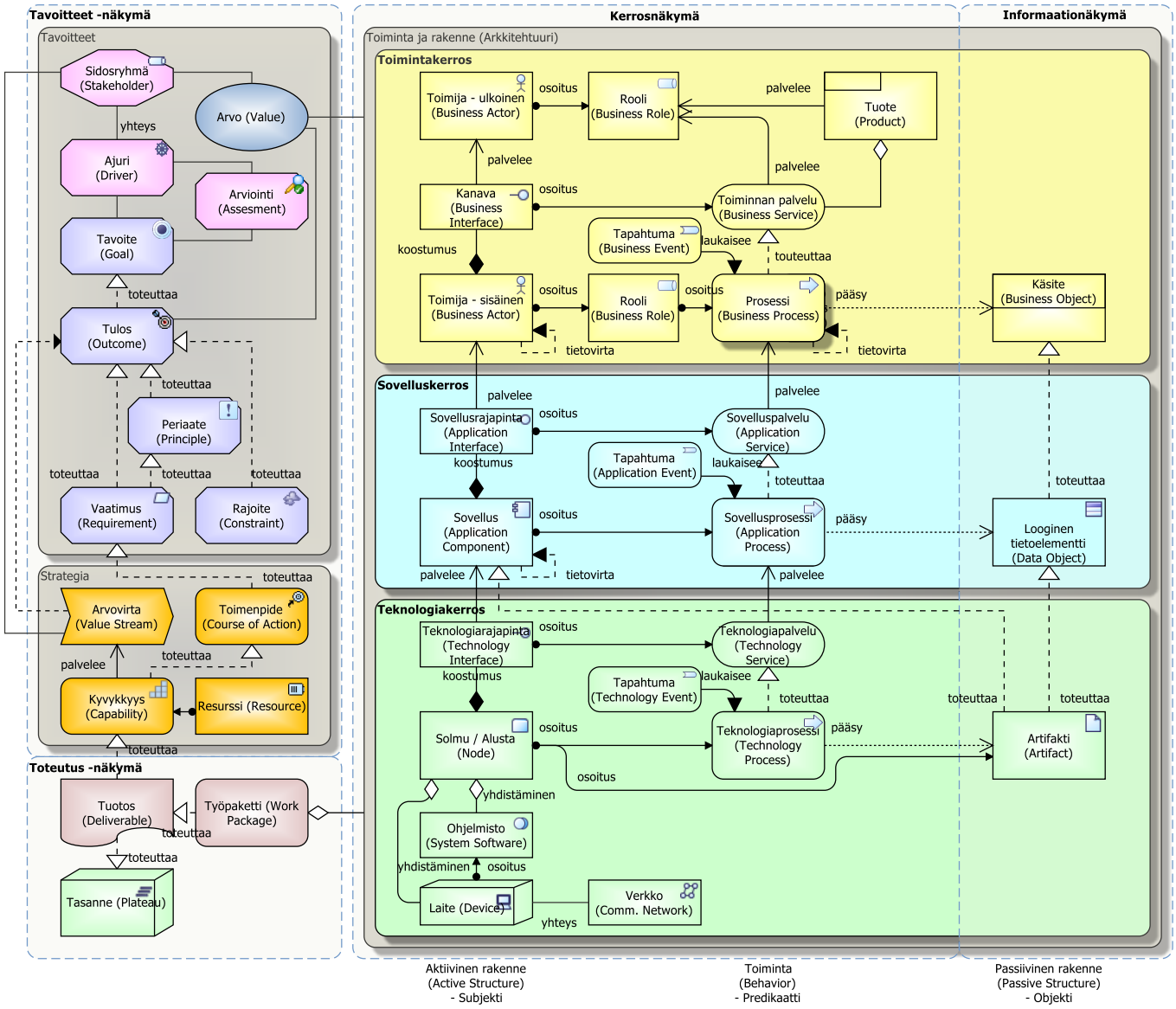
5. Metamalli

5.1 Metamalli (suppea)



Kuva 40: Metamalli (suppea). ArchiMate ydinelementtien osajoukko.

5.2 Metamalli (laajennettu)



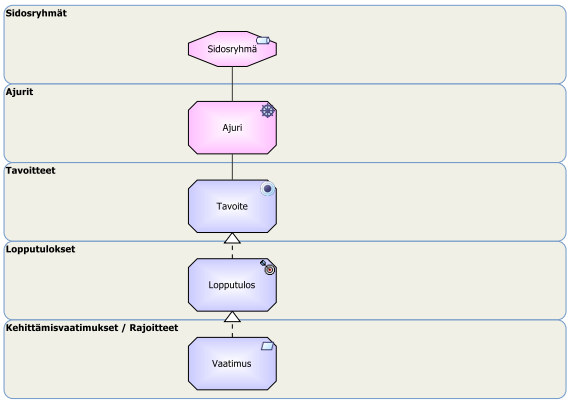
Kuva 41: Metamalli (laajennettu osajoukko).

HUOM! Metamallissa yllä esiintyy rakenteellisen elementin kuten Toimija (Business Actor), Sovellus (Application Component), Solmu (Node) toimintaa kuvaavana elementtinä yksinkertaisuuden vuoksi "prosessi" -elementti kussakin kerroksessa: liiketoimintaprosessi (Business Process), sovellusprosessi (Application Process) ja teknologiaprosessi (Technology Process). Kuitenkin, rakenteellisen elementin toimintaa voidaan mallintaa myös "toiminto" -elementillä kussakin kerroksessa vastaavasti: toiminto (Business Function), sovellustoiminto (Application Function) ja teknologiatoiminto (Technology Function).

6. Kaaviotyypit

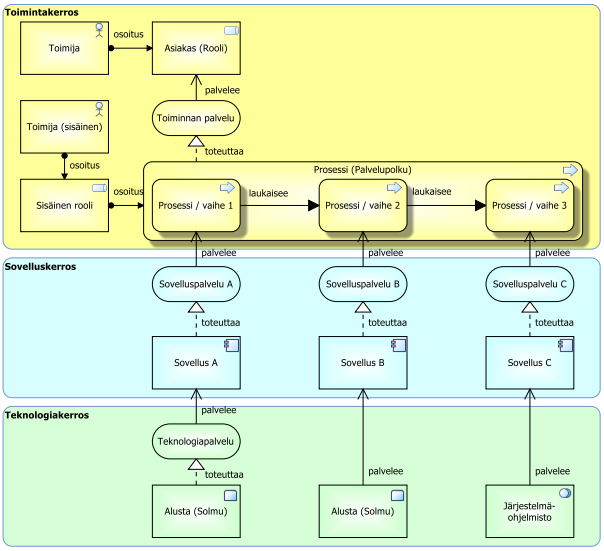
Keskeiset kaaviotyypit ovat:

1. Tavoitteet -näkymä (Goals View)
2. Kerrosnäkymä (Layered View)
3. Vuorovaikutusnäkymä (Interaction View / Co-operation View)
    a. Toimijoiden vuorovaikutus
    b. Prosessien vuorovaikutus
    c. Sovellusten vuorovaikutus



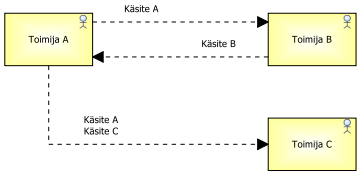
Sekä myös:

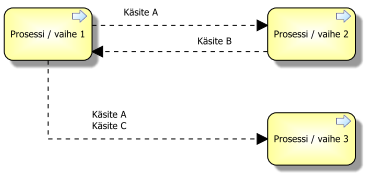
4. Käsitemalli
5. Tietomalli
6. Teknologia-alusta

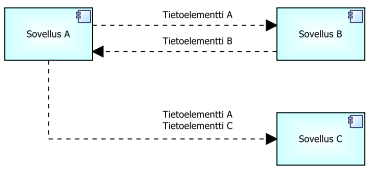


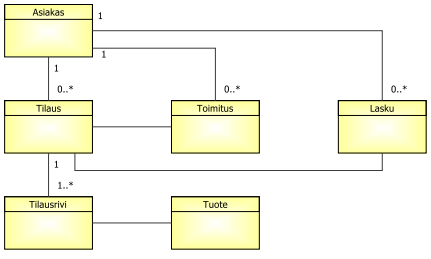
Lisäksi:

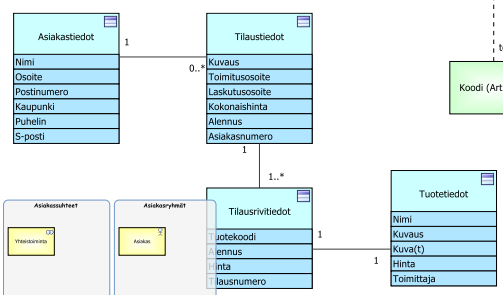
7. Prosessikaavio
8. Business Model Canvas (BMC)
9. Service Blueprint

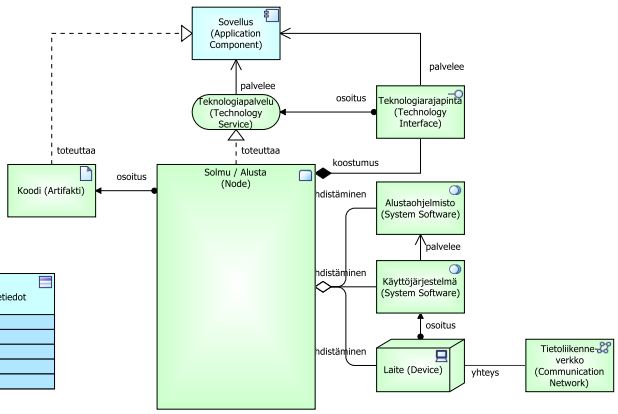


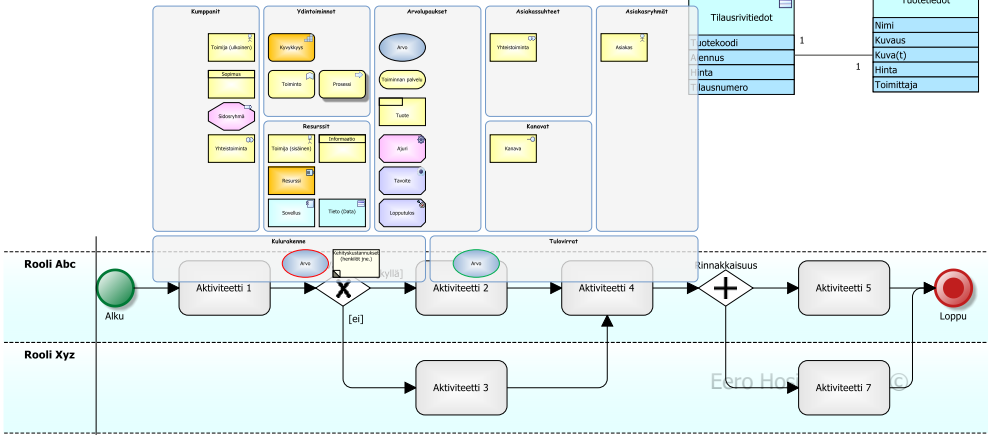


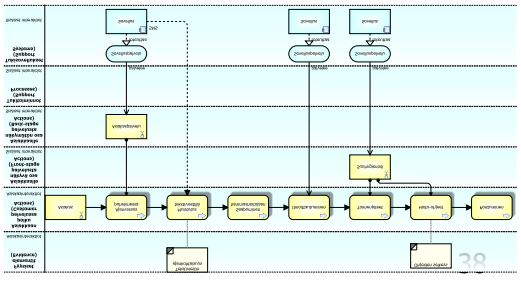












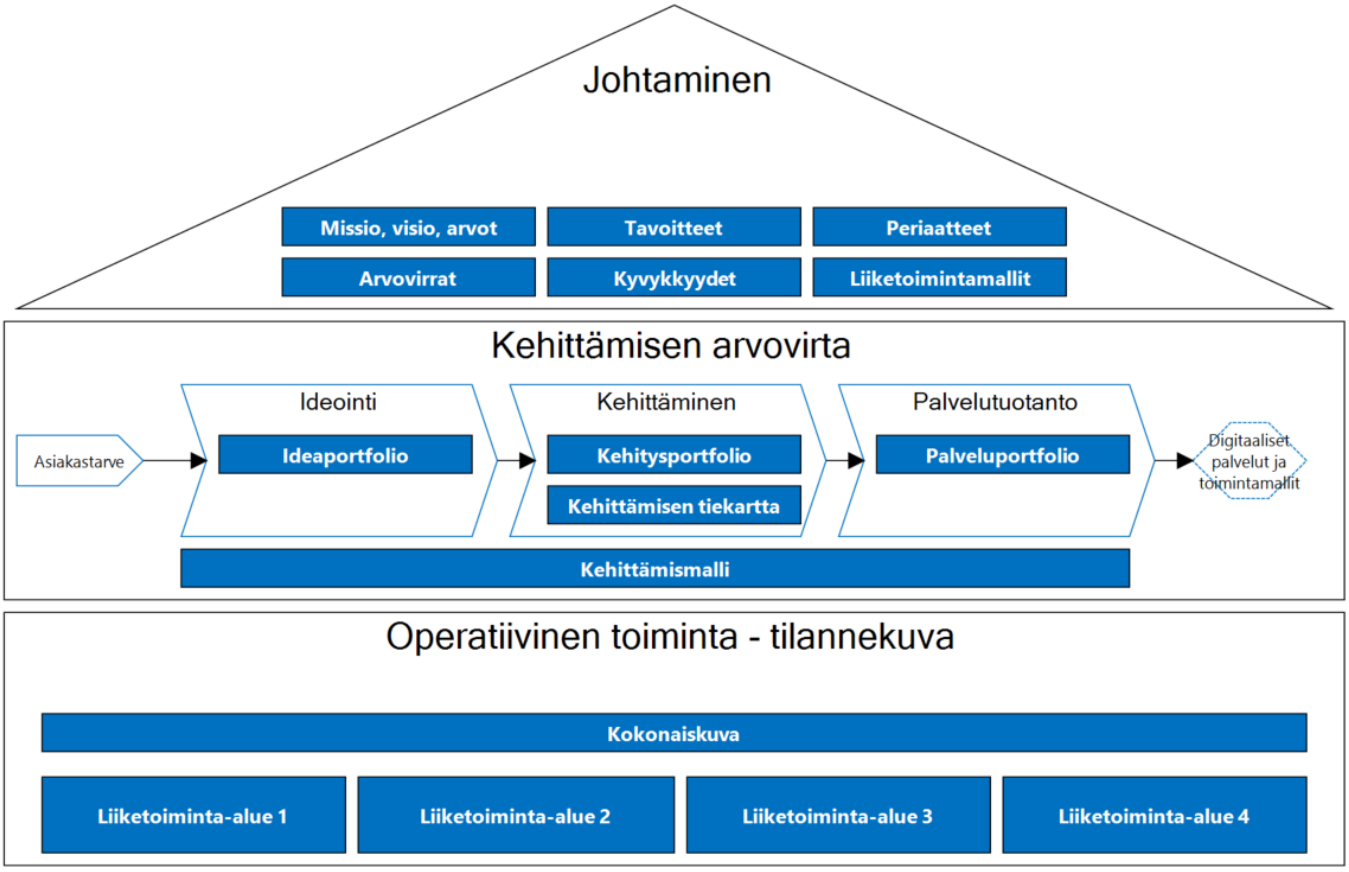
7. Menetelmät

7.1 LeanEA-viitekehys ja kehittämismalli

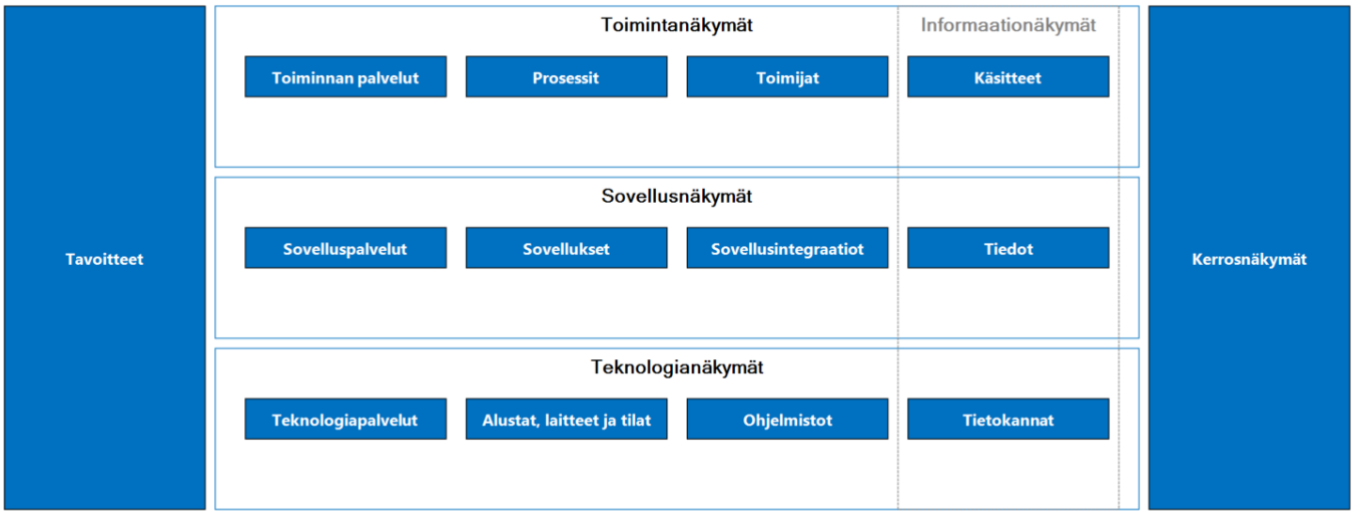
LeanEA-viitekehys tarjoaa uuden tavan tehdä kokonaisarkkitehtuuria. LeanEA-viitekehys tukee kehittämisen toimintamallia eli kehittämismallia, johon myös arkkitehtuuritoiminto on integroitu. Arkkitehtuuri osallistuu kehittämiskohteiden läpivientiin ideasta-tuotantoon heti alusta alkaen, osana moniammatillista virtuaalitiimiä. Arkkitehtuurikuvauksia tuotetaan vain tarvelähtöisesti (just in time) ja vain riittävästi (just enough).

LeanEA-viitekehys jakaantuu kahteen osaan:

1. Etusivu - taso-1 (kuva 40)
   - Sisältää sisältöalueet
     - Johtaminen
     - Kehittämismalli (arvoketjupohjainen toimintamalli)
     - Liiketoiminta-aluekohtaiset arkkitehtuurin sisältöalueet (alasivut 1-n)
2. Alasivu - taso-2 (kuva 41)
   - Sisältää liiketoiminta-aluekohtaiset arkkitehtuurin sisältöalueet.

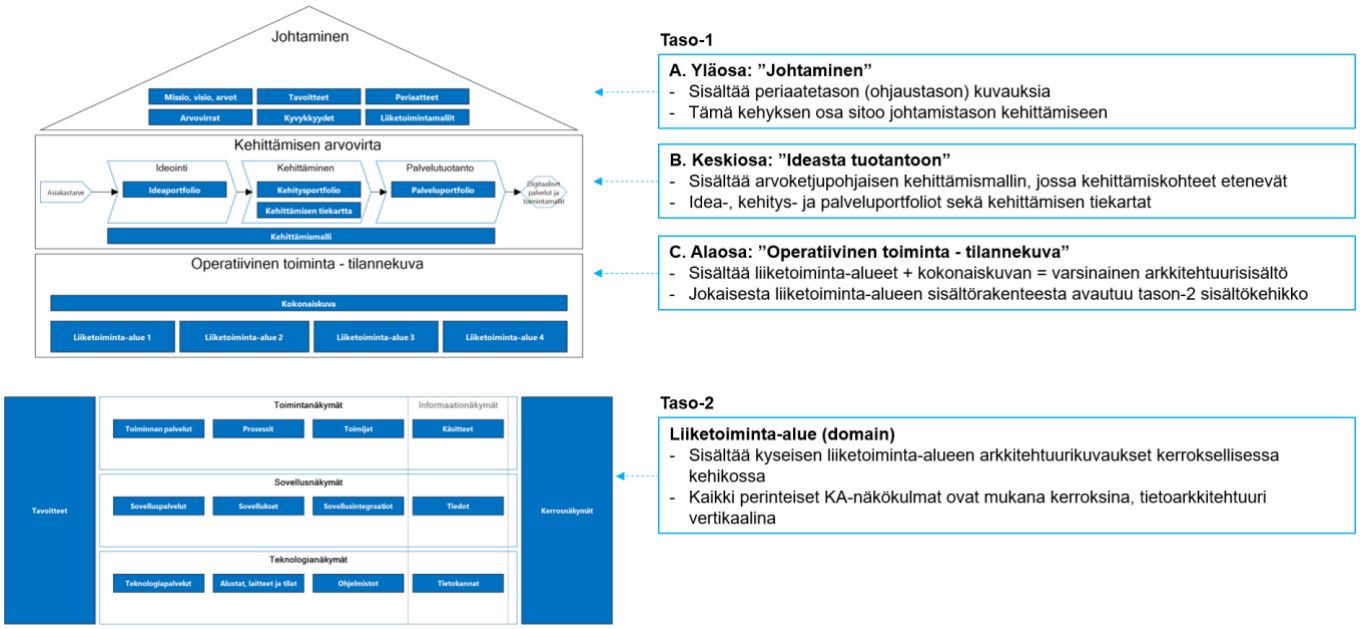


Kuva 42: LeanEA -viitekehys etusivu (taso-1).



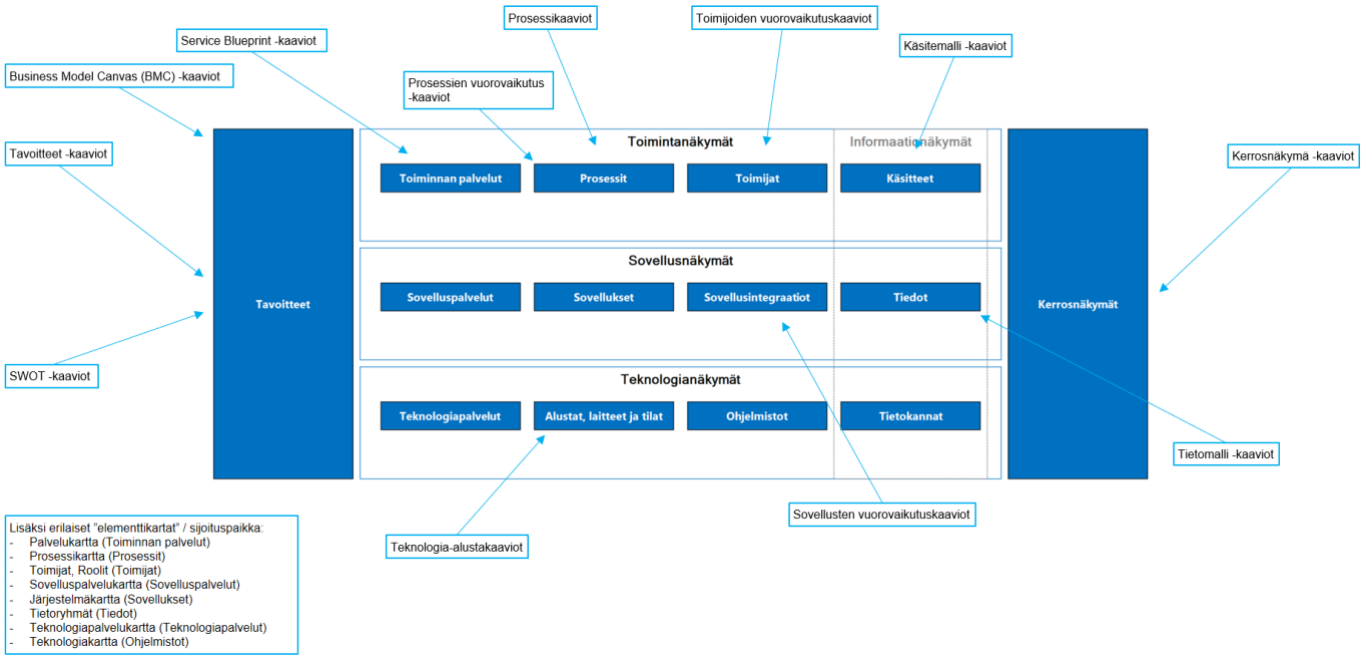
Kuva 43: LeanEA -viitekehys sisältösivu (taso-2).

LeanEA-viitekehyksen taso-1 tukee organisaatiotason kehittämisen johtamista ja hallintaa, sekä kehittämisen toimintamallia. Operatiivinen tilannekuva sisältönä ovat varsinaiset arkkitehtuurisisällöt, eli taso-2.



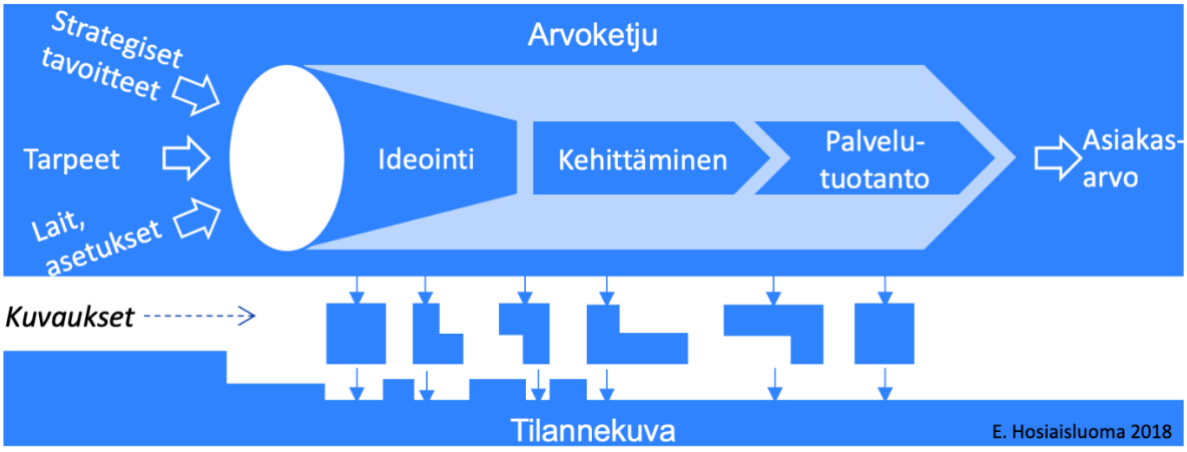
Kuva 44: LeanEA-viitekehys sisältörakenteet.

Tässä dokumentissa esitellyt kaaviotyypit, sekä muut kuvaukset, sijoitetaan LeanEA-viitekehyksen liiketoiminta-aluekohtaisiin sisältörakenteisiin (taso-2) alla olevan kuvan osoittamalla tavalla.



Kuva 45: LeanEA-viitekehys ja kaaviotyyppien sijoittelu.

LeanEA kehittämismalli (*Lean EA Development, LEAD method*) on menetelmäkokonaisuus, jossa koko kehittämisen toimintamalli on integroitu arvoketjuun, jossa kehittämiskohteiden läpivientiä tukee kuvausvälineeseen implementoitu LeanEA-viitekehys. Arkkitehtuuritoiminto toimii osana moniammatillista tiimiä, joka käsittelee kaikki kehittämistarpeet, ja vie ne valmiina konsepteina kehitysportfolioon. Organisaation operatiivisen toiminnan tilannekuva (Architecture Landscape) täydentyy sitä mukaa kun kehittämiskohteita kuvataan mallinnusvälineen kuvauskantaan (Repository).



Kuva 46: Arvoketjupohjainen kehittämisen toimintamalli ja jatkuvasti täydentyvä arkkitehtuurin tilannekuva.

LeanEA-kehittämismallia (*LeanEA Development, LEAD*) ja LeanEA-viitekehystä (*LeanEA Framework, LEAF*) on kuvattu tarkemmin mm. artikkelissa: *Lean Enterprise Architecture Method for Value Chain Based Development in Public Sector, Hosiaisluoma et al., 2018*.
[https://www.researchgate.net/publication/328560027_Lean_Enterprise_Architecture_Method_for_Value_Chain_Based_Development_in_Public_Sector]

7.2 Tavoitelähtöinen suunnittelu ja kehittäminen

Tavoitelähtöinen lähestymistapa (*Goal-Driven Approach, GDA*), tukee kaikkien kehittämiskohteiden suunnittelua. Kysymys on yksinkertaisesti siitä, että ensin keskitytään tavoitteisiin: analysoidaan kenelle, miksi ja mitä ollaan kehittämässä – mitä lisäarvoa kehittämiskohteella tavoitellaan. Elleivät nämä asiat ole selkeästi tunnistettavissa ja kiteytettävissä yhden sivun Tavoitteet -näkymälle, kehittämiskohde eli muutostarve ei todennäköisesti ole selkeä.

Kaiken kehittämisen perustana tulisi olla selkeä käsitys miksi muutos tarvitaan. Siksi olisi hyödyllistä aina ensin, ennen kuin tehdään tai kuvataan yhtään mitään, määritellä muutostarpeen eli kehittämiskohteen varsinaiset taustatekijät ja tavoitteet. Parhaiten tämä onnistuu kysymällä oikeita kysymyksiä, mikä johdattaa tarkkaan kehittämiskohteen juurisyiden analyysiin. Tämä alkuvaiheen analyysi voidaan tehdä hyödyntämällä Tavoitteet -näkymää. Tavoitteet -näkymän avulla voidaan kiteyttää yhteen näkymään kehittämiskohteen ”business case”. Tarvittaessa tätä voidaan täydentää hyödyntämällä Business Model Canvas (BMC) -näkymää.

Tavoitteet-näkymän avulla on mahdollista analysoida, ennen kuin tehdään yhtään mitään, seuraavia asioita:

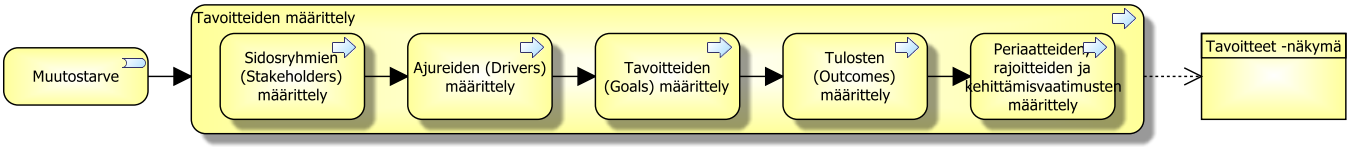
- KENELLE kehittämiskohde on tarkoitettu, mitä sidosryhmiä se koskee, mitkä sidosryhmät ovat kiinnostuneita tästä kehittämiskohteesta, muutoksesta, tai joilla on ylipäätään jokin intressi tähän kehittämistarpeeseen liittyen?
- MIKSI kehittämiskohde tarvitaan, mitkä ovat sen ajurit eli taustavaikuttimet, mitkä ovat varsinaiset ongelmat tai juurisyyt. On erittäin tärkeätä analysoida miksi kehittämistarve on merkityksellinen, miksi tarvitaan muutos nykyiseen asiantilaan?
- MITÄ itse asiassa tavoitellaan, mitä tavoitteita kehittämistarpeeseen liittyy.

  Tavoitteet (Goals) on syytä määritellä sanamuodoiltaan selkeästi ohjaaviksi, sisällyttämällä tavoitteeseen laadullinen määre, kuten esimerkiksi ”asiakastyytyväisyyden *parantaminen”*, ”hakemuskäsittelyn *nopeuttaminen”*, tai *helpottaminen*, ”manuaalityön *vähentäminen”*. Tavoitteille voidaan tunnistaa mitattavissa olevia lopputuloksia (Outcome), kuten esim. ”asiakastyytyväisyyden parantaminen 20%”, tai ”..parantaminen tasolle 4” jne.

  - MITEN tavoitteisiin päästään, voidaan tunnistaa mallintamalla kehittämiskohteeseen liittyviä periaatteita ja rajoitteita sekä kehittämisvaatimuksia.

Vaatimukset voidaan tarvittaessa viedä tarkalle tasolle, ja tunnistaa konkreettisia toiminnallisia- ja ei-toiminnallisia vaatimuksia kehittämiskohteelle.

Tavoitteet-näkymän mahdollistamalla yhden sivun tiivistyksellä havainnollistetaan kehittämistarpeen juurisyyt, tavoitteet, periaatteet ja rajaukset sekä kehittämisvaatimukset. Kaaviotyypillä saadaan aikaan yksinkertainen rautalankamalli kehittämistarpeesta, kaiken tekemisen pohjaksi. Tässä mielessä Tavoitteet -näkymät ovat erittäin hyödyllisiä jokaisen kehittämiskohteen osalta, sillä niiden avulla tehdään näkyväksi ja arvioitavaksi ennen kaikkea MIKSI kyseinen kehittämiskohde on merkityksellinen, ja mikä on sen tuottama lisäarvo.

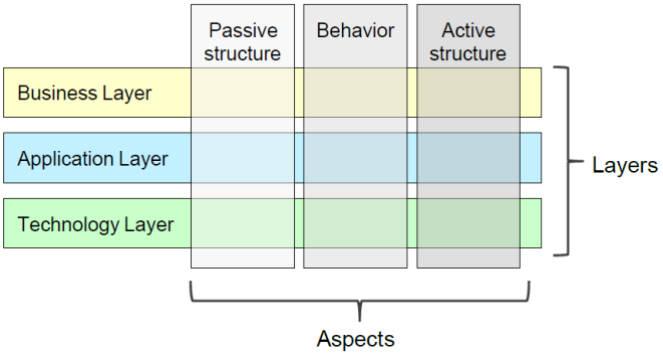


Kuva 47: Tavoitelähtöinen kehittäminen - aina liikkeelle tavoitteiden määrittelystä (KENELLE, MIKSI, MITÄ).

7.3 Palvelulähtöinen suunnittelu ja kehittäminen

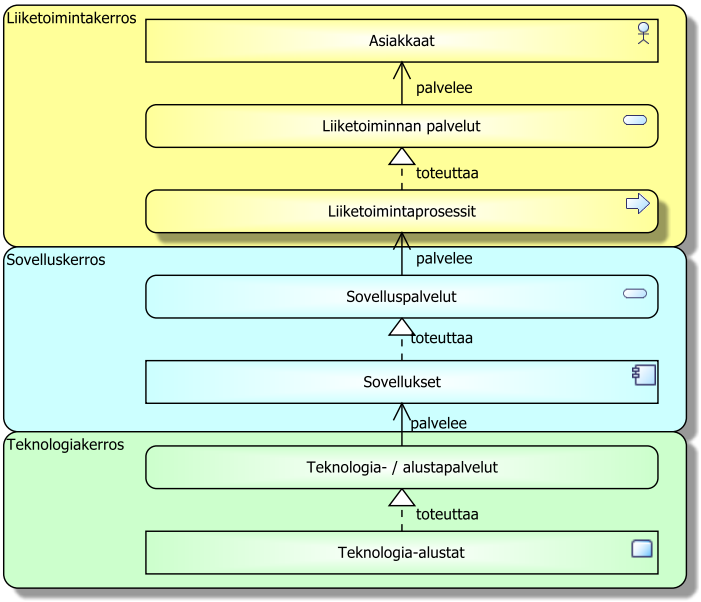
Palvelulähtöinen lähestymistapa (*Service-Driven Approach, SDA*) ottaa lähtökohdaksi eri kerrosten tarjoamat palvelut. Palveluita on kolmea (3) tyyppiä: 1) Liiketoiminnan palvelu (Business Service), 2) Sovelluspalvelu (Application Service) ja 3) Teknologia- eli alustapalvelu (Technology Service / Infrastructure Service).

Palvelulähtöisyys perustuu kerrokselliseen lähestymistapaan, jossa palvelut kytkevät eri kerrosten elementit toisiinsa. Kehittämiskohdetta tarkastellaan palveluiden kautta. Palvelut ovat liiketoiminnan kannalta merkityksellisiä toiminnallisia asioita, jotka antavat merkityksen niitä toteuttaville rakenteellisille osille. Tässä mielessä onkin syytä ymmärtää kehittämiskohteen toiminnalliset ja rakenteelliset aspektit. ArchiMate-notaation rakenne ja se eri elementtityypit perustuvat toiminnan ja rakenteen erottamiseen. ArchiMate-notaation perusrakenne pohjautuu sekä a) kerroksellisuuteen (*Layers*) että b) toiminnallisten ja rakenteellisten elementtien (*Aspects*) tunnistamiseen (kuva alla). Palvelut kytkevät nämä kerrokset toisiinsa.



Kuva 48: ArchiMate-rakenne jakaa elementit toimintaan (Behavior) ja rakenteeseen (Active structure, Passive structure) kussakin kerroksessa (Layer).

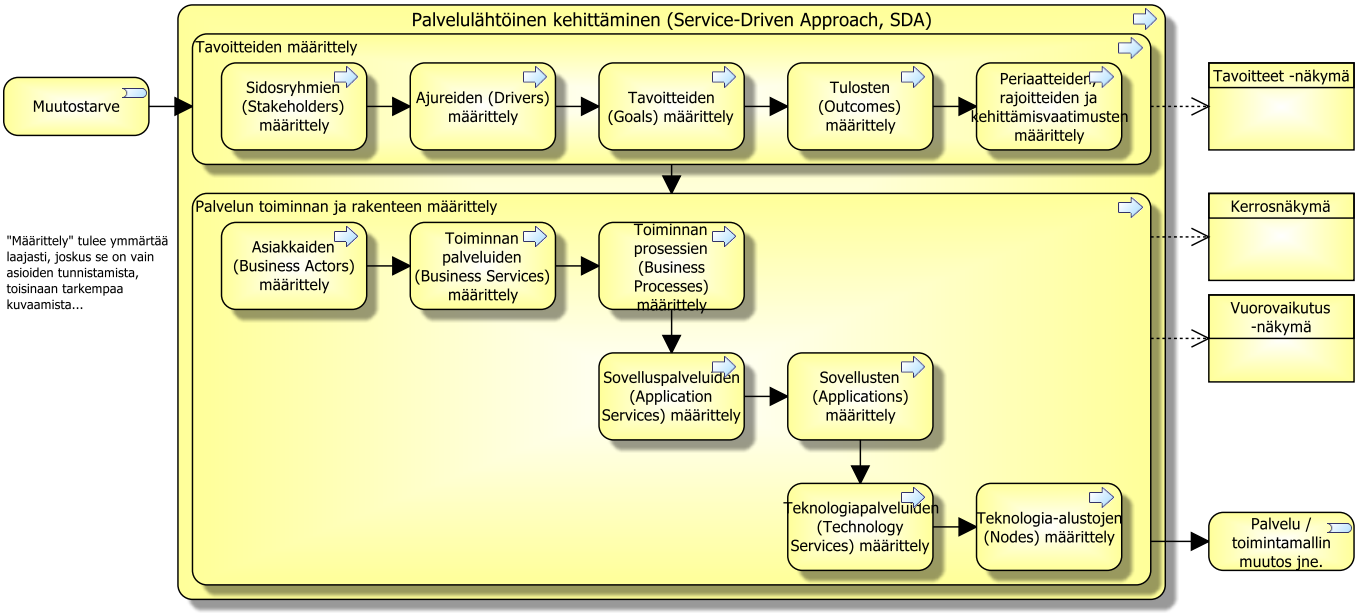
Alla on esitetty yksinkertaistettu ”kaava” palvelulähtöisestä ja kerroksellisesta lähestymistavasta, jonka avulla voidaan tunnistaa palveluita, sekä muita kehittämiskohteen toiminnallisia ja rakenteellisia osia.



Kuva 49: Palvelulähtöisen kehittämisen ylätason kuva.

Palvelulähtöinen kehittäminen tarkastelee kehittämiskohdetta palveluiden kautta: palvelu on keskeinen käsite. Palveluita tarjotaan asiakkaille, palveluita tuotetaan organisaation toiminnallisilla ja rakenteellisilla resursseilla kuten prosesseilla, sovelluspalveluilla, sovelluksilla ja alusta- eli teknologiapalveluilla. Kehittämiskohde voi olla koko organisaatio tai jokin sen osa-alue.

7.3.1 Service-Driven Approach (SDA)



Kuva 50: Palvelulähtöinen kehittäminen (Service-Driven Approach, SDA).

Palvelulähtöinen kehittäminen alkaa tavoitteiden tunnistamisella, kuten kaikkien kehittämiskohteiden kehittämisen tulisi alkaa. Sitten tarkastellaan kehittämisen kohteena olevaa toiminnan palvelua, mm.: mitä asiakasryhmiä se palvelee ja miten palvelua tuotetaan. Palvelun suunnittelua ja kehittämistä sekä operatiivista hallintaa tukee asianmukainen mallinnusväline. Sen avulla tehdyt kuvaukset julkaistaan kaikille keskeisille sidosryhmille, jatkuvasti. Näin kokonaisvaltainen kehittäminen ja kaikki oleelliset kuvaukset ovat hyödynnettävissä, koko palvelun elinkaaren ajan.

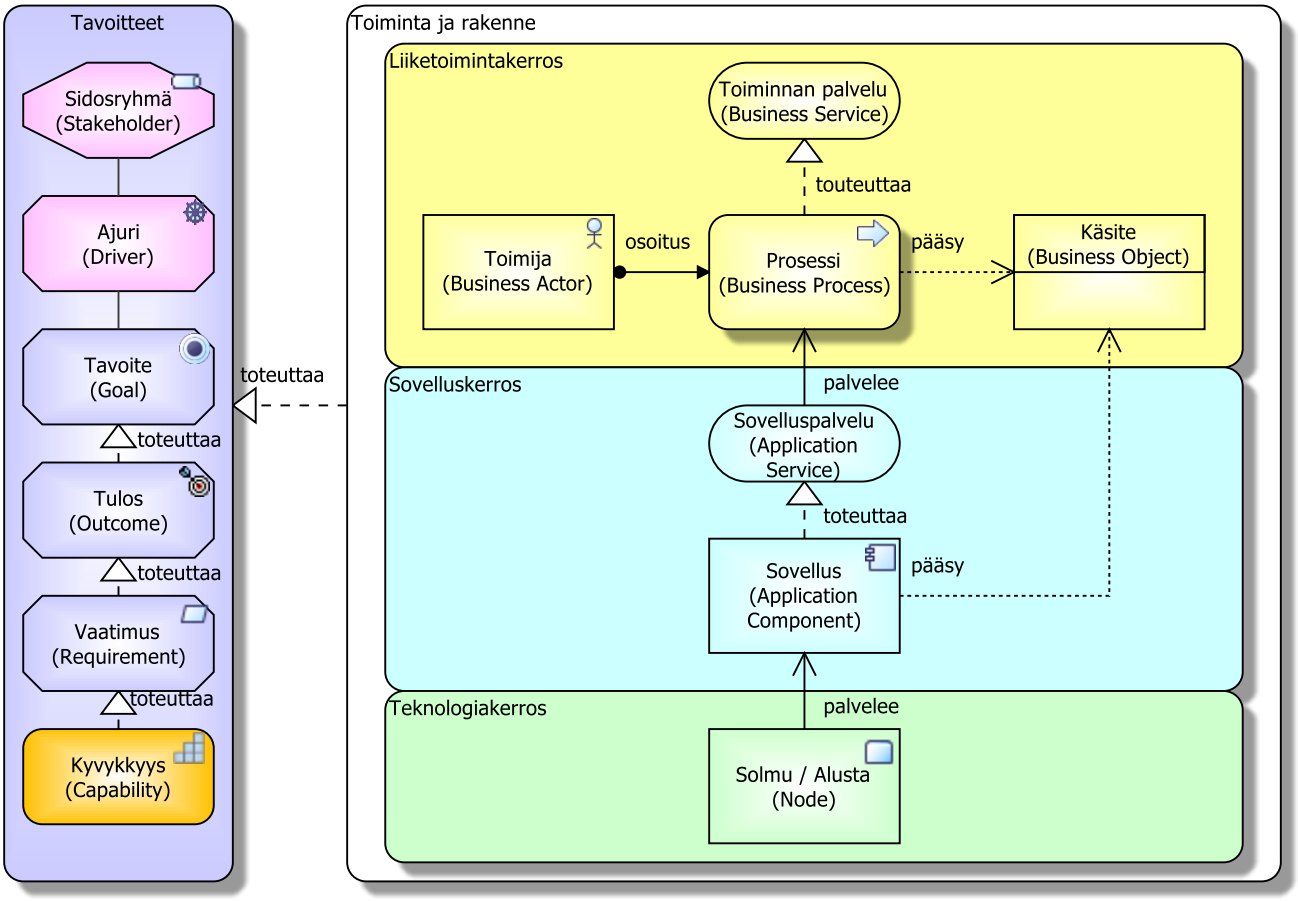
Palvelulähtöisessä kehittämisessä määritellään ja kuvataan mallinnusvälineeseen seuraavia asioita:

1. Tavoitteet, hyödynnetään Tavoitteet -näkymää (Goals View), tarvittaessa Business Model Canvas:ia
2. Palvelun asiakkaat
3. Liiketoiminnan palvelu(t), voidaan hyödyntää Palvelupolku- tai Service Blueprint -kaavioityyppejä
4. Liiketoiminnan prosessit tai -toiminnot, sekä niihin kytkeytyvät sisäiset toimijat, jotka yhdessä varsinaisesti tuottavat palvelun
5. Sovelluspalvelut, joita palvelun tuottamisessa käytetään
6. Sovellukset, joilla tuotetaan sovelluspalvelut
7. Teknologia-alustapalvelut ja
8. Teknologia-alustat (alustaohjelmistot, palvelimet ym. verkkokomponentit) tarpeen mukaan.

7.4 ArchiMate 1-2-3

Kuva 51: ArchiMate 1-2-3.

ArchiMate 1-2-3 on yksinkertaistettu ja helposti omaksuttava lähestymistapa, kokonaisarkkitehtuurin mallintamisen ”blueskaava”. Se perustuu pieneen osajoukkoon ArchiMate-elementtejä, joilla pääsee hyvin alkuun. Tarvittaessa elementtejä voidaan ottaa käyttöön helposti lisää. ArchiMate 1-2-3 tarkoittaa kolmea kerrosta: ”Liiketoimintakerros”, ”Sovelluskerros” ja ”Teknologiakerros”, sekä niitä laajentavia kahta näkökulmaa: ”Tavoitteet” ja ”Kerrosnäkymät”, jotka kaikki yhdessä muodostavat yhden kokonaisuuden. ArchiMate 1-2-3 lähestymistavan tavoitteena on tarjota valmiiksi jäsennelty elementtijoukko, jonka avulla arkkitehtuurin mallintaminen olisi mahdollisimman yksinkertaista ja helposti omaksuttavissa. On parempi aloittaa pienesti, helposti ja yksinkertaisesti, ja sitten tarvittaessa laajentaa.



Kuva 52: ArchiMate 1-2-3 metamalli.

ArchiMate 1-2-3 perustuu samoihin kaaviotyyppeihin, jotka on esitelty tässä dokumentissa. Liiketoiminta-, sovellus- ja teknologiakerrokset koostuvat ArchiMate:n ydinelementtien osajoukosta, ja Tavoitteet-osio koostuu ArchiMate:n Motivation-aspektin ja Strategy-kerroksen elementtien osajoukosta. Elementit on esitetty oheisessa metamallissa.

8. Liitteet

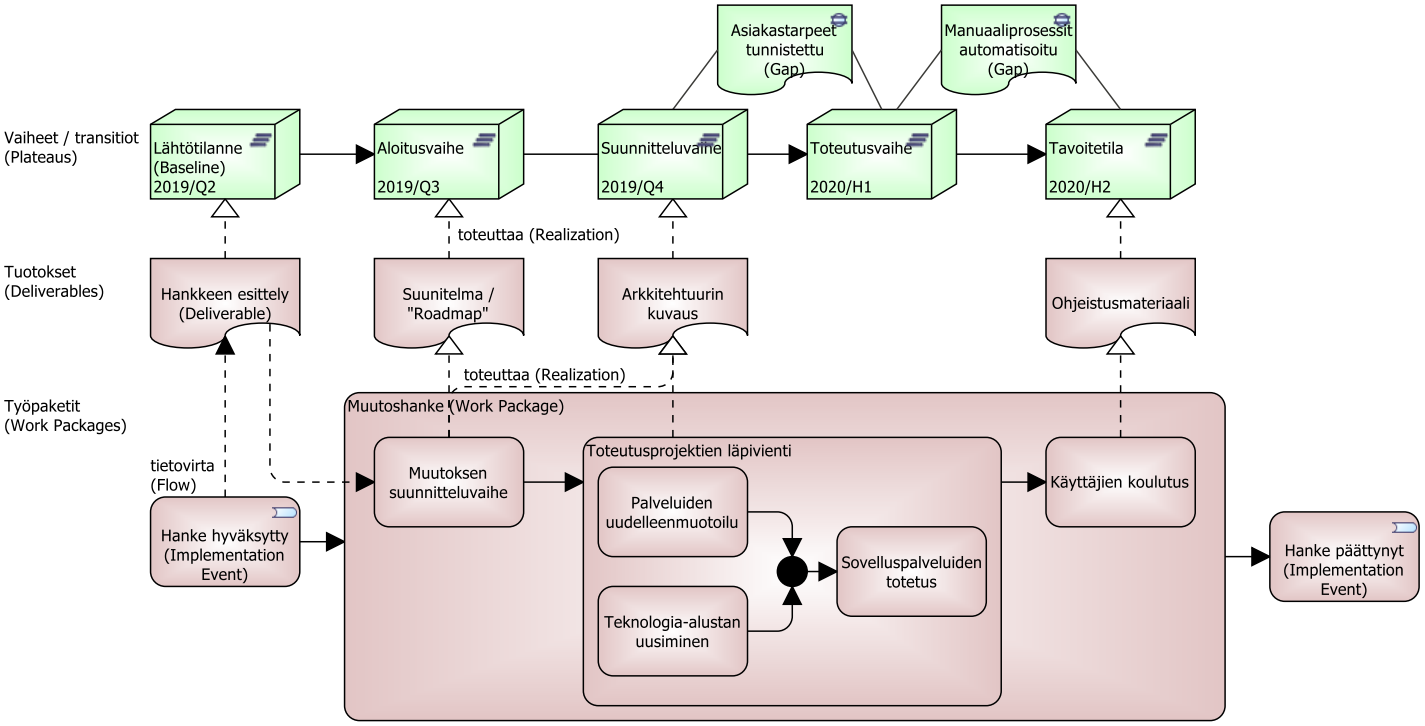
8.1 LIITE 1: Pilvipalvelumallit

Kuva 53: Pilvipalvelumallit.

8.2 LIITE 2: Muutosten toimeenpano

Strategian toimeenpano- ja toteutusnäkymät.

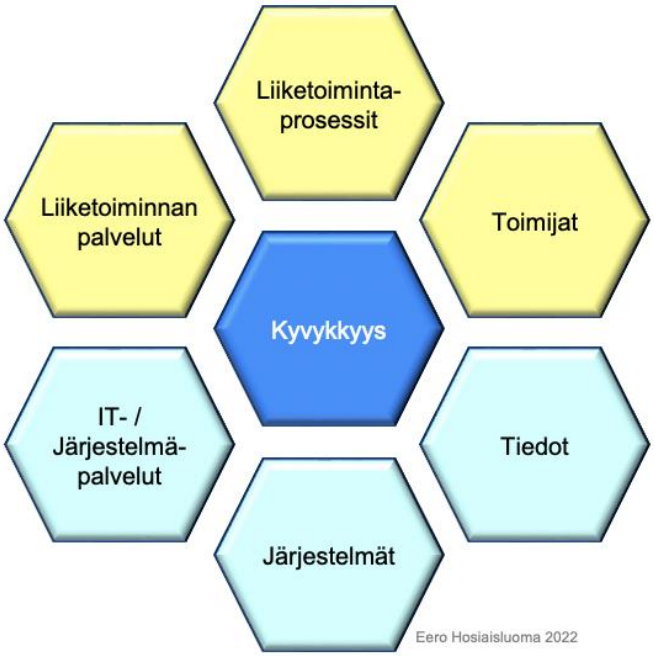
8.2.1 Tiekartta -näkymä



Kuva 54: Tiekartta (Roadmap) näkymä - esimerkki.

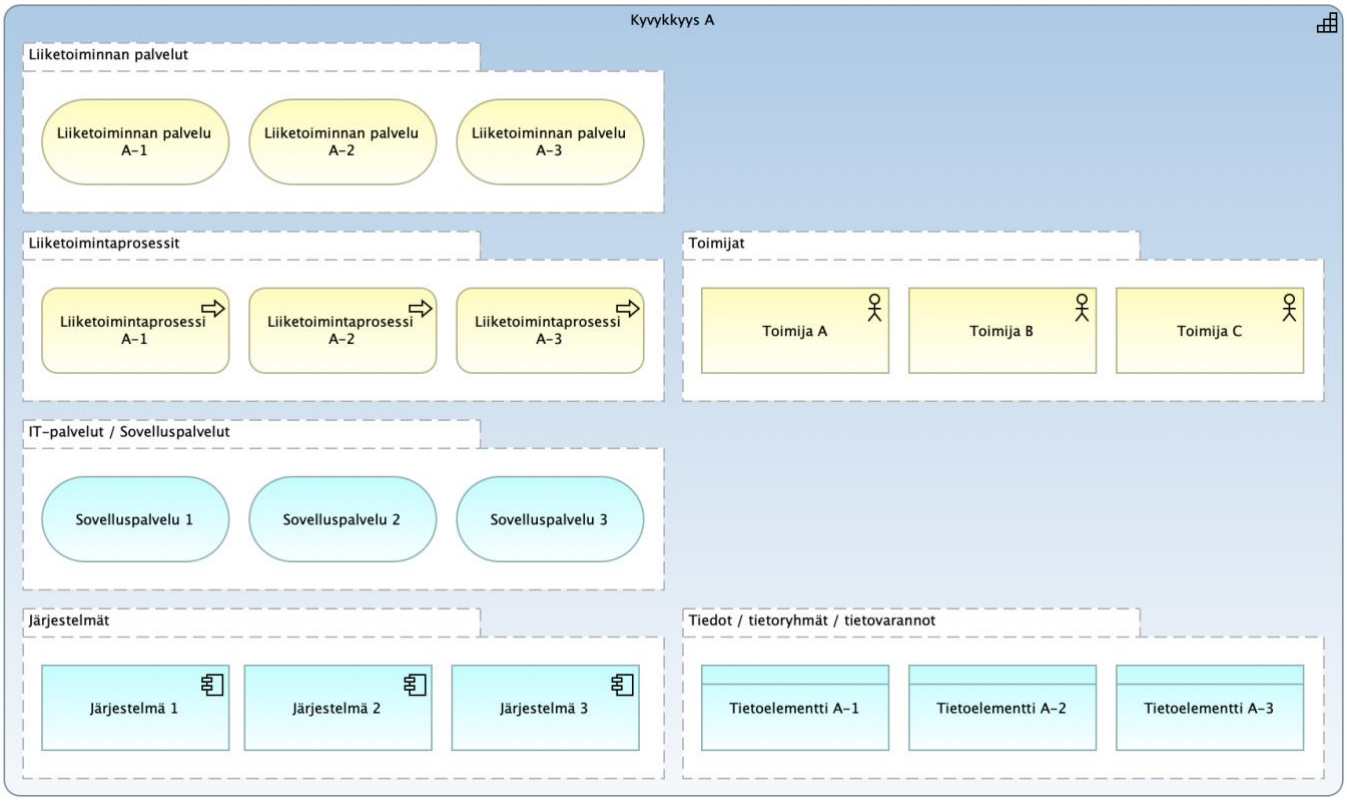
Tiekartta -näkymä (Roadmap) laaditaan ArchiMate Implementation and Migration -elementeillä: *siirtymävaihe* eli transitioarkkitehtuuri (Plateau), *puute* (Gap), *tuotos* (Deliverable), *työpaketti* (Word Package) ja *toteutustapahtuma* (Implementation Event). Yhteystyyppeinä käytetään *laukaisu* (Trigger), *toteuttaa* (Realization) ja *tietovirta* (Flow).

8.3 LIITE 3: Kyvykkyyden anatomia



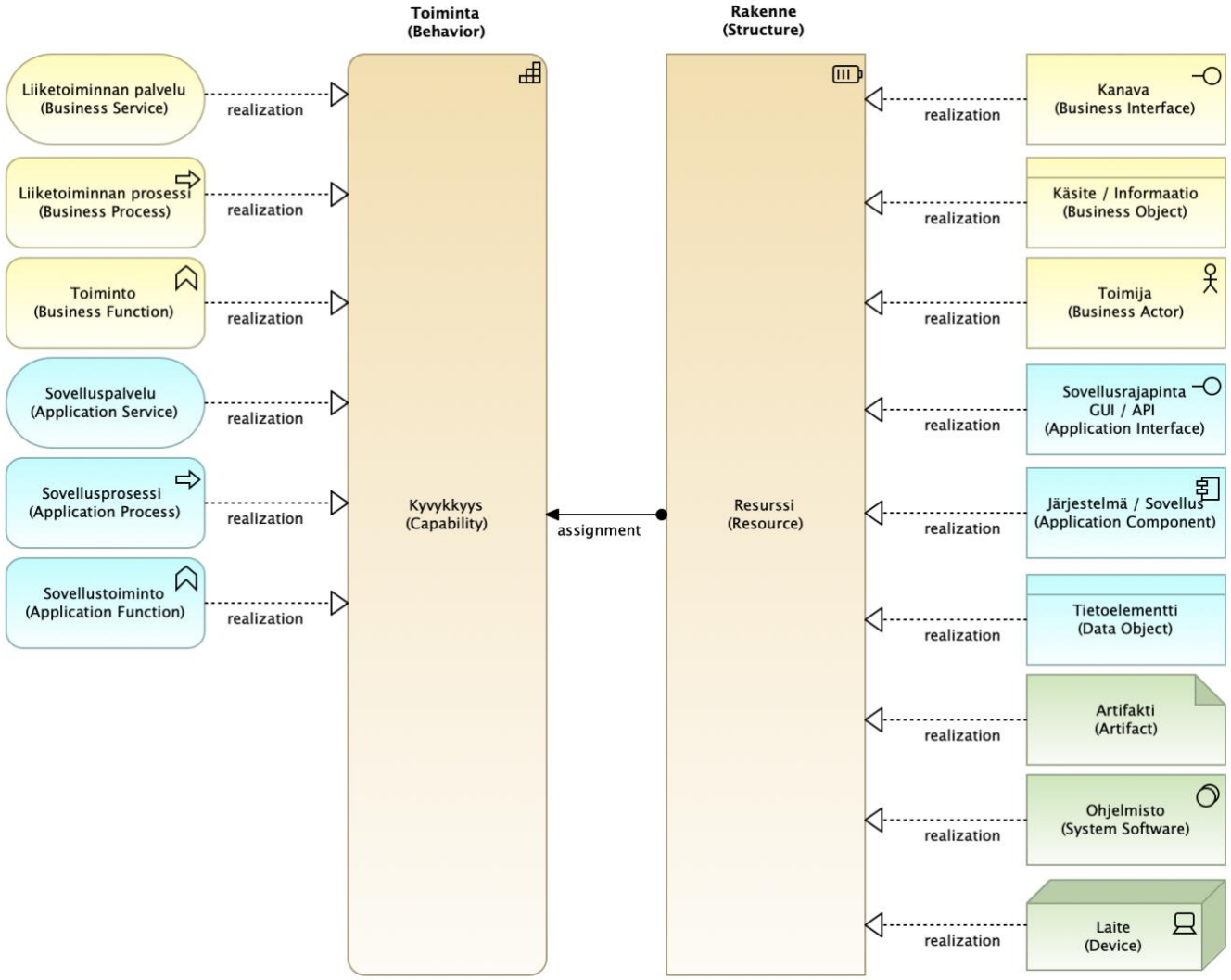
Kuva 55: Kyvykkyyden anatomia.

Kyvykkyys kapseloi toiminnallisia ja rakenteellisia elementtejä. Kyvykkyys on lähtökohtaisesti organisaation toiminnallinen (*behavioral*) osa-alue. Se edustaa jotain toiminnallista asiaa, joka organisaatiossa tulee olla, jotta se voi toteuttaa liiketoimintaansa.



Kuva 56: Kyvykkyyden mallintaminen.

Kyvykkyyden toteuttamiseen tarvitaan resursseja, jotka ovat organisaation rakenteellisia (*structural*) elementtejä. Tässä. Mielessä resurssit ovat kyvykkyyteen kytkeytyviä, mutta siitä erillisiä. Yksinkertaisuuden vuoksi resurssit voidaan mallintaa osaksi kyvykkyyttä.



Kuva 57: Kyvykkyys ja resurssit.