# Rajapintastandardien vaikutuksista järjestelmäkehityksessä

TURUN YLIOPISTO
Tietotekniikan laitos
Pro Gradu -tutkielma
Tietojenkäsittelytiede
Joulukuu 2024
Markus Oravainen

Ohjaajat:
Seppo Helle
Petri Sainio

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

TURUN YLIOPISTO
Tietotekniikan laitos

Markus Oravainen: Rajapintastandardien vaikutuksista järjestelmäkehityksessä
Pro Gradu 56 s., 5 liites.
Tietojenkäsittelytiede
Joulukuu 2024

---

Tämä tutkielma tutkii rajapintastandardien vaikutuksia ohjelmistokehityksen eri vaiheisiin, kuten arkkitehtuuriin, toteutukseen, testaukseen ja käyttöönottoon. Erityistä huomiota kiinnitetään standardien rooliin käytettävyyden, järjestelmän luotettavuuden ja kyberturvallisuuden parantamisessa. Rajapintastandardit mahdollistavat järjestelmien ja sovellusten yhteentoimivuuden sekä tehokkaan tiedonsiirron eri sektoreilla, kuten terveydenhuollossa, finanssialalla, koulutuksessa ja esineiden internetissä.

Rajapintastandardit ratkaisevat yhteentoimivuus- ja tiedonsiirtohaasteita tarjoamalla selkeät säännöt ja suuntaviivat, jotka tukevat suunnittelua, modulaarista arkkitehtuuria, tehokasta toteutusta ja automatisoitua testausta. Standardit ohjaavat intuitiivisten ja käyttäjäystävällisten järjestelmien suunnittelua, mutta voivat myös rajoittaa joustavuutta ja luovuutta. Lisäksi ne parantavat järjestelmien luotettavuutta ja kyberturvallisuutta tukemalla suojausprotokollia, kuten salauksia ja haavoittuvuuksien hallintaa. Tutkielma tarjoaa kattavan näkemyksen siitä, miten standardit muokkaavat ohjelmistokehitystä ja tukevat päätöksentekoa nykyaikaisissa ekosysteemeissä.

Asiasanat: rajapintastandardit, ohjelmistokehitys, yhteentoimivuus, käytettävyys, kyberturvallisuus, innovaatiot, järjestelmien luotettavuus

UNIVERSITY OF TURKU
Department of Computing

Markus Oravainen: Rajapintastandardien vaikutuksista järjestelmäkehityksessä
Master's Thesis 56 p., 5 app. p.
Computer Science
December 2024

---

This thesis examines the impact of interface standards on various stages of software development, such as architecture, implementation, testing, and deployment. Special attention is given to their role in enhancing usability, system reliability, and cybersecurity. Interface standards enable interoperability and efficient data transfer across sectors like healthcare, finance, education, and the Internet of Things.

Interface standards address challenges in interoperability and data transfer by providing clear rules and guidelines that support design, modular architecture, efficient implementation, and automated testing. They guide the development of intuitive and user-friendly systems but may also limit flexibility and creativity. Furthermore, they improve system reliability and cybersecurity by supporting security protocols, such as encryption and vulnerability management. This thesis offers a comprehensive view of how standards shape software development and inform decision-making in modern ecosystems.

Keywords: interface standards, software development, interoperability, usability, cybersecurity, innovations, system reliability

# Lyhenteet

| Lyhenne | Selitys |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| CDA | Clinical Document Architecture |
| CLI | Command Line Interface |
| CoAP | Constrained Application Protocol |
| DLT | Distributed Ledger Technology |
| FDX API | Financial Data Exchange API |
| FHIR | Fast Healthcare Interoperability Resources |
| GUI | Graphical User Interface |
| HDMI | High-Definition Multimedia Interface |
| HL7 | Health Level Seven |
| HTTP | Hypertext Transfer Protocol |
| IoT | Internet of Things |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| LMS | Learning Management System |
| LRS | Learning Record Store |
| LTI | Learning Tools Interoperability |
| ML | Machine Learning |
| MQTT | Message Queuing Telemetry Transport |
| REST | Representational State Transfer |
| RIM | Reference Information Model |
| SCORM | Sharable Content Object Reference Model |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| USB | Universal Serial Bus |
| UX | User Experience |
| XML | Extensible Markup Language |
| xAPI | Experience API |

# Sisällys

**1 JOHDANTO** **1**

**2 Rajapintojen ja standardien perusteet** **3**

2.1 Käyttöliittymät . . . 4

2.1.1 Komentoliittymä . . . 4

2.1.2 Graafinen käyttöliittymä . . . 5

2.2 Sovellusohjelmointirajapinnat (API) . . . 6

2.3 Rajapintastandardien merkitys ja käyttö . . . 7

**3 Rajapintastandardointi eri aloilla** **9**

3.1 Terveydenhuolto . . . 9

3.1.1 HL7 v2 . . . 10

3.1.2 HL7 v3 . . . 12

3.1.3 HL7 FHIR . . . 13

3.1.4 HL7 CDA . . . 14

3.2 Finanssiala . . . 15

3.2.1 SWIFT . . . 16

3.2.2 ISO 20022 . . . 17

3.2.3 2010-luvun innovaatiot . . . 17

3.2.4 Standardien tavoitteet . . . 18

3.2.5 Tulevaisuuden mahdolliset muutokset . . . 19

3.3 Opetus . . . 20

3.3.1 Merkitys . . . 20

3.3.2 Tavoitteet . . . 21

3.3.3 Hyödyt . . . 21

3.3.4 Haasteet . . . 22

3.3.5 SCORM . . . 22

3.3.6 Learning Tools Interoperability . . . 23

3.3.7 xAPI . . . 23

3.4 Internet of Things . . . 24

3.4.1 Standardoinnin tärkeys . . . 24

3.4.2 Haasteet . . . 25

3.4.3 MQTT . . . 25

3.4.4 CoAP . . . 26

3.5 Käytettävyyden standardisointi . . . 27

3.5.1 Nielsenin 10 heuristiikkaa . . . 27

3.5.2 TURF . . . 29
3.5.3 ISO 9241-11 . . . 30
3.6 Kyberturvallisuuden standardointi . . . 31
3.6.1 OAuth 2.0 . . . 31
3.6.2 PKI . . . 32
3.6.3 AES . . . 33

**4 Kysely ja haastattelut** **34**
4.1 Kysymykset . . . 34
4.2 Haastattelu . . . 36

**5 Vaikutukset järjestelmän eri alueilla** **38**
5.1 Vaikutukset kehityksen eri vaiheisiin . . . 38
5.1.1 Suunnittelu . . . 39
5.1.2 Arkkitehtuuri . . . 40
5.1.3 Toteutus . . . 41
5.1.4 Testaus ja laadunvarmistus . . . 42
5.1.5 Käyttöönotto ja toiminta . . . 43
5.2 Vaikutukset käytettävyyteen . . . 44
5.2.1 Suunnittelijoiden rooli . . . 46
5.2.2 Kehittäjäkokemus . . . 47
5.3 Vaikutus järjestelmän luotettavuuteen . . . 48
5.3.1 Hyödyt ja riskit . . . 49
5.3.2 Tiedon ylläpitäminen . . . 50
5.4 Vaikutukset kyberturvallisuuteen . . . 51
5.4.1 Suojausprotokollat ja -standardit . . . 52
5.4.2 Turvallinen viestien käsittely . . . 52
5.4.3 Haavoittuvuuksien hoitaminen . . . 53

**6 Yhteenveto** **54**

**Lähteet** **57**

# 1 JOHDANTO

Rajapinnat ovat keskeinen osa nykyaikaisia tietojärjestelmiä, sillä ne mahdollistavat järjestelmien ja sovellusten välisen tiedonsiirron. Ne tarjoavat selkeät säännöt ja toimintamallit, joiden avulla eri teknologiat voivat kommunikoida keskenään. Näiden rajapintojen kautta tapahtuva tiedonsiirto varmistaa, että monimutkaiset järjestelmät toimivat vaivattomasti yhteen. Se parantaa yleistä tehokkuutta ja luotettavuutta.

Rajapintastandardit, kuten HL7 terveydenhuollossa, ISO 20022 rahoitusalalla ja SCORM koulutussektorilla, ovat esimerkkejä standardeista, jotka tukevat yhteensopivuutta eri järjestelmissä. Nämä standardit varmistavat, että data siirtyy yhtenäisesti ja virheettömästi järjestelmien välillä. Se edistää tiedon laatua ja turvallisuutta. Toisaalta standardointi ei ole täysin ongelmatonta; sen mukana voi tulla haasteita, kuten luovuuden rajoittaminen ja yhteensopivuusongelmat eri standardien välillä.

Tässä työssä tarkastellaan, miten rajapintastandardit vaikuttavat ohjelmistokehityksen eri vaiheisiin, kuten ohjelmistoarkkitehtuurisuunnitteluun, koodaukseen, testaukseen ja käyttöönottoon. Tavoitteena on selvittää, kuinka standardointi voi parantaa kehitysprosessien laatua ja turvallisuutta. Lisäksi tutkitaan, mitkä haasteet voivat nousta esiin, kun standardeja käytetään laajasti. Työtä varten laadittiin kysely, joka suunnattiin ohjelmistokehittäjille heidän ajatustensa, tuntemustensa ja kokemustensa kartoittamiseksi rajapintastandardien parissa.

Tutkimuskysymykset ovat seuraavat:

TK1: Mitä ongelmia standardointi on ratkonut ja onko se luonut uusia ongelmia?

TK2: Miten rajapintastandardit vaikuttavat ohjelmistokehityksen prosesseihin ja eri vaiheisiin?

TK3: Millä tavoin rajapintastandardit voivat parantaa käytettävyyttä?

TK4: Miten rajapintastandardit vaikuttavat kyberturvallisuuteen ja järjestelmien luotettavuuteen?

Näiden kysymysten avulla pyritään tarjoamaan kattava ymmärrys siitä, kuinka rajapintastandardit voivat tukea ohjelmistokehitystä ja mitä kompromisseja tai etuja ne tuovat mukanaan eri aloilla. Tässä tutkielmassa käsiteltävät alat – terveydenhuolto, finanssiala,

koulutus ja esineiden internet (IoT eng. Internet of Things) – valittiin niiden erilaisten tarpeiden ja lähestymistapojen vuoksi.

Monialainen tarkastelu hyödyttää tätä tutkielmaa tarjoamalla laajemman näkökulman standardien vaikutuksista. Eri alojen esittely paljastaa, miten standardit mukautuvat yksilöllisiin toimintaympäristöihin ja niiden erityistarpeisiin. Samalla tämä lähestymistapa korostaa standardoinnin universaalia arvoa. Monipuoliset esimerkit osoittavat standardoinnin monimuotoisuuden, mikä vahvistaa tutkielman johtopäätöksiä.

Tämä tutkielma tarkastelee rajapintastandardeja sekä teoreettisesta että käytännön näkökulmasta, keskittyen niiden merkitykseen ohjelmistokehityksessä. Työssä hyödynnetään aiheeseen liittyvää kirjallisuutta sekä haastattelu- ja kyselytutkimusten tuloksia tutkien, miten standardit vaikuttavat ohjelmistokehitysprosessien rakenteeseen ja toimintaan. Tutkimuksellisesti on kiinnostavaa syventää ymmärrystä rajapintastandardien roolista innovaation edistämisessä ja ohjelmistoekosysteemien muovaamisessa tulevaisuudessa. Ohjelmistoekosysteemi on kokonaisuus, joka koostuu ohjelmistokomponenteista, sovelluksista ja alustoista, jotka toimivat vuorovaikutuksessa ja ovat toisistaan riippuvaisia yhtenäisten toimintojen tuottamiseksi.

Tässä tutkimuksessa tarkastellaan rajapintaviestintästandardien vaikutuksia ohjelmistokehitykseen. Johdantoa seuraavassa toisessa luvussa esitellään tutkimuksen tausta ja määritellään keskeiset käsitteet. Kolmannessa luvussa käsitellään standardoinnin vaikutuksia eri toimialojen ratkaisuihin. Neljännessä luvussa kuvataan tutkimuksessa käytetty menetelmä. Viidennessä luvussa syvennytään standardien vaikutuksiin ohjelmistokehityksen eri vaiheissa, kuten ohjelmistoarkkitehtuurissa, toteutuksessa, testauksessa ja käyttöönotossa. Lisäksi analysoidaan standardien merkitystä järjestelmien luotettavuudelle. Tutkimus päättyy johtopäätöksiin, jatkotutkimusehdotuksiin ja tutkimuksen kritiikkiin.

Tämän työn kirjoittamisessa on käytetty OpenAI:n luomaa tekoälyä ChatGPT, kirjoitusasun tarkistamiseen.

# 2 Rajapintojen ja standardien perusteet

Rajapinta itsessään ei ole mikään teknologia tai sen osa. Rajapinta on olemassa eri teknologioiden ja niiden vaikuttajien vuorovaikutuksen seurauksena. Sillä tarkoitetaan sitä väylää, jolla käyttäjät ja sovellukset kommunikoivat. Rajapinnat määrittelevät, miten kommunikointi tapahtuu [1]. Rajapinnat voidaan määritellä työkaluiksi tai konsepteiksi, jotka ohjaavat käyttäjän ja laitteen tai eri laitteiden välistä kommunikointia.

Rajapinnat kattavat laajan alan kaikesta teknologiaan liittyvästä, kuten ohjelmisto-, laitteisto- ja verkkorajapinnat sekä käyttöliittymät. Kaikki nämä mahdollistavat eri järjestelmien sujuvan kommunikaation ja yhteensopivuuden.

Rajapintojen merkitys on korostunut olio-ohjelmoinnin yleistyessä, jossa rajapinnat tarjoavat määritellyt tavat vuorovaikutukseen olioiden välillä. Yleisin tällä hetkellä käytetty olio-ohjelmointikieli on Java. Javassa rajapinta on kokoelma metodeja, joissa muiden kehittäjien ja järjestelmien tarvitsee tuntea ainoastaan syötteiden ja tulosteiden muodot sekä osata käyttää niitä. Java-dokumentaatiossa käytetään esimerkkinä futuristista yhteiskuntaa, jossa autot ovat täysin itseajavia. Autovalmistajien täytyy huolehtia siitä, että autot osaavat pysähtyä, kiihdyttää, kääntyä ja niin edelleen. Toinen yritys taas voi luoda järjestelmiä, jotka käyttävät GPS-dataa liikenteen valvomiseen ja siellä navigoimiseen. Auton valmistajien tulee julkaista auton hallintaan käytettävät rajapinnat, jotta GPS-järjestelmä voi antaa ohjeita autolle ja hallita sen liikkeitä. Kummankaan järjestelmän ei kuitenkaan tarvitse tietää miten kaikki tämä tapahtuu. Riittää kun ohjelma osaa kutsua oikeaa metodia rajapinnan määritelmän mukaisesti. [2]

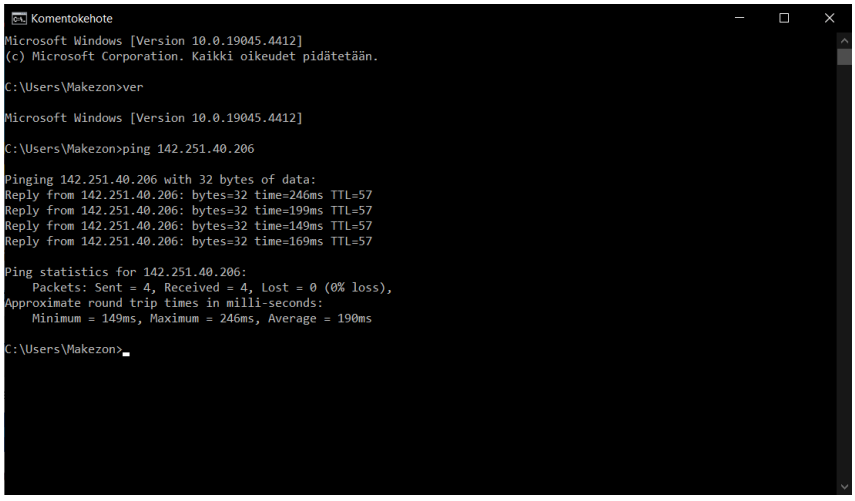
Näitä rajapintoja on kehitetty, jotta käyttäjät oppisivat käyttämään olemassa olevia vaikeasti saavutettavia teknologioita. Helposti ymmärrettävä esimerkki on kaikkien tietokoneiden käyttöliittymät. Ensimmäisten tietokoneiden käyttöliittymät muistuttavat nykyisiä komentokehotteita ja tietokoneiden käyttö tapahtui komentorivien avulla. 1970-luvulla Xerox alkoi kehittää tietokoneisiinsa ensimmäisiä graafisia käyttöliittymiä. Vuonna 1972 Xerox julkaisi ensimmäisen tietokoneen, jossa oli graafinen käyttöliittymä, Xerox Alto. [3] Tämä kehitys aloitti nykyisten graafisten käyttöliittymien synnyn, mikä muutti pysyvästi tapaa, jolla käyttäjät ovat vuorovaikutuksessa teknologian kanssa.

## 2.1 Käyttöliittymät

Käyttöliittymä on järjestelmä tai ohjelmisto, jonka avulla käyttäjä voi kommunikoida elektronisten laitteiden kanssa. Käyttöliittymät ovat keskeisessä asemassa; ne määrittävät, kuinka käyttäjä voi antaa komentoja, syöttää tietoja ja saada palautetta laitteelta. Käyttöliittymät voidaan yleisesti ottaen jakaa kahteen päätyyppiin: graafisiin käyttöliittymiin (GUI) ja komentoliittymiin (CLI).

### 2.1.1 Komentoliittymä

Komentoliiittymä eli CLI (Command Line Interface) oli yksi varhaisimmista käyttöliittymistä, ja se nousi merkittäväksi 1960- ja 1970-luvuilla, kun tietokonepäätteitä otettiin käyttöön. CLI on tekstipohjainen käyttöliittymä (kuva 2.1) jossa käyttäjät antavat komentoja kirjoittamalla ne komentoriville. CLI:n olivat yleisiä varhaisissa tietokonejärjestelmissä ja ovat yhä suosittuja tiettyjen käyttäjäryhmien, kuten järjestelmänvalvojien ja kehittäjien, keskuudessa. CLI:n avulla käyttäjät voivat syöttää komentoja suoraan järjestelmään. Se tarjoaa usein tehokkaamman ja tarkemman tavan hallita järjestelmää verrattuna graafiseen käyttöliittymään (GUI).



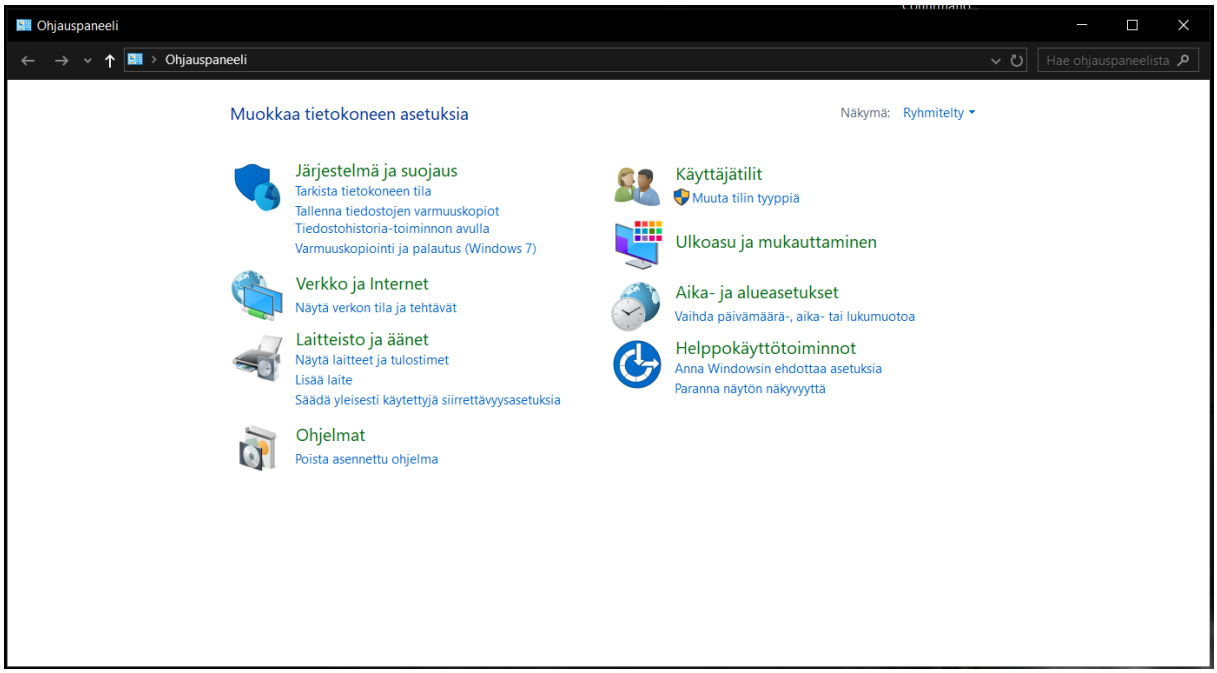

Kuva 2.1: Kuvankaappaus Windows 10 -käyttöjärjestelmän komentoliittymästä

CLI:n etuna on keveys ja suoraviivaisuus. Se vie vähemmän järjestelmäresursseja ja mahdollistaa nopean vuorovaikutuksen kokeneille käyttäjille. Lisäksi CLI tarjoaa usein enemmän hallintamahdollisuuksia ja yksityiskohtaisia asetuksia, joita ei välttämättä ole saatavilla GUI:ssa. CLI:n käyttö vaatii kuitenkin tarkkaa syntaksin tuntemusta ja kokemus-

ta, joka voi tehdä siitä haastavan aloittelijoille. [4] CLI on edelleen käytössä erityisesti järjestelmänvalvojien, kehittäjien ja kokeneiden käyttäjien keskuudessa. Sitä käytetään tehtäviin, jotka vaativat tarkkuutta ja automaatiota, kuten skriptaukseen ja palvelinten hallintaan. [5]

### 2.1.2 Graafinen käyttöliittymä

Graafinen käyttöliittymä eli GUI (Graphical User Interface) mullisti käyttäjien tavan olla vuorovaikutuksessa tietokoneiden kanssa, kun se esiteltiin 1970-luvun lopulla ja 1980-luvun alussa. GUI käyttää graafisia elementtejä, kuten ikkunoita, kuvakkeita, painikkeita ja valikoita, jotka tarjoavat visuaalisen ja semanttisen esityksen toiminnoista ja ohjaimista (kuva 2.2). Tämä esitystapa helpottaa käyttöä, sillä käyttäjien ei tarvitse muistaa jokaista komentoa ulkoa, vaan he voivat navigoida vaihtoehtojen välillä ja valita haluamansa toiminnot intuitiivisesti. Semanttinen esitys auttaa käyttäjiä ymmärtämään toimintojen merkityksen ja vaikutukset selkeämmin, mikä vähentää virheitä ja lisää käytön tehokkuutta. [6] [7] [8]




Kuva 2.2: Kuva Windows 10 -käyttöjärjestelmän ohjauspaneelista

GUI:n suurin etu on sen käyttäjäystävällisyys, joka johtuu visuaalisen esityksen selkeydestä. Esimerkiksi roskakori tiedostojen poistamiseen. GUI ei lähtökohtaisesti vaadi aikaisempaa teknistä osaamista, mikä tekee teknologiasta lähestyttävää laajemmalle yleisölle.

Käyttöliittymä antaa välitöntä visuaalista palautetta, mikä parantaa käyttäjien ymmärrystä heidän toiminnoistaan. GUI:n ominaisuuksiin usein kuuluu nykyaikaisissa järjestelmissä moniajon tukeminen, joka mahdollistaa useiden sovellusten samanaikaisen hallinnan. Visuaaliset elementit yksinkertaistavat vuorovaikutusta käyttäjien ja laitteiden välillä esittämällä tietoa ja ohjaimia helposti lähestyttävässä ja navigoitavassa muodossa. [6] [7] [8]

Perinteisten komentorivikäyttöliittymien ja graafisten käyttöliittymien lisäksi luonnolliset käyttöliittymät ovat nousemassa intuitiivisempana tapana, jolla käyttäjät voivat vuorovaikuttaa järjestelmien kanssa. Nämä käyttöliittymät hyödyntävät ihmisen eleitä, ääni komentoja ja jopa silmäliikkeitä mahdollistamaan hands-free vuorovaikutuksen, tarjoten sujuvamman ja esteettömämmän käyttökokemuksen. Elepohjaiset käyttöliittymät perustuvat liikesensoreihin, jotka tulkitsevat fyysisiä liikkeitä, kun taas ääni käyttöliittymät, kuten virtuaaliavustajat, hyödyntävät puheentunnistusteknologioita äänen komentoihin vastaamiseksi.

## 2.2 Sovellusohjelmointirajapinnat (API)

Sovellusohjelmointirajapinta (API eng. Application Programming Interface) toimii rakenteellisena tapana eri ohjelmistosovellusten väliselle viestinnälle. API tarjoaa joukon määriteltyjä sääntöjä ja protokollia, joiden avulla yksi ohjelmisto voi olla vuorovaikutuksessa toisten kanssa. Vuorovaikutus toimii ilman, että järjestelmän sisäisiä rakenteita täytyy paljastaa. Tämä abstraktio yksinkertaistaa kehitysprosessia, sillä se antaa kehittäjille mahdollisuuden hyödyntää olemassa olevia järjestelmiä uusien ominaisuuksien rakentamisessa. Se johtaa nopeampaan ja tehokkaampaan ohjelmistojen luomiseen.[9]

API:t tarjoavat selkeän ja rakenteellisen tavan viestiä. Ne helpottavat moduulijärjestelmien luomista, joiden avulla kolmannen osapuolen palvelut tai komponentit voidaan integroida luontevasti. API-järjestelmät ovat kehittyneet sisäisistä työkaluista, joita käytettiin yksittäisissä järjestelmissä. Tämä on ollut ratkaisevaa, kun ohjelmistoekosysteemit ovat muuttuneet yhä kytkeytyneemmiksi.

API:t ovat merkittävästi muokanneet ohjelmistokehitystä, erityisesti standardoinnin, moduulirakenteiden ja uudelleenkäytettävyyden osalta. Ne antavat kehittäjille mahdollisuuden välttää jo olemassa olevien ratkaisujen uudelleen keksimisen hyödyntämällä valmiita palveluita ja toimintoja. Tämä ei ainoastaan nopeuta kehitystä, vaan takaa myös johdonmukaisempia ja luotettavampia ohjelmistotuotteita.

Yksi API-järjestelmien suurimmista vaikutuksista on niiden rooli mikropalveluarkkitehtuurissa, jossa suuret sovellukset rakennetaan löyhästi yhteen kytketyistä palveluista, jotka kommunikoivat keskenään API:en avulla. Tämä ohjelmistoarkkitehtuuri mahdollistaa yksittäisten komponenttien helpomman skaalautuvuuden, ylläpidon ja päivittämisen. Se parantaa ohjelmistojärjestelmien yleistä laatua ja joustavuutta.

Lisäksi API-järjestelmät ovat mahdollistaneet alustapohjaisen kehityksen, jossa yritykset avaavat palvelujaan ulkopuolisille kehittäjille edistäen näin innovaatiota ja yhteistyötä. Esimerkiksi API:n avulla kehittäjät voivat integroida maksupalveluita, käyttää pilvitallennusta tai hyödyntää koneoppimisalgoritmeja ilman, että heidän tarvitsee itse kehittää näitä monimutkaisia järjestelmiä alusta alkaen. [9]

REST API (Representational State Transfer Application Programming Interface) on toteutus tyyli, joka mahdollistaa tietojärjestelmien välisen viestinnän verkossa noudattaen HTTP-protokollaa (Hypertext Transfer Protocol). REST API perustuu yksinkertaisiin ja selkeisiin periaatteisiin, joissa resurssit tunnistetaan yksilöllisesti URL-osoitteilla ja käsitellään standardoiduilla HTTP -menetelmillä, kuten GET, POST, PUT ja DELETE. REST API on tunnettu sen keveydestä, skaalautuvuudesta ja helppoudesta käyttää. Sen joustavuus tekee siitä tehokkaan tavan rakentaa hajautettuja ja yhteistyökykyisiä järjestelmiä. [10]

## 2.3 Rajapintastandardien merkitys ja käyttö

APIt ja rajapintaviestintästandardit mahdollistavat molemmat ohjelmistojärjestelmien välisen viestinnän, mutta niillä on eri tarkoitukset. APIt määrittelevät menetelmiä ja protokollia, joiden avulla ohjelmistokomponentti voi hyödyntää toisen toiminnallisuuksia keskittyen vuorovaikutuksen rakenteeseen. Rajapintaviestintästandardit taas luovat yhtenäiset muodot ja säännöt tiedon vaihtoon eri järjestelmien välillä varmistaen, että viestintä tapahtuu yleisesti ymmärrettävällä tavalla ohjelmistototeutuksesta riippumatta.

Rajapintastandardit ovat yleisiä sopimuksia ja toimintatapoja, jotka ohjeistavat miten eri laitteet ja ohjelmistot keskustelevat toistensa kanssa. Standardit mahdollistavat yhtenäisen tavan jakaa ja hyödyntää monenlaisia eri tietolähteitä, riippumatta käytetystä teknologiasta tai alustasta. Ne myös helpottavat kehittäjien työtä, sillä standardien avulla jokaiselle ohjelmalle ei tarvitse kirjoittaa niiden omaa rajapintakirjastoa. Tämä vapauttaa kehittäjille enemmän aikaa keskittyä toiminnallisuuksien ja käyttäjäkokemukseen parantaen tuotteen laatua.

Rajapintastandardeja on kehitetty jokaiselle erityyppiselle rajapinnalle. Standardit voivat määritellä ja ohjata miten laitteistot, ohjelmistot, kommunikointi tai dataformaatit toimivat.

Laitteistoille määritellyt standardit mahdollistavat fyysisten laitteiden saumattoman yhteensopivuuden. USB (universal serial bus) [11] ja HDMI (High-Definition Multimedia Interface) [12] ovat esimerkkejä laiteliitännöistä. HDMI on liitäntä kuvan ja äänen siirtämiseen. USB on oheislaitteiden liittämiseen suunniteltu liitäntä. Laiteliitännät ovat helpottaneet kuluttajien elämää tekemällä eri laitteiden liittämisen helpommaksi ja saumattomammaksi. Myös kehittäjien on helpompi keskittyä tuotekehitykseen, kun jokaista osaa ei tarvitse rakentaa alusta asti vaan voidaan käyttää valmiita ja toimivia ratkaisuja.

Dataformaattien standardit määrittelevät tallennetun datan rakenteen ja salauksen. Näin voidaan varmistua tiedon oikeellisuudesta ja tarkkuudesta. Esimerkiksi CSV (Comma-Separated Values ja JSON (JavaScript Object Notation). Näiden standardien avulla varmistutaan siitä, että data tulkitaan oikein riippumatta alustasta tai ohjelmointikielestä. JSON on noussut suosioon sen keveyden ja luettavuuden ansiosta. Monet API sovellukset välittävät datan JSON-formaatissa.[13] CSV yksinkertainen tekstiformaatti, jossa jokainen kenttä on eroteltu pilkulla. Se on yleisesti käytössä tietokantojen datan tuonnissa ja viennissä.

Kommunikaatiostandardit määrittelevät säännöt ja käytännöt tiedonsiirrolle verkkojen kautta. TCP/IP (Transmission Control Protocol/Internet Protocol) ja HTTP ovat esimerkkejä tiedonsiirto standardeista. TCP/IP protokolla koostuu kahdesta osasta; TCP ja IP. Yhdessä ne kertovat säännöt, miten datapaketit kuljetetaan niiden haluttuun kohteeseen. IP osuus sisältää lähteen ja kohteen osoitteet ja myös optimaalisen reitin päästä kohteeseen. TCP pitää huolen siitä, että paketit kulkevat oikeaan osoitteeseen ja lähettävät vahvistukset, kun paketti on lähtenyt matkaan ja saapunut kohteeseen. TCP määrittelee kuinka paljon tietoa järjestelmä lähettää kerralla liikkeelle ja näin ylläpitää järjestystä. HTTP taas käsittelee yksittäisen laitteen ja palvelimen välisiä pyyntöjä. HTTP-protokolla yksinkertaistettuna toimittaa nettisivut käyttäjille, niiden pyynnöstä. Se mahdollistaa netin selailun.

# 3 Rajapintastandardointi eri aloilla

Rajapintastandardit kattavat laajan valikoiman protokollia, tiedonmuotoiluja ja teknologioita, jotka tukevat yhteentoimivuutta eri toimialoilla ja eri järjestelmissä. Tässä työssä keskitytään erityisesti viestintärajapintastandardeihin. Ne muodostavat keskeisen osan tietojärjestelmien välistä tiedonsiirtoa ja toiminnallisuutta. Näiden standardien tarkastelu tarjoaa näkökulman siihen, miten ne vaikuttavat ohjelmistokehitysprosesseihin, tukevat tehokkuutta, luotettavuutta sekä järjestelmien kehittämistä monimutkaisissa ympäristöissä.

Tässä tutkielmassa käsiteltävät alat – terveydenhuolto, finanssiala, koulutus ja esineiden internet (IoT) – valittiin niiden erilaisten tarpeiden ja lähestymistapojen vuoksi. Ne valottavat rajapintastandardien monipuolista roolia. Standardointi mahdollistaa yhteentoimivuuden, tehokkuuden ja innovaatioiden edistämisen näillä aloilla, vaikka niiden tavoitteet, ympäristöt ja haasteet vaihtelevat merkittävästi. Lisäksi käsitellään käytettävyyden ja kyberturvallisuuden standardisointia, sillä nämä näkökulmat korostavat standardien merkitystä käyttäjäystävällisten ja turvallisten järjestelmien kehittämisessä.

Monialainen tarkastelu hyödyttää tätä tutkielmaa tarjoamalla laajemman näkökulman standardien vaikutuksista. Eri alojen esittely paljastaa, miten standardit mukautuvat yksilöllisiin toimintaympäristöihin ja niiden erityistarpeisiin. Samalla tämä lähestymistapa korostaa standardoinnin universaalia arvoa. Monipuoliset esimerkit osoittavat standardoinnin monimuotoisuuden, mikä vahvistaa tutkielman johtopäätöksiä.

Seuraavissa kappaleissa esiteltävissä artikkeleissa muotoutuu osittaisia vastauksia tutkimuskysymykseen TK1.

## 3.1 Terveydenhuolto

Health Level Seven (HL7) on kansainvälinen rajapintastandardi, joka määrittelee, miten terveydenhuoltoalan taloudelliset, kliiniset ja hallinnolliset viestit ja tiedot siirtyvät eri järjestelmien välillä. Health Level Seven International on vuonna 1987 perustettu voittoa tavoittelematon yritys, joka vastaa standardien tuottamisesta, ylläpidosta ja kehityksestä. [14]

Ennen HL7-standardien luomista tiedon kulku oli hidasta, monimutkaista ja kallista [15].

Uusien tietojärjestelmien luominen oli haastavaa, jokaisen järjestelmän yksilöllisyyden takia. Jos haluttiin saada kaksi järjestelmää keskustelemaan keskenään, niin jokainen rajapinta piti konfiguroida ja luoda yksitellen. Tämä tarkoitti tiedon lähetystä, tallentamista ja vastaanottamista.

Nykyisin sairaaloissa tietojärjestelmät ovat elektronisia. Nämä järjestelmät saattavat olla kokoelmia eri toimittajien tuotteista. Esimerkiksi laskutukset, kliiniset hoidot ja hallinnolliset järjestelmät ovat kaikki eri toimittajilta, vaikka niitä käyttää vain yksi sairaala. HL7-standardit ovat mahdollistaneet tällaisten yhdistelmien luomisen ja toiminnan varmistamisen. Terveydenhuoltoalan organisaatiot eivät ole lukittuina vain yhteen toimittajaan, vaan voivat kilpailuttaa tarjouksia ja järjestelmiä vapaammin.

HL7-standardeista on julkaistu useita eri versioita, jotka vastaavat terveydenhuollon tietojärjestelmien kehittyviin tarpeisiin. Uusia versioita kehitetään parantamaan tietojen yhteensopivuutta, tehokkuutta ja turvallisuutta sekä ratkaisemaan aiempien versioiden rajoituksia. Näiden versioiden kehitys heijastaa terveydenhuollon muuttuvia vaatimuksia ja teknologian edistysaskelia.

HL7 versiot:

1. HL7 v2
2. HL7 v3
3. HL7 FHIR
4. HL7 CDA

### 3.1.1 HL7 v2

Vuonna 1989 esitelty HL7v2 on tullut laajasti käytetyksi HL7-standardin versioksi, toimien perustana yhteensopivuudelle terveydenhuollon tietotekniikassa. [16]

Perustanaan HL7v2 määrittelee rakenteellisen formaatin viestien välittämiseen eri terveydenhuollon järjestelmien välillä, mahdollistaen saumattoman kommunikaation ja datan integraation eri alustojen välillä. Nämä viestit sisältävät strukturoitua dataa, joka on järjestetty segmentteihin, joissa kukin segmentti edustaa tiettyä dataelementtiä tai siihen liittyvää tietoaineistoa. Segmenttien sisäiset kentät on eroteltu, jotta vastaanottavat järjestelmät voivat jäsentää ja tulkita viestejä. [17]

HL7v2-viestit kattavat laajan valikoiman terveystietoa, kuten potilastiedot, kliiniset havainnot, laboratoriotulokset, lääkitykset ja hallinnolliset tiedot. Ne tukevat erilaisia transaktioita, kuten potilaiden vastaanottoja, laboratoriotilauksia, kliinisiä havaintoja ja kotiuttamisraportteja, mahdollistaen kattavan tiedonvaihdon hoitopolun aikana.

Yksi HL7v2:n vahvuuksista on sen joustavuus ja sopeutumiskyky erilaisiin terveydenhuollon ympäristöihin ja käyttötapauksiin. Standardi tarjoaa mahdollisuuden räätälöidä viestirakenteita ja sisältöä käyttämällä viestiprofiileja ja toteutusoppaita, mikä mahdollistaa organisaatioille viestirakenteiden ja sisällön muokkaamisen vastaamaan erityisiä vaatimuksia tai paikallisia työnkulkuja samalla säilyttäen yhteensopivuuden muiden järjestelmien kanssa.

Vaikka HL7v2:lla on laaja käyttö ja pitkäaikainen historia, siinä on myös rajoituksia, kuten sen riippuvuus tekstipohjaisista formaateista, kuten ASCII:sta tai erotellusta tekstistä, mikä voi aiheuttaa haasteita monimutkaisten tietorakenteiden parsimisessa ja käsittelyssä.[18] Lisäksi HL7v2:lla ei ole sisäänrakennettua tukea moderneille verkkopohjaisille viestintäprotokollille, kuten RESTful API:eille, mikä voi haitata yhteensopivuutta nykyaikaisissa terveydenhuollon ekosysteemeissä. [19]

Silti HL7v2 pysyy perustavanlaatuisena standardina terveystiedon vaihdossa, muodostaen monien terveystietojärjestelmien selkärangan maailmanlaajuisesti. Se toimii rinnakkain uudempien versioiden, kuten V3:n ja Fast Healthcare Interoperability Resources (FHIR), kanssa [20]. Sitä ei ole täysin korvattu, sillä monet terveydenhuoltojärjestelmät luottavat edelleen V2:n yksinkertaisuuteen ja vakiintuneisiin integrointeihin.

Esimerkki HL7v2 sanomasta:

```
MSH|^~\&|HospitalSystem|HospitalA|RecipientSystem|HospitalB|
20241207083000||ADT^A01|123456|P|2.5
EVN|A01|20241207083000
PID|1||12345^^^HospitalA^MRN||Doe^John^A^||19800101|M|||
123 Main St^^Anytown^CA^12345^USA||(123)456-7890|||S||123456789|||ENG
PV1|1|I|W^123^1^HospitalA||||12345^Smith^Jane^A^^Dr.^MD|
67890^Johnson^Mark^B^^Dr.^MD|||MED|||||||||20241207083000
```

HL7v2-sanoma koostuu useista segmenteistä, jotka välittävät tietoa potilaan sisäänkirjauksesta sairaalaan. MSH (Message Header) -segmentti määrittää viestin tyypin, lähettäjän ja vastaanottajan sekä viestin luontiajan. EVN (Event Type) -segmentti sisältää tie-

dot tapahtumatyypistä ja -ajasta. PID (Patient Identification) -segmentti tarjoaa potilaan tunnistamiseen liittyviä tietoja, kuten yhteystiedot. PV1 (Patient Visit) -segmentti kuvaa potilaan hoitoon liittyviä tietoja, kuten hoitopaikan ja sisäänkirjauksen ajankohdan. Yhdessä nämä segmentit tarjoavat kokonaiskuvan tapahtumasta, joka tässä tapauksessa on potilaan sisäänkirjaus sairaalaan.

### 3.1.2 HL7 v3

HL7v2 tunnetaan yksinkertaisuudestaan ja joustavuudestaan, mutta siihen liittyy myös epäjohdonmukaisuuksia ja valinnaisuuksia, jotka voivat vaikeuttaa todellista yhteentoimivuutta. Näiden haasteiden ratkaisemiseksi kehitettiin HL7v3, joka käyttää perustanaan viitetietomallia (RIM, eng. Reference Information Model). RIM määrittelee yleiset käsitteet ja niiden väliset suhteet terveydenhuollon kontekstissa, mikä auttaa standardoimaan tietomallinnusta ja parantaa tietojen ymmärrettävyyttä eri järjestelmissä [21]

RIM toimii jäsentämällä terveydenhuollon tapahtumia toisiinsa liittyvien luokkien avulla. Esimerkiksi potilaan käynti klinikalla, jossa hän saa diagnoosin ja reseptin, esitetään eri luokkien kautta: Act Class mallintaa kliinisen havainnon ja hoidon, kun taas Role Class (patient role) kuvaa lääkärin roolia hoidon antajana ja potilaan roolia vastaanottajana. Entity Class edustaa potilasta, lääkäriä ja lääkettä, ja Participation Class yhdistää nämä entiteetit tekoihin, kuten lääkärin osallistuminen diagnoosin tekemiseen. Näiden luokkien väliset suhteet määrittelevät, miten ne ovat vuorovaikutuksessa. [22]

Esimerkki HL7v3 sanomasta:

```
<ClinicalDocument>
  <recordTarget>
    <patientRole>
      <id extension="123456" root="Hospital"/>
      <patient>
        <name>
          <family>Doe</family>
          <given>John</given>
          <suffix>III</suffix>
        </name>
        <birthTime value="19800101"/>
        <administrativeGenderCode code="M"/>
```

```
      </patient>
    </patientRole>
  </recordTarget>
</ClinicalDocument>
```

Tämä esimerkki sisältää potilaan perustiedot. HL7v3 lähettää sanomat XML (eng. Extensible Markup Language) formaatissa. Jäsennelty XML-rakenne mahdollistaa koneellisesti luettavan yhdenmukaisuuden ja integroinnin eri terveydenhuoltojärjestelmien kanssa. Tässä HL7v3 XML-viestissä RIM-mallia hyödynnetään tiedon asianmukaisen järjestelyn ja luokitelun varmistamiseksi. Jokainen komponentti vastaa RIM-luokkia ja suhteita. Toiminnot (kuten kliiniset asiakirjat) on määritelty. Entiteetit (potilaat) on tunnistettu ja kuvattu. Roolit ja suhteet terveydenhuollon vuorovaikutuksessa on ilmaistu.

HL7v3:ssa käytetään XML -pohjaisia viestien rakenteita, mikä mahdollistaa monimutkaisempien tietorakenteiden ja tiedonkuvauksien kuvaamisen. Standardi pyrkii tukemaan semanttista yhteentoimivuutta, mikä tarkoittaa, että viestien sisältö on yhdenmukaisesti määritelty ja tulkittu eri järjestelmissä. [21]

HL7v3 on merkittävä standardi terveystiedon vaihtoon, mutta sitä ei ole otettu laajasti käyttöön. HL7v3:sta puuttui joustavuus, jota tarvittiin käytännön työssä. Vaikka HL7v3 oli mahdollista mukauttaa, sen laajat spesifikaatiot edellyttivät asiantuntijoiden ymmärrystä RIM:stä ja tuettujen työkalujen käytöstä. Tämä ei vastannut HL7:n alkuperäistä tavoitetta, joka oli suunnitella edullinen ja helposti toteutettava ratkaisu. V3-asiantuntijoiden määrä oli pieni ja onnistuneet toteutukset vaativat merkittäviä resursseja. [23]

HL7 V3 oli tarkoitettu V2:n päivitykseksi, jossa on tarkempia rakenteita ja semantiikkaa. Se ei kuitenkaan täysin korvannut V2:ta vaan V3 toimii usein rinnakkain V2:n kanssa. Vaikka HL7v3:lla onkin tavoitteena parantaa terveystiedon vaihtoa, sen käyttöönotto ja käyttö ovat olleet haastavia. Standardi on monimutkainen, ja sen käyttöönotto vaatii merkittäviä resursseja ja teknistä osaamista.

### 3.1.3 HL7 FHIR

HL7 FHIR on suunniteltu modernisoimaan ja yksinkertaistamaan terveydenhuollon tietojen vaihtoa entisestään, ja se ratkaisee sekä V2:n että V3:n rajoituksia [24]. Vaikka FHIR voi joissain tapauksissa korvata vanhemmat versiot, sitä käytetään usein yhdessä

V2:n ja V3:n kanssa, erityisesti hybridijärjestelmissä, joissa eri komponentit palvelevat eri tarkoituksia. [20]

FHIR:n ytimessä on resurssiorientoitunut ohjelmistoarkkitehtoninen tyyli, jossa terveystieto jaetaan modulaarisiin osiin, joita kutsutaan *resursseiksi*. Nämä resurssit edustavat erillisiä kliinisiä, hallinnollisia ja taloudellisia tietoja, kuten potilaita, lääkkeitä, diagnooseja, toimenpiteitä ja muuta. Jokainen resurssi on standardoitu ja määritelty yhtenäisen tietomallin avulla, mikä helpottaa tiedon ymmärtämistä ja käsittelyä eri terveydenhuollon sovelluksissa ja järjestelmissä. [25]

Yksi FHIR:n keskeisistä vahvuuksista on sen RESTful API -suunnittelu, joka mahdollistaa yksinkertaisen ja tehokkaan viestinnän terveydenhuollon järjestelmien välillä standardien verkkoprotokollien, kuten HTTP:n, avulla. Tämä API-keskeinen lähestymistapa mahdollistaa kehittäjille terveystiedon nopeamman noutamisen, luomisen, päivittämisen ja poistamisen käyttämällä tuttuja verkkoteknologioita, mikä edistää terveydenhuollon sovellusten nopeaa integrointia ja kehittämistä. [25]

Lisäksi FHIR tukee laajaa valikoimaa tietomuotoja, kuten JSON:ia ja XML:ää (eng. Extensible Markup Language), mikä tarjoaa joustavuutta erilaisten käyttötapojen ja mieltymysten huomioimiseksi. [25] Tämä sopeutumiskyky ulottuu eri terveydenhuollon aloille, kuten kliiniseen hoitoon, tutkimukseen, kansanterveyteen ja terveydenhuollon hallintoon, mikä tekee FHIR:sta monipuolisen standardin terveystiedon vaihtoon eri konteksteissa.

Tämä versio edustaa merkittävää edistysaskelta terveydenhuollon yhteensopivuudessa, tarjoten modernin, joustavan ja standardoidun lähestymistavan terveystiedon vaihtoon. Se mahdollistaa sidosryhmien potilaiden hoitoa, virtaviivaistaa työnkulkuja ja edistää innovaatioita terveydenhuollon alalla.

### 3.1.4 HL7 CDA

HL7 CDA (Clinical Document Architecture) käytetään usein yhdessä HL7 V3:n ja FHIR:n kanssa asiakirjapohjaiseen viestintään, sen sijaan että se korvaisi muita versioita. Se toimii täydentävänä standardina, joka tukee terveydenhuollon järjestelmissä rakenteellista tiedonvaihtoa. [26]

Tämä täydennys mahdollistaa kliinisten dokumenttien vaihdon eri järjestelmien välillä. CDA-dokumentit perustuvat XML-formaatissa määriteltyyn rakenteeseen, joka mahdollistaa niiden käsittelyn ja tulkinnan tietokoneohjelmien avulla. Jokainen CDA-dokumentti

koostuu useista osista, kuten otsikko, potilastiedot, kliiniset havainnot ja hoitosuunnitelma. Osat on järjestetty hierarkkiseen rakenteeseen. [27]

Yksi CDA:n tärkeimmistä ominaisuuksista on sen kyky tarjota ymmärrettävää ja yhtenäistä kliinistä tietoa eri järjestelmien välillä. Dokumentit sisältävät tietoa sekä ihmisen että tietokoneen luettavassa muodossa, mikä helpottaa niiden käyttöä monissa terveydenhuollon sovelluksissa ja käyttötapauksissa.

Vaikka CDA on osoittautunut hyödylliseksi monissa terveydenhuollon yhteyksissä, sen käyttöönotto ja käyttö ovat saattaneet olla haastavia joillekin organisaatioille. Dokumenttien laaja-alainen rakenne ja monimutkaisuus voivat aiheuttaa vaikeuksia niiden luomisessa ja käsittelyssä ilman asianmukaista koulutusta ja tietoteknistä osaamista.

HL7 CDA on tärkeä standardi terveydenhuollon tiedonvaihdossa, joka tarjoaa strukturoidun ja yhdenmukaisen lähestymistavan kliinisen tiedon jakamiseen ja tallentamiseen. Sen avulla voidaan parantaa tiedon saavutettavuutta, vaihdettavuutta ja ymmärrettävyyttä eri terveydenhuollon sidosryhmien kesken. Ajantasainen tieto tukee potilashoidon laadun parantamista ja tukee päätöksentekoa .

## 3.2 Finanssiala

Rahoituslaitokset ovat jatkuvasti kanssakäymisissä toistensa sekä asiakkaiden kanssa. Suurin osa tästä kommunikaatiosta tapahtuu tietokoneiden välillä. Kommunikoitava data vaatii molemmilta osapuolilta ymmärrystä viestin sisällöstä. Jotta ihmisten ei tarvitse tulkita jokaista riviä dataa ja kääntää viestejä omaan järjestelmäänsä, nämä laitokset voivat luoda viestimäärityksiä. Määritykset ovat sopimuksia siitä, kuinka tieto voidaan järjestää syötteen (syntaksin) ja merkityksen (semantiikan) perusteella.

Finanssialan rajapintastandardit edistävät taloudellista osallisuutta mahdollistamalla saumattoman yhteentoimivuuden erilaisten rahoitusjärjestelmien välillä. Nämä standardit vähentävät rahoitukseen liittyviä esteitä. Korkeat kustannukset, rajallinen pääsy pankkien palveluihin ja yleinen tietämättömyys ovat suurimpia esteitä kehittyvissä maissa.[28] Standardit alentavat kansainvälisen kaupankäynnin ja transaktioiden kuluja sekä edistävät kilpailua eri toimittajien välillä. Ne helpottavat reaaliaikaisia liiketoimia ja yhteensopivuutta. [29]

SWIFT ja ISO 20022 valittiin niiden merkittävän roolin vuoksi finanssialan viestinnässä:

SWIFT on pitkään ollut vakiintunut standardi kansainvälisissä rahansiirroissa, kun taas ISO 20022 edustaa nousevaa globaalia standardia finanssidatan vaihtoon.

### 3.2.1 SWIFT

SWIFT on pitkään käytössä ollut viestintästandardi, jota käytetään pääasiassa kansainvälisiin pankkisiirtoihin. Vuonna 1973 perustettu SWIFT tarjoaa verkoston, joka mahdollistaa rahoituslaitosten maailmanlaajuisen viestinnän turvallisessa, standardoidussa ja luotettavassa ympäristössä. Formaatti on tämän verkoston selkäranka, ja sen avulla voidaan suorittaa laaja valikoima pankkitoimintoja, kuten maksuja, arvopapereita, varainhallintatoimintoja ja kaupan rahoitusta. [30]

SWIFT-viestit ovat tarkasti määriteltyjä ja jäsenneltyjä, jotta viestintä laitosten välillä olisi yksiselitteistä ja virheetöntä. Jokainen viestityyppi on nimetty kolminumeroisella koodilla, ja eri transaktioille on omat tarkat formaattinsa. Esimerkiksi MT103 käytetään yksittäisiin asiakasmaksuihin ja MT202 rahoituslaitosten välisiin siirtoihin. Tämä standardoitu lähestymistapa vähentää virheriskiä ja tehostaa kansainvälisten maksujen käsittelyä ja täsmäytystä, mikä tekee siitä keskeisen osan globaalia rahoitusjärjestelmää. [30]

Rahoitusala jatkaa kehittymistään, ja SWIFT-verkosto siirtyy kohti ISO 20022 -standardin käyttöönottoa. Se tuo mukanaan joustavamman rakenteen ja tarjoaa paremmat tietomuodot XML:n avulla. Tämä muutos edistää parempaa yhteensopivuutta, tehokkuutta ja tietojen selkeyttä rahoitusviestinnässä. Siirtymävaiheessa SWIFT säilyy laajassa käytössä ja jatkaa keskeisenä osana kansainvälistä rahoitusviestintää vielä pitkään. Vanhoja standardeja, kuten SWIFT MT, tuetaan vuoteen 2025 asti. [31]

Kun Venäjä aloitti hyökkäyksensä Ukrainaan helmikuussa 2022, monet länsimaat reagoivat asettamalla pakotteita Venäjän taloutta vastaan. Yksi pakotteista oli useiden venäläisten pankkien sulkeminen pois SWIFT-järjestelmästä. Tämä poissulkeminen SWIFT-verkosta merkitsi sitä, että nämä pankit eivät voineet suorittaa tai vastaanottaa kansainvälisiä maksuja SWIFT-järjestelmän kautta. Se vaikeutti merkittävästi niiden kykyä harjoittaa kansainvälistä liiketoimintaa ja heikensi maan taloudellisia yhteyksiä ulkomaailmaan. SWIFT-järjestelmän kautta tapahtuva sulkeminen voidaan toteuttaa estämällä pankkien pääsy viestintäverkkoon, jolloin ne menettävät kykynsä lähettää ja vastaanottaa standardoituja maksusanomia. [32]

### 3.2.2 ISO 20022

ISO 20022 on kansainvälinen standardi, jonka on kehittänyt International Organization for Standardization (ISO). Se tarjoaa yhteisen alustan finanssialan viestinnälle ja otettiin käyttöön vuonna 2004 korvaamaan vanhempia viestintästandardeja, kuten SWIFT. XML-pohjaisen formaattinsa ansiosta se mahdollistaa yksityiskohtaisen ja rakenteellisen tiedonvaihdon. Se parantaa tiedon laatua ja prosessien tehokkuutta.

Standardin laaja sovellettavuus kattaa erilaisia rahoituspalveluita, kuten maksut, arvopaperit, kaupan rahoitus ja valuutanvaihto. Esimerkiksi maksujen osalta se mahdollistaa reaaliaikaiset maksut ja rajat ylittävät transaktiot. Se tehostaa kansainvälisiä maksuprosesseja. Arvopaperikaupoissa ISO 20022 parantaa selvitystä ja täsmäytystä. Kaupan rahoituksessa se helpottaa rahoitusinstrumenttien, kuten remburssien, käsittelyä.

Uuden standardin käyttöönotto tuo mukanaan merkittäviä etuja, kuten parantuneen tietojen laadun ja yksityiskohtaisuuden. Se vähentää virheitä ja nopeuttaa käsittelyaikoja. Standardin joustavuus ja laaja sovellettavuus tekevät siitä ihanteellisen ratkaisun monimutkaisten ja nopeasti muuttuvien rahoitusmarkkinoiden tarpeisiin. Vaikka siirtyminen ISO 20022 vaatii merkittäviä investointeja ja järjestelmien mukauttamista, pitkällä aikavälillä se johtaa huomattaviin kustannussäästöihin ja prosessien tehostumiseen. [31]

### 3.2.3 2010-luvun innovaatiot

#### FDX API

FDX API (Financial Data Exchange API) on nykyaikainen rajapintastandardi, joka mahdollistaa turvallisen ja johdonmukaisen pääsyn taloudellisiin tietoihin eri rahoituslaitosten ja palveluntarjoajien välillä. Se perustuu OAuth 2.0 -valtuutuskoodivirtaan, joka hyödyntää Rich Authorization Requests (RAR) - ja Pushed Authorization Requests (PAR) -ominaisuuksia tarkkojen suostumuskokonaisuuksien hallintaan. Tämä lähestymistapa takaa vahvan käyttäjien tunnistautumisen ja valtuutuksen, mikä minimoi tietoturvariskit ja parantaa tiedonhallintaa käyttäjän suostumuksen pohjalta. [33]

FDX API on suunniteltu vastaamaan käyttäjien kasvaviin tarpeisiin hallita ja jakaa taloudellisia tietojaan turvallisesti. Standardi tarjoaa yhdenmukaisen teknisen kehyksen, joka tukee järjestelmien ja palveluntarjoajien välistä yhteentoimivuutta ja vähentää integroinnin monimutkaisuutta [33]. Esimerkiksi FDX mahdollistaa käyttäjän suostumuksen perus-

teella tapahtumien, tiliotetietojen ja tilikohtaisen datan turvallisen jakamisen kolmansille osapuolille.

FDX API pyrkii edistämään turvallisempaa ja läpinäkyvämpää tiedonhallintaa. Käyttäjien suostumus varmistaa, että data on käytettävissä vain käyttäjän valtuuttamille osapuolille.

#### Mojaloop

Mojaloop on avoimen lähdekoodin ohjelmisto, joka on suunniteltu parantamaan maksujärjestelmien yhteentoimivuutta, erityisesti kehittyvissä maissa. Toisin kuin SWIFT tai ISO 20022, jotka keskittyvät pääasiassa pankkien ja muiden rahoituslaitosten väliseen standardoituun viestintään, Mojaloop tarjoaa ratkaisun niille, joilla ei ole pääsyä perinteisiin pankkipalveluihin. Näitä "pankittomia"ihmisiä ovat esimerkiksi köyhissä ja syrjäisissä yhteisöissä asuvat, joilla ei ole pääsyä pankkipalveluihin korkean kustannuksen, maantieteellisten esteiden tai henkilöllisyystodistusten puuttumisen vuoksi. Mojaloop pyrkii poistamaan nämä esteet tarjoamalla avoimen ja helposti käyttöönotettavan alustan. [34]

Mojaloopin perusidea on yhdistää useita digitaalisia finanssipalveluiden tarjoajia kilpailukykyiseksi ja yhteentoimivaksi verkostoksi, jonka kautta rahansiirrot ja maksut voidaan toteuttaa alhaisilla tai olemattomilla maksuilla. Ohjelmisto hyödyntää mobiilipankkitekniikoita ja digitaalisia lompakoita, joita tavalliset ihmiset voivat käyttää ilman, että heillä on perinteistä pankkitiliä. Tämä edistää taloudellista osallisuutta tarjoamalla matalan kynnyksen pääsyn maksujärjestelmiin niin yksityishenkilöille kuin pienyrityksille. Mojaloop tukee myös kehittyvien maiden keskuspankkeja ja maksujärjestelmiä mahdollistamalla avoimen infrastruktuurin, joka ehkäisee monopolien syntymistä ja edistää kilpailua markkinoilla. [34]

### 3.2.4 Standardien tavoitteet

Rahoitusalan viestintästandardit, kuten ISO 20022, pyrkivät mullistamaan globaalin finanssiteollisuuden tarjoamalla yhteisen kielen parantamaan yhteentoimivuutta, tehokkuutta ja turvallisuutta eri rahoitusjärjestelmien ja -laitosten välillä. Ensisijainen tavoite on helpottaa saumatonta ja tarkkaa tiedonvaihtoa vähentäen kansainvälisiin transaktioihin liittyviä monimutkaisuuksia ja riskejä.

Keskeinen tavoitte on myös parantaa tiedon laatua. Käyttämällä rakenteellisia formaatteja rahoitusviestit voivat sisältää yksityiskohtaisempaa ja tarkempaa tietoa. Tämä johtaa tehokkaampaan käsittelyyn, vähentää virheitä ja parantaa täsmäytysprosesseja. Lisäksi paremmat tietomuodot tukevat sääntelyn noudattamista varmistamalla, että kaikki tarvittavat tiedot sisällytetään ja välitetään oikein.

Standardien tavoitteena on myös turvata rahoitusviestintä tulevaisuudessa olemalla mukautuvia ja joustavia vastaamaan rahoitusalan kehittyviin tarpeisiin. Kun uusia rahoitustuotteita, palveluita ja sääntelyvaatimuksia syntyy, ISO 20022 kaltaiset standardit on suunniteltu mukautumaan näihin muutoksiin varmistaen pitkän aikavälin merkityksellisyyden ja käyttökelpoisuuden. Tämä mukautuvuus on ratkaisevan tärkeää rahoitusviestinnän eheyden ja tehokkuuden ylläpitämiseksi globaalissa markkinassa.

Merkittävä tavoite on edistää globaalia harmonisointia. Yleisstandardin omaksumalla rahoituslaitokset maailmanlaajuisesti voivat saavuttaa suuremman yhdenmukaisuuden toiminnoissaan, joka helpottaa sujuvampia rajat ylittäviä transaktioita ja edistää kansainvälistä yhteistyötä. Tämä globaali harmonisointi on olennaista yhteisemmän ja kestävämmän rahoitusekosysteemin edistämiseksi.

### 3.2.5 Tulevaisuuden mahdolliset muutokset

Rahoitusala kehittyy jatkuvasti. Useat uudet teknologiat voivat korvata tai merkittävästi parantaa nykyisiä standardeja kuten blockchain- ja hajautetun tilikirjan teknologian (DLT eng. distributed ledger technology) kasvava käyttö. Lohkoketju tarjoaa hajautetun ja muuttumattoman tilikirjan rahoitustapahtumille, mikä voi parantaa turvallisuutta, läpinäkyvyyttä ja tehokkuutta. Esimerkiksi Stellar, joka on hajautettu avoimeen lähdekoodiin perustuva maksujärjestelmä, hyödyntää lohkoketjua rajat ylittävien maksujen helpottamiseen, vähentäen transaktioaikoja ja kustannuksia perinteisiin SWIFT-siirtoihin verrattuna. [35] [36]

Toinen lupaava kehityspolku on avoimen pankkitoiminnan API-rajapintojen nousu. Avoin pankkitoiminta mahdollistaa kolmansien osapuolien kehittäjille sovellusten ja palveluiden rakentamisen rahoituslaitosten ympärille käyttäen avoimia API-rajapintoja turvalliseen tiedonjakoon. Tämä voi johtaa innovatiivisempiin rahoitustuotteisiin ja -palveluihin sekä sujuvampaan integraatioon eri rahoitusjärjestelmien välillä. Euroopassa PSD2-direktiivi edistää avoimen pankkitoiminnan käyttöönottoa lisäten kilpailua ja parantaen kuluttajien valinnanmahdollisuuksia. [37]

Tekoäly (AI eng. Artificial intelligence) ja koneoppiminen (ML eng. Machine learning) ovat myös keskeisessä asemassa tulevaisuuden rahoitusviestinnässä. AI ja ML voivat parantaa petostentorjuntaa, riskienhallintaa ja tapahtumien seurantaa analysoimalla valtavia määriä tietoa reaaliajassa. Nämä teknologiat voivat auttaa automatisoimaan ja optimoimaan monia prosesseja, joita perinteiset viestintästandardit tällä hetkellä käsittelevät. AI ja ML tarjoaa tarkempia ja ajantasaisempia vastauksia rahoitustapahtumiin. [38]

Kvanttitietokoneiden kehitys voisi mullistaa rahoitustapahtumat tarjoamalla ennennäkemätöntä laskentatehoa salaukseen ja tietojen analysointiin. Kvanttitietokoneet voivat murtaa nykyiset salausmenetelmät, joka vaatii kvanttivarmojen kryptografisten algoritmien kehittämistä. Tämä muutos johtaisi uuteen sukupolveen turvallisia rahoitusviestintäprotokollia, jotka pystyvät vastaamaan kvanttiajan tietojenkäsittelyn ja turvallisuuden haasteisiin. [39]

## 3.3 Opetus

Opetussektorilla xAPI, SCORM ja LTI ovat standardeja, jotka tukevat digitaalisen oppimisen tarpeita. SCORM varmistaa oppimateriaalien yhteensopivuuden eri alustojen välillä, kun taas LTI mahdollistaa oppimistyökalujen integraation saumattomasti oppimisympäristöihin. xAPI puolestaan laajentaa oppimiskokemusten analysointia kattamaan monimuotoisia tapahtumia eri ympäristöissä. Näiden standardien valinta perustuu niiden vakiintuneeseen käyttöön, monipuolisuuteen ja kykyyn tukea nykyaikaisia, yhteentoimivia oppimisratkaisuja.

### 3.3.1 Merkitys

Koulujen ja oppilaitosten digitaalisten työkalujen ja järjestelmien käyttö kasvaa jatkuvasti [40] . Näiden teknologioiden välinen tehokas viestintä ja yhteensopivuus ovat entistä tärkeämpiä. Rajapintastandardit varmistavat sen, että opettajat ja opiskelijat voivat käyttää oppimateriaaleja, seurata edistymistä ja integroida ulkopuolisia resursseja ilman häiriöitä.

LMS-järjestelmät (Learning Management Systems) ovat oppimisympäristöjä, joissa hallitaan oppimateriaaleja, seurataan opiskelijoiden edistymistä ja tuetaan opetuksen eri osa-alueita. LMS-järjestelmien avulla opettajat ja oppilaitokset voivat tehokkaasti suunnitella, jakaa ja seurata oppimateriaalien käyttöä sekä hallita oppimiskokemusta kokonaisuutena.

### 3.3.2 Tavoitteet

Rajapintastandardit tarjoavat selkeät säännöt ja käytännöt, joiden avulla eri järjestelmät voivat kommunikoida keskenään tehokkaasti. Tämä on erityisen tärkeää koulutuksessa, jossa monenlaiset oppimisalustat, sovellukset ja digitaaliset työkalut ovat käytössä. Standardit, kuten xAPI, SCORM ja LTI, varmistavat tietojen siirtymisen saumattomasti järjestelmien välillä. [41]

Nämä standardit mahdollistavat myös ulkoisten resurssien integroinnin ja varmistavat, että kaikki opetusmateriaalit ovat helposti käytettävissä ja jaettavissa eri oppimisympäristöissä. Standardit helpottavat myös opettajien ja opiskelijoiden välistä yhteistyötä, jonka ajatellaan parantavan oppimisen laatua ja saavutettavuutta. [41]

### 3.3.3 Hyödyt

Rajapintastandardit mahdollistavat esimerkiksi opiskelijoiden suoritusten automaattisen seurannan ja raportoinnin eri järjestelmien välillä. Samalla opettajat ja kouluttajat voivat hyödyntää laajempaa valikoimaa työkaluja ja sisältöjä ilman huolta siitä, että ne eivät toimisi yhteen olemassa olevien järjestelmien kanssa.

Standardointi tarjoaa myös vakautta ja kestävyyttä koulutusjärjestelmille myös tulevaisuudessa. Oppimateriaalit ja sisällöt, jotka noudattavat kansainvälisesti hyväksyttyjä standardeja, säilyvät käyttökelpoisina teknologiakehityksen mukana ja vähentävät tarvetta toistuviin järjestelmäpäivityksiin. Tämä ei pelkästään säästä resursseja, vaan auttaa varmistamaan, että opetus ja oppiminen jatkuvat saumattomasti, vaikka teknologiat kehittyisivät ja muuttuisivat ajan myötä. Lisäksi standardit tarjoavat paremmat mahdollisuudet oppimisanalytiikan käyttöön, mikä tukee yksilöllisempää ja dataan pohjautuvaa oppimisen kehittämistä.

Merkittävä hyöty on innovatiivisuuden mahdollistaminen koulutusteknologiassa. Standardit luovat avoimet rajapinnat, joiden kautta uudet ja innovatiiviset oppimistyökalut voidaan integroida olemassa oleviin järjestelmiin ilman laajoja muutosprojekteja. Tämä kannustaa kehittäjiä luomaan uusia ratkaisuja, jotka rikastuttavat oppimiskokemusta ja tuo koulutusorganisaatioille enemmän valinnanvaraa sekä joustavuutta teknologian käytössä. Yhdessä nämä rajapintastandardien edut tukevat koulutusjärjestelmien tehokkuutta, jatkuvuutta ja innovatiivisuutta.

### 3.3.4 Haasteet

Rajapintastandardien käyttöönotto e-oppimisjärjestelmissä kohtaa useita haasteita, jotka voivat vaikuttaa järjestelmien tehokkuudessa ja saavutettavuudessa. Haasteihin lukeutuu esimerkiksi:

- Legacy-järjestelmät
- Laaja käyttöönoton puute
- Resurssien puute

Yhteentoimivuusongelmat johtuu usein standardien erilaisista tulkinnoista kehittäjien välillä, mikä voi vaikeuttaa yhteistä ymmärrystä ja heikentää järjestelmien yhteentoimivuutta. Erityisesti tilanteissa, joissa koulutusorganisaatiot käyttävät vanhoja tai räätälöityjä järjestelmiä, jotka eivät noudata uusimpia standardeja, yhteensopivuusongelmat voivat haitata tietojen sujuvaa siirtymistä ja järjestelmien integraatiota [42]. Lisäksi pienemmillä oppilaitoksilla voi olla haasteita investoida tarvittaviin päivityksiin, joka voi edelleen hankaloittaa tilannetta.

Rajapintastandardien kehittäminen ja ylläpito vaatii usein laajaa yhteistyötä eri toimijoiden kesken [41]. Yhteistyöhaasteet voivat hidastaa uusien standardien käyttöönottoa ja aiheuttaa viiveitä teknologian kehityksessä. Kun standardit kehittyvät, on tärkeää varmistaa, että ne pysyvät ajan tasalla nopeasti muuttuvan teknologian kanssa. Tämän lisäksi koulutuksen globaalissa kentässä voi esiintyä alueellisia eroja standardien soveltamisessa ja hyväksymisessä, joka voi edelleen vaikeuttaa yhteensopivuuden saavuttamista eri maiden ja alueiden välillä.

Haasteena on myös kustannukset ja resurssit [43]. Vaikka standardit tarjoavat pitkäaikaista hyötyä, niiden käyttöönotto ja ylläpito vaativat sekä alkuinvestointeja että henkilöstön kouluttamista. Tämä voi olla taloudellisesti haastavaa. Mikäli standardit vaativat suuria päivitysprojekteja joissa useita järjestelmiä on integroitu yhtenäiseksi kokonaisuudeksi, yhteistyö- ja resurssihaasteet voivat olla jopa esteenä.

### 3.3.5 SCORM

SCORM (Sharable Content Object Reference Model) on yksi yleisimmin käytetyistä e-oppimisen kehysmalleista. Se määrittelee koulutussisällön luomisen, paketoinnin ja jake-

lun. SCORM varmistaa, että oppimateriaalit voidaan toimittaa tehokkaasti ja yhdenmukaisesti eri alustoilla. SCORM jakaa sisällön "Learning Objecteiksi", jotka voivat siirtyä saumattomasti järjestelmien välillä, mikä tekee siitä keskeisen työkalun organisaatioille, jotka haluavat tehostaa sisällönhallintaansa. [44]

Vaikka SCORMin vahvuus on sen kyky standardisoida sisällön jakelu, sen rajallisuus näkyy oppijan sitoutumisen seurannassa. SCORM keskittyy lähinnä siihen, mitä sisältöä on jaettu, mutta ei syvällisesti analysoi, miten oppijat ovat vuorovaikutuksessa sisällön kanssa muissa kuin LMS-ympäristöissä. [44]

### 3.3.6 Learning Tools Interoperability

LTI on IMS Global Learning Consortiumin kehittämä standardi, joka mahdollistaa kolmannen osapuolen oppimisvälineiden ja sovellusten saumattoman integroinnin oppimisenhallintajärjestelmiin. Se tarjoaa oppilaitoksille mahdollisuuden sisällyttää ulkoisia palveluja, kuten virtuaalisia laboratorioita, arviointeja ja simulaatioita, suoraan LMS (Learning Management System) käyttöliittymään, laajentaen oppimiskokemusta ilman, että käyttäjän tarvitsee siirtyä alustalta toiselle. LTI parantaa käyttökokemusta sallimalla yhden kirjautumisen periaatteen, jolloin käyttäjät pääsevät kaikkiin ulkoisiin työkaluihin ilman erillistä kirjautumista. [45]

LTI:n etuna on sen modulaarinen lähestymistapa digitaaliseen oppimiseen. Oppilaitokset voivat lisätä uusia työkaluja tarpeen mukaan varmistaen, että niiden oppimisympäristöt voivat kehittyä ja laajentua ilman suuria järjestelmäuudistuksia. [45]

### 3.3.7 xAPI

Experience API (xAPI), tunnetaan myös nimellä Tin Can API, on kehittynyt e-oppimisen standardi, joka mahdollistaa oppijan toiminnan seurannan eri ympäristöissä. Se voi tallentaa tietoa niin verkko- kuin offline-oppimisesta, simulaatioista ja muista todellisista kokemuksista, eikä rajoitu pelkästään perinteisiin oppimisenhallintajärjestelmiin (LMS). xAPI keskeinen piirre on Learning Record Store (LRS), joka tallentaa ja analysoi kaikki oppimisen tapahtumat eri järjestelmistä. Tämä mahdollistaa yksityiskohtaisemman tiedonkeruun ja oppimisprosessin ymmärtämisen, mikä auttaa oppilaitoksia tekemään datalähtöisiä päätöksiä ja mukauttamaan opetustaan reaaliajassa. [46]

xAPI:n joustavuus on sen suurin vahvuus. Se tarjoaa kokonaisvaltaisen kuvan oppimisesta, joka ei rajoitu pelkästään digitaalisille alustoille. Tämä mahdollistaa koulutusorganisaatioille paremman tavan arvioida oppimisen tuloksia ja hyödyntää analytiikkaa pedagogisten päätösten tukena. [46]

## 3.4 Internet of Things

Internet of Things (IoT) eli esineiden internet viittaa verkkoon kytkettyihin fyysisiin laitteisiin, jotka pystyvät keräämään ja vaihtamaan tietoa toistensa kanssa. Nämä laitteet voivat olla mitä tahansa kodinkoneista teollisuuslaitteisiin ja jopa ajoneuvoihin. IoT-teknologia mahdollistaa automatisoidut toiminnot ja datan analysoinnin reaaliajassa.

### 3.4.1 Standardoinnin tärkeys

Internet of Things (IoT) viittaa laitteiden ja järjestelmien verkostoon, jotka ovat yhteydessä toisiinsa ja vaihtavat dataa suorittaakseen erilaisia tehtäviä. Se kattaa monenlaisia laitteita, kuten antureita, älylaitteita ja teollisuuden järjestelmiä, jotka toimivat yhdessä luoden älykkäitä ja automatisoituja ympäristöjä. Ilman liitäntästandardeja IoT-ekosysteemi sirpaloituisi, mikä heikentäisi laitteiden toimivuutta ja vähentäisi niiden vaikuttavuutta merkittävästi. [47]

Rajapintastandardit parantavat IoT:n skaalautuvuutta ja turvallisuutta. Kun IoT-järjestelmät laajenevat ja laitteiden integrointi eri verkkoihin ja alustoihin kasvaa, uusien laitteiden liittäminen olemassa oleviin verkkoihin tulee yhä haastavammaksi. Tämä johtuu teknisistä haasteista, kuten erilaisten tietomuotojen ja tiedonsiirtomekanismien yhteensopivuusvaatimuksista, sekä tarvittavista rajapinnoista, jotka mahdollistavat erillisten järjestelmien yhdistämisen. Standardoidut rajapinnat helpottavat tätä prosessia, jolloin kehittäjät voivat lisätä uusia laitteita ilman, että koko järjestelmä tarvitsee muokata. Standardit varmistavat myös, että laitteiden välinen tiedonsiirto on turvallista, estäen luvattoman pääsyn ja varmistaen IoT-ekosysteemin eheyden. Täten rajapintastandardit ovat keskeisiä IoT-verkkojen onnistuneessa toteutuksessa ja kasvussa eri teollisuudenaloilla. [47]

IoT hyödyntää laajan valikoiman standardeja ja protokollia, jotka mahdollistavat laitteiden, järjestelmien ja palveluiden välisen vuorovaikutuksen. Tässä työssä tarkasteltavat MQTT ja CoAP ovat vain pieni osa IoT-kokonaisuudesta. Ne valittiin keskeisiksi tutki-

muskohteiksi niiden ominaisuuksien vuoksi. Ne edustavat erilaisia lähestymistapoja resurssitehokkuuteen ja yhteentoimivuuteen, ovat laajasti käytössä eri aloilla ja tarjoavat arvokkaita näkökulmia standardien vaikutuksesta IoT-ekosysteemin kehitykseen.

### 3.4.2 Haasteet

Vaikka rajapintastandardit ovat IoT:n menestyksen kulmakivi, niiden käyttöönotto ja hallinta tuovat mukanaan useita haasteita. Yksi merkittävimmistä haasteista on standardien moninaisuus. Koska eri valmistajat ja toimialat kehittävät omia standardejaan, voi syntyä yhteensopimattomuutta laitteiden ja järjestelmien välillä. Tämä voi johtaa siihen, että IoT-laitteet eivät aina toimi saumattomasti yhdessä, mikä heikentää käyttäjäkokemusta ja estää laajempaa käyttöönottoa.

Toinen haaste liittyy tietoturvaan ja yksityisyyteen. IoT-verkot keräävät ja käsittelevät valtavia määriä henkilökohtaisia ja arkaluonteisia tietoja, mikä altistaa ne kyberuhkille. Liitäntästandardit voivat tarjota perustan turvalliselle tiedonsiirrolle, mutta niiden käyttöönotto ei automaattisesti takaa riittävää tietoturvaa. Lisäksi globaalissa IoT-ekosysteemissä on vaikea luoda ja ylläpitää yhtenäisiä turvallisuusstandardeja, kun otetaan huomioon eri maiden sääntelyerot ja tekniset vaatimukset. [47]

Haasteet eivät rajoitu vain tietoturvaan ja yhteensopivuuteen. IoT-verkkojen jatkuva kehittyminen vaatii myös standardien joustavuutta ja kykyä mukautua uusiin teknologioihin. Standardit, jotka eivät pysy kehityksen mukana, voivat rajoittaa innovaatioita ja hidastaa IoT-järjestelmien edistymistä, mikä luo lisää haasteita ekosysteemin kasvulle ja ylläpidolle. [47]

### 3.4.3 MQTT

MQTT (Message Queuing Telemetry Transport) on kevyt viestinvälitysprotokolla, joka on suunniteltu laitteille, joiden resurssit, kuten kaistanleveys ja virrankulutus, ovat rajallisia. MQTT vähentää verkkoliikennettä ja prosessointitehoa lähettämällä dataa vain laitteille, jotka sitä tarvitsevat [48]. Resurssien tehokas käyttö on tärkeää IoT-ympäristöissä, joissa kaistanleveys ja akun kesto ovat usein rajallisia. MQTT on käytössä esimerkiksi älykodeissa ja teollisuuden laitteiden valvonnassa.

MQTT tunnetaan skaalautuvuudestaan ja joustavuudestaan. Sen avulla käyttäjät voi-

vat säätää viestien toimitusvarmuutta sovelluksen tarpeiden mukaan. Koska MQTT on suunniteltu kevyeksi protokollaksi, siihen ei kuitenkaan sisälly sisäänrakennettua salausta tai todennusta, mikä voi aiheuttaa tietoturvariskejä IoT-järjestelmissä [49]. Näiden riskien vähentämiseksi MQTT:tä käytetään usein yhdessä lisäturvakerrosten kanssa. Näillä suojauksilla MQTT toimii turvallisena ja tehokkaana protokollana IoT-sovelluksissa, mahdollistaen laitteiden reaaliaikaisen viestinnän kaistanleveyttä ja virtaa säästäen.

### 3.4.4 CoAP

CoAP (Constrained Application Protocol) on kevyt verkkosiirtoprotokolla, joka on suunniteltu erityisesti rajallisille laitteille ja vähävirtaisille verkoille, mikä tekee siitä sopivan IoT sovelluksiin. CoAPin käyttää pyyntö-vastausmallia, joka muistuttaa HTTP:tä, mutta on optimoitu säästämään kaistanleveyttä ja vähentämään virrankulutusta. CoAP hyödyntää yksinkertaista otsikkorakennetta, jolloin IoT-laitteet voivat kommunikoida tehokkaasti myös rajoitetuissa verkoissa. CoAP on yhteensopiva HTTP:n kanssa, mikä tekee siitä hyödyllisen sovelluksissa, joissa IoT-laitteiden on oltava yhteydessä tavanomaisiin verkkopalveluihin. [50]

CoAP sisältää useita ominaisuuksia, jotka parantavat sen käyttöä IoT-sovelluksissa, kuten asynkroninen viestintä ja monilähetyksen tuki, jotka mahdollistavat tehokkaan tiedonjaon useiden laitteiden kesken. CoAP:n yksinkertaisuus ja tehokkuus tekevät siitä hyödyllisen IoT-sovelluksissa, kuten älykodeissa ja teollisuusautomaatioteknologioissa, joissa energiatehokkuus ja luotettava viestintä ovat tärkeitä. Turvallisen viestinnän varmistamiseksi laitteiden välillä, voidaan hyödyntää lisäturvamekanismeja. Yksi tällainen mekanismi on DTLS-protokolla, joka suojaa viestintää sovellustasolla tarjoamalla tietoturvaa, kuten eheyden, tunnistamisen ja luottamuksellisuuden. [50]

CoAPissa on kuitenkin rajoitetut sisäänrakennetut tietoturvaominaisuudet, joten se tarvitsee usein lisäsuojausta [50]. Yhdistelmä yksinkertaisuutta, tehokkuutta ja lisäturvaa tekee CoAPista arvokkaan IoT-sovelluksissa, erityisesti älykaupungeissa, ympäristön valvonnassa ja teollisuusautomaatiosta, joissa energiatehokas ja luotettava viestintä on olennaista.

## 3.5 Käytettävyyden standardisointi

Käytettävyyden standardisointi pyrkii varmistamaan, että järjestelmät ja sovellukset ovat intuitiivisia, tehokkaita ja yhdenmukaisia käyttäjän näkökulmasta. Tämä luku esittelee, kuinka standardit voivat ohjata käyttöliittymien ja vuorovaikutusmallien kehitystä sekä parantaa käyttäjäkokemusta.

Suurin osa huonoista rajapintatoteutuksista johtuvat huolimattomuudesta, tietämättömyydestä tai välinpitämättömyydestä kehitysvaiheessa. Tämä on yleinen ongelma etenkin suurissa tuotekehitysorganisaatioissa. Todella usein edellä mainittuja standardeja käyttävät suuret tuotekehitysorganisaatiot, joissa kehitystä ohjaa helppo markkinointi ja yksinkertainen suunnittelu.[51]

On tärkeää huomata, että käyttöliittymien ja käytettävyyden standardien luominen poikkeaa merkittävästi tietokoneiden välisten rajapintastandardien kehittämisestä. Kun tietokoneiden välisissä standardeissa keskitytään yhteentoimivuuden varmistamiseen tarkasti määritellyillä viestintäsäännöillä, ihmisen ja tietokoneen vuorovaikutukseen liittyvissä ohjeistuksissa ja standardeissa korostetaan käyttäjäkokemuksen sujuvuutta ja ymmärrettävyyttä. Tämä asettaa erilaisia haasteita, koska ihmisten tarpeet, taidot ja odotukset vaihtelevat huomattavasti. Lisäksi näitä ohjeistuksia ei tule nähdä lupauksina hyvästä käytettävyydestä, vaan ennemminkin suosituksina, jotka voivat auttaa kehittäjiä suunnittelussa. Käytettävyysstandardien ja -ohjeiden noudattaminen ei yksinään takaa hyvää lopputulosta.

### 3.5.1 Nielsenin 10 heuristiikkaa

Nielsenin 10 käytettävyyden heuristiikkaa ovat käytettävyysekspertti Jakob Nielsenin kehittämä joukko laajalti tunnustettuja periaatteita käyttäjäystävällisten käyttöliittymien suunnitteluun ja arviointiin. Nämä heuristiikat tarjoavat yleisiä ohjeita käytettävyyden parantamiseksi keskittymällä esimerkiksi palautteeseen, hallittavuuteen ja johdonmukaisuuteen. Ne tarjoavat viitekehyksen, jonka avulla suunnittelijat voivat luoda käyttöliittymiä, jotka ovat intuitiivisia, tehokkaita ja käyttäjille miellyttäviä. Noudattamalla näitä periaatteita kehittäjät voivat parantaa käyttäjäkokemusta. [52]

Nielsenin 10 heuristiikkaa ovat:

1. `Järjestelmän tilan näkyvyys:` Pidä käyttäjät aina ajan tasalla siitä, mitä tapah-

tuu. Anna sopivaa palautetta kohtuullisessa ajassa (esimerkiksi edistymispalkki).

2. `Järjestelmän ja todellisen maailman vastaavuus:` Käytä käyttäjän kieltä. Käytä sanoja ja ilmauksia, jotka ovat käyttäjälle tuttuja. (esimerkkinä roskakorimetafora käyttöjärjestelmässä)

3. `Käyttäjän hallinta ja vapaus:` Anna käyttäjän valita toimintojen järjestys (lomakkeen täyttäminen). Tukee toimintoja 'kumoa' ja 'tee uudelleen'.

4. `Johdonmukaisuus ja standardit:` Käyttäjien ei pitäisi joutua pohtimaan, tarkoittavatko eri sanat, tilanteet tai toiminnot samaa asiaa. On oltava selvää, mitkä elementit ovat interaktiivisia. (napit näyttävät napeilta)

5. `Virheiden ehkäisy:` Selkeät virheilmoitukset ovat tärkeitä. Vielä parempaa on huolellinen suunnittelu, joka estää virheet.

6. `Tunnistaminen muistinvaraisen muistamisen sijaan:` Tee objektit, toiminnot ja valinnat näkyviksi. Ohjeiden tulisi olla näkyvillä tai helposti saatavilla.

7. `Joustavuus ja käytön tehokkuus:` Salli käyttäjien mukauttaa usein toistuvia toimintoja. Järjestelmän tulisi palvella sekä aloittelevia että kokeneita käyttäjiä. (vihjeet pikanäppäimistä)

8. `Esteettinen ja minimalistinen suunnittelu:` Selkeä ja tyylikäs käyttöliittymä on miellyttävämpi kuin sekava. Sen ei tulisi sisältää tarpeetonta tietoa.

9. `Auta käyttäjiä tunnistamaan, diagnosoimaan ja korjaamaan virheitä:` Virheilmoitusten tulisi olla selkeää kieltä. Ehdota ratkaisua rakentavasti ja osoita ongelma täsmällisesti.

10. `Ohjeet ja dokumentaatio:` Ohjeiden tulisi olla helposti haettavissa. Ohjeet ja dokumentaatio tulisi olla saatavilla tarvittaessa.

Hyvä UX-suunnittelu korostaa myös käyttäjän hallintaa ja vapautta tarjoamalla mekanismeja virheiden kumoamiseen, sekä johdonmukaisuutta ja standardeja, jotka varmistavat yhtenäisen käyttäytymisen järjestelmän eri osissa. Nämä periaatteet liittyvät tiiviisti käyttöliittymästandardien tavoitteisiin, joissa johdonmukaisuus ja yhteensopivuus järjestelmien välillä ovat kriittisiä sujuvien ja tehokkaiden käyttäjävuorovaikutusten kannalta.

Lisäksi virheiden ehkäisy ja tunnistaminen pikemminkin kuin muistin varaan jättäminen ovat keskeisiä kognitiivisen kuormituksen vähentämisessä ja käyttöliittymien helpottamisessa. Vähentämällä virheiden todennäköisyyttä ja tekemällä avainominaisuudet helposti

näkyviksi suunnittelijat luovat järjestelmiä, jotka ovat sekä intuitiivisia että luotettavia. Tämä ei ainoastaan vähennä käyttäjien turhautumista, vaan myös varmistaa, että standardoidut järjestelmät ovat helpommin saavutettavissa laajemmalle yleisölle.

Avun ja dokumentaation tarjoaminen tukee käyttäjiä heidän kohdatessaan vaikeuksia, ja selkeät, hyödylliset virheilmoitukset ohjaavat heitä nopeasti ongelmien ratkaisemisessa. Nämä heuristiikat varmistavat, että jopa monimutkaisissa järjestelmissä, käyttäjät voivat navigoida, oppia ja toipua virheistä tehokkaasti, mikä edistää saumatonta käyttäjäkokemusta.

### 3.5.2 TURF

Useat ohjelmistot, jotka vaativat korkeantason turvallisuutta ja luotettavuutta, ovat monimutkaisia ja usein hankalia käyttää. Tämä on erityisesti nähtävissä sähköisten potilastietojärjestelmien kanssa. Monet sovellukset vaativat laajaa koulutusta ja eivät noudata kaikkia "hyvän käyttöliittymän"periaatteita. Kliinikot, jotka työskentelevät useilla osa-aluiella ja eri järjestelmien kanssa, voivat kokea järjestelmien välisten erojen hankalaksi ja pahimmassa tapauksessa vaarantaa potilasturvallisuuden. [53]

TURF-viitekehys (Task, User, Representation, Function) käsittelee käytettävyyden ydinalueita jakamalla sen käytännöllisiin osa-alueisiin, joita voidaan analysoida ja optimoida. TURF parantaa käytettävyyttä varmistamalla, että järjestelmän suunnittelu vastaa läheisesti käyttäjien tarpeita ja todellisia tehtäviä. Se keskittyy neljään keskeiseen osa-alueeseen: [54]

1. `Tehtävä:` TURF varmistaa, että järjestelmä tukee käyttäjien suoritettavia tehtäviä. Käytettävän järjestelmän on sovitettava toiminnallisuutensa käyttäjän työnkulkuun, poistamalla tarpeettomat vaiheet ja tekemällä tehtävistä tehokkaampia.

2. `Käyttäjä:` Viitekehys ottaa huomioon käyttäjien taidot, mieltymykset ja rajoitukset. Keskittymällä siihen, kuka järjestelmää käyttää, TURF varmistaa, että käyttöliittymä on suunniteltu niin, että se on kaikkien käyttäjien saavutettavissa ja opittavissa, olivatpa he aloittelijoita tai kokeneita käyttäjiä.

3. `Edustus/representation:` Yksi TURF:in vahvuuksista on sen keskittyminen siihen, miten tietoa esitetään järjestelmässä. Tietojen esittämistavalla on suuri vaikutus käytettävyyteen. TURF auttaa suunnittelijoita valitsemaan tehokkaimmat esitystavat, vähentäen kognitiivista kuormitusta ja parantaen tehtävien suorituskykyä.

4. `Funktio:` Järjestelmän toiminnallisuuden tulisi vastata käyttäjien tarpeita ja mahdollistaa tehtävien suorittaminen tehokkaasti. TURF arvioi järjestelmän toimintoja varmistaakseen, että ominaisuudet tukevat suoraan käyttäjän tavoitteiden saavuttamista eivätkä monimutkaista työnkulkua.

Näiden periaatteiden avulla TURF auttaa suunnittelijoita tunnistamaan käytettävyyden parantamisalueet ja tarjoaa rakenteellisen lähestymistavan intuitiivisempien järjestelmien luomiseen. Monimutkaisissa järjestelmissä, kuten potilastietojärjestelmissä, joissa käytettävyysongelmat voivat vaikuttaa suoraan tuottavuuteen ja käyttäjätyytyväisyyteen, TURF varmistaa, että järjestelmä ei ole vain toiminnallisesti vahva, vaan myös käyttäjäkeskeinen ja tehokas.

TURF:in soveltaminen suunnitteluprosessiin johtaa parempaan tehtävien kohdentamiseen, parantuneeseen käyttäjäkokemukseen ja tehokkaampaan käyttöliittymän esittämiseen, jotka ovat kaikki olennaisia käytettävyyden parantamiseksi. Järjestelmien muuttuessa yhä integroidummiksi ja monimutkaisemmiksi, viitekehykset, kuten TURF, ovat välttämättömiä sen varmistamiseksi, että teknologia pysyy käytettävänä, saavutettavana ja käyttäjien tarpeita vastaavana. [54]

### 3.5.3 ISO 9241-11

ISO 9241-11 on kansainvälinen standardi, joka määrittelee käytettävyyden osana laajempaa ihmisen ja järjestelmän vuorovaikutusta. Se tarjoaa ohjeita järjestelmän käytettävyyden arviointiin kolmen keskeisen osatekijän kautta: tehokkuus, tuottavuus ja tyytyväisyys [55]. Hyvä käytettävyys saavutetaan, kun käyttäjät pystyvät suorittamaan tehtävät tarkasti ja virheettömästi, mahdollisimman pienellä vaivalla ja kun heidän kokemuksensa on positiivinen. ISO 9241-11 standardi korostaa, että käytettävyys ei rajoitu vain käyttöliittymään, vaan kattaa koko järjestelmän vuorovaikutuksen käyttäjien kanssa tietyissä konteksteissa.

Yksi ISO 9241-11 -standardin mukaan käytettävyys on riippuvainen kontekstista, mikä tarkoittaa, että järjestelmä voi olla erittäin käytettävä yhdessä ympäristössä mutta ei toisessa. Tämä konteksti sisältää käyttäjien ominaisuudet, suoritettavat tehtävät ja fyysisen tai sosiaalisen ympäristön, jossa järjestelmää käytetään. Ottamalla huomioon nämä muuttujat ISO 9241-11 tarjoaa viitekehyksen, jolla käytettävyyttä voidaan mitata ja mukauttaa käyttäjien erityistarpeiden ja -olosuhteiden mukaan. Tämä lähestymistapa varmistaa, että käytettävyysarvioinnit ovat relevantteja ja perustuvat todellisiin käyttötapauksiin eivätkä

abstrakteihin, yksi koko kaikille -periaatteisiin. [56]

Käytännössä ISO 9241-11 -standardi toimii pohjana käyttäjäkeskeisten suunnittelujen luomiselle, rohkaisten suunnittelijoita huomioimaan todelliset olosuhteet, joissa järjestelmiä käytetään. Se tarjoaa rakenteelliset kriteerit, joiden avulla organisaatiot voivat arvioida ja parantaa tuotteidensa käytettävyyttä. Keskittymällä käyttäjien tarpeisiin ja tehtävien suorittamiseen tietyissä ympäristöissä tämä standardi auttaa varmistamaan, että järjestelmät eivät ole vain toiminnallisia, vaan myös intuitiivisia, tehokkaita ja käyttäjille miellyttäviä käyttää. Näin ISO 9241-11 ohjaa käytettävyyden parantamista eri toimialoilla, kuten terveydenhuollossa ja rahoituksessa, ja varmistaa, että järjestelmät suunnitellaan käyttäjät ja konteksti huomioiden.

## 3.6 Kyberturvallisuuden standardointi

Kyberturvallisuuden standardisointi keskittyy tarjoamaan menetelmiä ja protokollia, joilla varmistetaan tietojen turvallisuus ja järjestelmien eheys. Tässä luvussa käsitellään muutamia standardeja ja niiden roolia digitaalisen infrastruktuurin suojaamisessa.

Kyberturvallisuuden standardit, kuten OAuth 2.0, PKI ja AES, valittiin niiden laajan käytön, vakiintuneen aseman ja kyvyn tarjota tehokkaita ratkaisuja monimutkaisiin tietoturvahaasteisiin. OAuth 2.0 soveltuu erityisesti verkkopalveluiden pääsynhallintaan, PKI rakentaa luottamusta salausteknologioilla, ja AES tarjoaa korkean tason salauksen tiedonsiirtoon ja -tallennukseen. Nämä standardit edustavat erilaisia mutta toisiaan täydentäviä lähestymistapoja turvallisuuden parantamiseen.

### 3.6.1 OAuth 2.0

OAuth 2.0 on vakiintunut protokolla, joka tarjoaa standardoidun tavan valtuuttaa kolmansien osapuolten sovelluksia pääsemään käyttäjän resursseihin ilman, että käyttäjän salasana paljastuu. Tämä protokolla on erityisesti suunniteltu mahdollistamaan turvallinen pääsy API-rajapintoihin ja yksinkertaistamaan käyttäjien kirjautumisprosessia useissa sovelluksissa. [57]

Toiminta perustuu valtuutustunnusten käyttöön, joissa käyttäjä myöntää sovellukselle oikeudet käyttää tiettyjä resursseja omasta puolestaan. Tämä tapahtuu yleensä käyttäjien syöttäessä kirjautumistiedot kerran ja antamalla sitten sovellukselle pääsyn rajattuihin

resursseihin, kuten profiilitietoihin tai sähköposteihin. Tällä tavalla OAuth 2.0 vähentää salasanojen hallinnan riskejä ja parantaa tietoturvaa. [57]

OAuth 2.0 käyttää useita erilaisia valtuutustyyppisiä virkoja, kuten valtuutuskoodia, suoraa pääsykoodia ja asiakastunnusta, riippuen käyttötapauksista ja sovelluksen vaatimuksista. Näiden mekanismien avulla voidaan mukauttaa autentikointiprosesseja ja hallita käyttöoikeuksia tarkasti. Protokolla on laajalti käytössä monilla verkkosovelluksilla ja -palveluilla, koska se mahdollistaa sujuvan ja turvallisen pääsyn käyttäjän tietoihin ilman monimutkaista salasananhallintaa. [57]

Esimerkiksi FDX API hyödyntää OAuth 2.0 -protokollaa taloudellisen tiedon jakamiseen avoimessa pankkiekosysteemissä [33]. Käyttäjä voi valtuuttaa henkilökohtaisen talouden-hallintasovelluksen pääsemään pankkitilinsä tietoihin turvallisesti. OAuth 2.0:n avulla pankki toimii valtuutuspalvelimena ja tuottaa pääsytunnuksen käyttäjän suostumuksen jälkeen. Tämä tunnus sallii sovelluksen käyttää rajattua joukkoa tietoja, kuten tilitapahtumia, paljastamatta käyttäjän pankkitunnuksia. Tämä lähestymistapa takaa sekä tietoturvan että käyttäjän hallinnan tiedonjaosta.

### 3.6.2 PKI

Julkisen avaimen infrastruktuuri (PKI eng. Public key infrastructure) on korkeantason tietoturvakehys, joka hyödyntää epäsymmetristä salausta viestinnän turvaamiseen ja identiteettien todentamiseen käyttämällä kahta avaintyyppiä: julkista avainta, joka jaetaan avoimesti, ja yksityistä avainta, joka pidetään salassa [58]. PKI mahdollistaa turvallisen tiedonvaihdon antamalla käyttäjille mahdollisuuden salata ja purkaa tietoja näiden avainten avulla, varmistaen, että vain valtuutetut osapuolet voivat käyttää arkaluonteisia tietoja. Luotettujen varmenteiden myöntäjien myöntämät digitaaliset varmenteet vahvistavat käyttäjien ja laitteiden aitouden. Tämä prosessi on välttämätön tiedon suojaamiseksi sen kulkiessa verkon kautta, erityisesti ympäristöissä, joissa tietoturvan on oltava korkea.

PKI on erityisen suosittu IoT-ympäristöissä [59], joissa tarvitaan tehokkaita ja skaalautuvia ratkaisuja laitteiden autentikointiin ja viestinnän salaamiseen. Sen avulla IoT-laitteet, kuten älylukot ja valvontakamerat, voivat tunnistaa toisensa luotettavasti ja varmistaa, että tiedonsiirto on suojattua ja luottamuksellista. Tämä tekee PKI:sta keskeisen teknologian IoT-ekosysteemin turvallisuuden varmistamisessa.

### 3.6.3 AES

Advanced Encryption Standard (AES eng. Advanced Encryption Standard) on yksi laajimmin käytetyistä salausmenetelmistä, joka tarjoaa vahvan tietoturvan sovelluksille aina henkilökohtaisista laitteista yritystason järjestelmiin. Yhdysvaltojen National Institute of Standards and Technology (NIST) kehitti AES:in vuonna 2001. Se on symmetrinen salausalgoritmi, mikä tarkoittaa, että samaa avainta käytetään sekä salaukseen että salauksen purkuun. AES avain voi olla joko 128, 192 tai 256 bittiä pitkä. [60]

AES on erityisen arvokas tilanteissa, joissa tietojen luottamuksellisuus on ratkaisevan tärkeää, kuten rahoitustransaktioissa, potilastiedoissa ja valtion viestinnässä. Sitä käytetään yleisesti suojaamaan arkaluonteisia tietoja siirrettäessä. AES:n korkea suorituskyky mahdollistaa suurten tietomäärien suojaamisen nopeasti, mikä tekee siitä sopivan sekä reaaliaikaisten sovellusten että turvallisen tietojen tallennuksen käyttöön. Sen laaja käyttöönotto ja integrointi moniin tietoturvaprotokolliin ja -standardeihin korostavat sen merkitystä tietosuojan varmistamisessa eri toimialoilla.

AES:ää hyödynnetään parantamaan IoT-laitteiden tietoturvaa, esimerkiksi suojaamaan palvelunestohyökkäyksiltä (DoS) ja varmistamaan tiedonsiirron luottamuksellisuus [61]. Se tarjoaa tehokkaan ratkaisun suojaamaan IoT-verkkoja ja -laitteita yleisiltä turvallisuusuhilta, kuten luvattomalta pääsyltä ja tietovuodoilta.

# 4 Kysely ja haastattelut

Tässä luvussa esitetään tutkimusta varten tehty kysely ja haastattelut. Niiden avulla kerättiin kehittäjien näkemyksistä rajapintastandardien vaikutuksesta ohjelmistojen käytettävyyteen, tietoturvaan ja kehitysprosesseihin. Kyselyiden ja haastatteluiden avulla pyrittiin saamaan monipuolisia näkemyksiä ja käytännön kokemuksia, jotka täydentävät teoreettista analyysia ja tarjoavat syvällisemmän ymmärryksen aiheesta. Kysely lähetettiin viidelletoista kehittäjälle usealta eri alalta. Vastauksia saatiin kahdeksalta kehittäjältä, ja kaikilla heillä oli kokemusta rajapintastandardien käytöstä. Kaikille lähetettiin myös pyyntö haastatteluun, mutta ainoastaan kolme osallistui.

## 4.1 Kysymykset

Kysymyksillä (taulukko 4.1) pyrittiin kartoittamaan osallistujien kokemuksia ja näkemyksiä rajapintastandardeista sekä niiden vaikutuksista ohjelmistokehitykseen. Mukana oli kyllä/ei-kysymyksiä, joiden tarkoituksena oli saada suoria vastauksia tiettyihin aiheisiin, kuten yhteensopivuusongelmiin ja standardien vaikutukseen kehitysprosessiin. Lisäksi kysely sisälsi asteikolla 1–5 vastattavia kysymyksiä, joiden avulla arvioitiin vastaajien mielipiteitä ja kokemuksia esimerkiksi standardien käytettävyydestä ja vaikutuksesta työntekoon. Lopuksi kysely sisälsi avoimia kysymyksiä, joiden avulla vastaajat saivat mahdollisuuden kertoa vapaamuotoisesti näkemyksistään ja antaa syvällisempiä vastauksia muun muassa standardisoinnin hyödyistä, haasteista ja kehitysehdotuksista.

Kysymykset keskittyivät erityisesti käytännön kokemuksiin ja näkemyksiin, jotka täydentävät teoreettista analyysia. Kysymykset jaettiin vastaajille englanniksi monipuolisemman otannan varmistamiseksi.

Taulukko 4.1: Kyselyn sisältämät kysymykset

| **Kysymykset** |
|---|
| Q1: How long have you been working in the IT field? |
| Q2: What standard you are working or have worked with? |
| Q3: Rate your experience level in working with standards. (1 = Beginner, 5 = Expert) |
| Q4: How do you feel about standardization in the field of software development? (1 = Hate it, 5 = Love it) |
| Q5: Did following a standard affect your productivity? (1 = Worsened significantly, 5 = Greatly improved)? |
| Q6: How much do you believe interface standards affect software quality and reliability? (1 = Worsened significantly, 5 = Greatly improved) |
| Q7: Does standardized interfaces affect development time in your experience? (1 = Worsened significantly, 5 = Greatly improved)? |
| Q8: Within your system do you need to take other standards into consideration? |
| Q9: Do you encounter interoperability issues with other systems? |
| Q10: Do you use authentication mechanisms for message integrity? |
| Q11: Does standardization make testing easier? |
| Q12: Does standardization make code more difficult to maintain? |
| Q13: Does standardization affect usability and UX design? |
| Q14: Does standardization make for better UX? |
| Q15: Does standardization make UX design easier? |
| Q16: How do you ensure good UX? |
| Q17: How do you feel about the impact of standardization on innovation in software development? |
| Q18: Does future updates to interface standards affect the planning and design phases or is it typically overlooked? |
| Q19: What challenges or limitations have you encountered due to the standardization of interfaces? |
| Q20: In your opinion, how could the current standards in your field be improved to better meet the evolving needs of developers and the industry? |

## 4.2 Haastattelu

Osana tätä tutkielmaa haastateltiin kolmea kehittäjää, joilla kaikilla oli kokemusta erilaisten rajapintastandardien käytöstä: yksi työskenteli ISO 20022 -standardin parissa, toinen HL7-standardin ja kolmas SCORM-standardin kanssa. Haastattelut järjestettiin verkossa ja kestivät noin 30 minuuttia. Kaikki haastateltavat vastasivat myös kyselyyn, jossa syvennyttiin kyselyn vastauksiin tarkentavien kysymysten avulla. Haastatteluissa käsiteltiin yhteistyötä muiden tiimien kanssa ja sitä, miten kaikki osapuolet noudattavat käytäntöjä ja menetelmiä. Lisäksi selvitettiin, kuinka yrityksen tai liiketoiminnan vaatimukset vaikuttavat heidän työhönsä sekä kuinka he käsittelevät kyberturvallisuuteen liittyviä kysymyksiä.

Haastattelut suunniteltiin alun perin pidettäväksi englanniksi, jotta saatisiin mahdollisimman laaja osallistujajoukko. Kaikki haastateltavat olivat suomalaisia, minkä vuoksi haastattelut päätettiin lopulta toteuttaa suomeksi. Tämä valinta mahdollisti sujuvamman keskustelun ja varmisti, että vastaajat pystyivät ilmaisemaan näkemyksensä mahdollisimman tarkasti ja luonnollisesti.

**Yhteistyö muiden tiimien kanssa**

- How do you typically collaborate with other teams during the development process? Are there any challenges you face when aligning with their practices?
- Can you share an example where differences in approaches or tools between teams affected the progress of a project? How was it resolved?

**Ongelmia muiden järjestelmien integroinnissa**

- What are some of the most common challenges you've encountered when integrating your system with others?
- How do you approach situations where other systems use different versions of the same standard or no standards at all?
- Can you describe a specific instance where integration issues caused delays or required significant changes in your project?

**Kyberturvallisuus**

- What are your main considerations when addressing cybersecurity in systems using standardized interfaces?
- How do you ensure that communication between systems remains secure, especially when integrating with external platforms?
- Have you encountered any cybersecurity vulnerabilities that were directly related to the implementation of a standard? How did you address them?

**Yritysten ja liiketoiminnan vaikutus**

- How do you think business priorities and corporate goals influence the adoption or implementation of standardized systems in your work?
- Have you ever had to compromise on best practices or standards to meet business requirements or deadlines? How did that impact the project?
- In your experience, do corporations and standardization bodies collaborate effectively? What improvements would you suggest?

# 5 Vaikutukset järjestelmän eri alueilla

Rajapintastandardien vaikutukset ulottuvat laajasti ohjelmistokehitykseen ja järjestelmäkehitykseen, vaikuttaen monin tavoin prosessien tehokkuuteen, lopputulosten laatuun ja järjestelmien toimivuuteen. Tämän luvun tavoitteena on tarkastella, miten rajapintastandardit vaikuttavat järjestelmä-ekosysteemien keskeisiin osa-alueisiin. Ensimmäisessä alaluvussa (5.1) tutkitaan niiden merkitystä kehityksen eri vaiheissa, kuten suunnittelussa, ohjelmistoarkkitehtuurissa, toteutuksessa, testauksessa ja käyttöönotossa. Toisessa alaluvussa (5.2) keskitytään siihen, miten standardit parantavat käytettävyyttä ja tukevat käyttäjäkokemuksen kehittämistä. Kolmannessa alaluvussa (5.3) tarkastellaan, miten rajapintastandardit vaikuttavat järjestelmien luotettavuuteen, ja viimeisessä alaluvussa (5.4) syvennytään niiden rooliin kyberturvallisuuden näkökulmasta. Tämä luku pyrkii avaamaan näkökulmia siihen, miten standardit muovaavat järjestelmäkehityksen käytäntöjä.

## 5.1 Vaikutukset kehityksen eri vaiheisiin

Tämä luku vastaa lopullisesti tutkimuskysymyksiin TK1 ja TK2, jotka käsittelevät standardoinnin ratkaisemia ongelmia ja sen vaikutuksia ohjelmistokehityksen prosesseihin. Rajapintastandardit vaikuttavat ohjelmistokehityksen kaikkiin vaiheisiin, alkaen suunnittelusta ja ohjelmistoarkkitehtuurista ja jatkuen toteutukseen, testaukseen ja käyttöönottoon. Ne tarjoavat suuntaviivat tiedonsiirron protokollille ja tietomuodoille, edistävät järjestelmien skaalautuvuutta ja joustavuutta, helpottavat kehitystä valmiiden ratkaisujen avulla sekä tukevat tehokasta yhteensopivuuden varmistamista ja nopeaa käyttöönottoa. Jokainen näistä näkökulmista käsitellään tarkemmin seuraavissa osioissa.

Ohjelmistokehitysprosessit kattavat joukon eri menetelmiä, jotka ohjaavat ohjelmistojärjestelmien suunnittelua, testausta ja käyttöönottoa. Nämä prosessit voivat vaihdella perinteisistä lineaarisista lähestymistavoista, kuten vesiputousmallista, joustavampiin ja iteratiivisempiin menetelmiin, kuten Agile, DevOps tai Scrum. Yksi keskeinen tekijä, joka vaikuttaa kaikkiin kehitysprosesseihin, on rajapintastandardien rooli. Tässä luvussa tarkastellaan, miten rajapintastandardit vaikuttavat ohjelmistokehityksen eri vaiheisiin: suunnittelusta ja ohjelmistoarkkitehtuurista testaukseen ja käyttöönottoon.

Hyvättinen mainitsee [62], standardien varmistavan, että järjestelmät pysyvät yhteentoimivina ja skaalautuvina, mikä mahdollistaa kehittäjille modulaaristen ja mukautuvien

ratkaisujen rakentamisen. Tämä johdonmukaisuus on tärkeää nykyaikaisissa kehitysympäristöissä, joissa tiimit työskentelevät usein hajautettujen järjestelmien parissa ja tarvitsevat kykyä integroida useita palveluja ja alustoja.

Standardit voivat tuoda mukanaan uusia haasteita, kuten joustavuuden vähenemisen, integroinnin monimutkaisuuden ja resurssien vaativuuden käyttöönotossa. Näistä kompromisseista huolimatta standardoinnin vaikutus järjestelmäkehitykseen on pääosin positiivinen, edistäen luotettavuutta ja tehokkuutta.

### 5.1.1 Suunnittelu

Haastatteluista selvisi, että standardien huomioiminen ohjelmistokehityksen suunnitteluvaiheessa on ratkaisevaa, jotta lopputulos on yhteensopiva, sääntöjenmukainen ja laadukas. Ensimmäinen askel on vaatimusten analysointi ja määrittely. Tämä tarkoittaa, että on tunnistettava, mitkä standardit ovat sovellettavissa projektiin sen teollisuuden, laajuuden ja tavoitteiden mukaan. Standardit tulisi sisällyttää projektin vaatimusten dokumentointiin, jotta kaikki sidosryhmät ovat tietoisia niiden merkityksestä ja vaatimuksista.

Suunnitteluvaiheessa on tärkeää integroida standardit järjestelmän suunnitteluun, jotta varmistetaan yhteensopivuus muiden järjestelmien kanssa. Tämä sisältää tietomuotojen, viestintäprotokollien ja integraatiopisteiden määrittelyn sovellettavien standardien mukaisesti. Rajapinnat on määriteltävä niin, että ne toimivat toivotusti [62]. Lisäksi ohjelmistoarkkitehtuurin tulisi tukea sääntöjenmukaisuutta sääntelystandardien osalta suunnittelemalla ominaisuuksia, jotka käsittelevät tietoja turvallisesti, ylläpitävät tarkastusjälkiä tai täyttävät erityiset raportointivaatimukset.

Järjestelmän jatkokehityksen kannalta on tärkeää luoda yksityiskohtainen dokumentaatio, joka kuvaa, miten standardit tullaan soveltamaan kehitysprosessin aikana. Tämä sisältää suunnitteludokumentit, toteutussuunnitelmat ja vaatimustenmukaisuuslistat.

Sovelluksen suunnittelussa on arvioitava mahdollisia riskejä, jotka liittyvät standardien noudattamatta jättämiseen, ja kehittää strategioita näiden riskien lieventämiseksi. Tämä voi tarkoittaa lisäkontrollien toteuttamista tai varasuunnitelmien luomista. Lisäksi on perustettava prosessi standardien noudattamisen seuraamiseksi koko kehitysjakson ajan ja oltava valmis mukauttamaan suunnitelmia uusien standardien tai muutosten myötä.

### 5.1.2 Arkkitehtuuri

Viestintärajapintastandardit vaikuttavat merkittävästi nykyaikaisten ohjelmistojärjestelmien ohjelmistoarkkitehtuuriin tarjoamalla selkeät säännöt ja protokollat komponenttien väliselle viestinnälle. Ohjelmistoarkkitehtuurin ytimessä on tarve rakentaa järjestelmiä siten, että eri komponentit voivat olla vuorovaikutuksessa ja vaihtaa tietoja vaivattomasti [63]. Rajapintastandardit mahdollistavat tämän varmistamalla, että palveluiden välinen viestintä on johdonmukaista ja luotettavaa riippumatta taustalla olevista teknologioista. Tämä mahdollistaa suuremman modulaarisuuden järjestelmän suunnittelussa. Komponentit voidaan kehittää itsenäisesti, mutta ne silti viestivät tehokkaasti standardoitujen rajapintojen kautta.

Rajapintastandardit vaikuttavat merkittävästi ohjelmistoarkkitehtuurien suunnitteluun muokkaamalla järjestelmien välistä vuorovaikutusta ja varmistamalla yhteentoimivuuden. Kysymyksen Q8 (taulukko 5.1) vastaajista kuusi totesi, että sovellusten on otettava huomioon muita standardeja, mikä korostaa tarvetta suunnitella ohjelmistoarkkitehtuureja, jotka mukautuvat ulkoisiin vaatimuksiin. Kysymyksen Q9 (taulukko 5.2) vastaajista viisi kertovat kohtaavansa yhteensopivuusongelmia, mikä tuo esiin haasteen integroida järjestelmiä, jotka käyttävät erilaisia tai vanhentuneita standardeja. Tämä edellyttää joustavia ohjelmistoarkkitehtuuriratkaisuja, jotka voivat hallita tällaisia monimutkaisuuksia samalla säilyttäen järjestelmän luotettavuuden. Lisäksi vastaajat huomauttivat, että standardointi edistää yhteentoimivuutta ja nopeuttaa kehitystä, mutta se voi myös rajoittaa ohjelmistoarkkitehtuurin luovuutta ja innovaatioita. Standardien kehittyvä luonne tuo suunnitteluun lisähaasteita, sillä päivitykset voivat johtaa tekniseen velkaan, jos niitä ei huomioida jo varhaisessa vaiheessa. Ohjelmistoarkkitehtuurien suunnittelussa on tasapainotettava standardien tuoma vakaus ja rajoitteet. Samalla on varmistettava kyky mukautua muutoksiin ja tukea innovaatioita.

Taulukko 5.1: Kysymys Q8

| **Q8: Within your system do you need to take other standards into consideration?** | **Vastausten määrä** |
|---|---|
| Yes | 6 |
| No | 2 |
| Maybe | 0 |

Taulukko 5.2: Kysymys Q9

| **Q9: Do you encounter interoperability issues with other systems?** | **Vastausten määrä** |
|---|---|
| Yes | 5 |
| No | 3 |
| Maybe | 0 |

Hajautetuissa ja palvelusuuntautuneissa ohjelmistoarkkitehtuureissa viestintästandardit ovat olennaisia skaalautuvuuden ja joustavuuden säilyttämisessä. Varmistamalla, että palvelut noudattavat yhteisiä viestintäprotokollia, kehittäjät voivat suunnitella järjestelmiä, jotka ovat löyhästi kytkettyjä [64]. Tämä tarkoittaa, että yksittäisiä komponentteja voidaan päivittää, korvata tai laajentaa vaikuttamatta koko järjestelmään. Tämä joustavuus helpottaa sellaisten järjestelmien rakentamista, jotka voivat kehittyä ajan myötä ja integroida uusia ominaisuuksia tai palveluita samalla säilyttäen järjestelmän yleisen toimivuuden. Standardoidut viestintäprotokollat tukevat myös yhteentoimivuutta, mikä mahdollistaa järjestelmien välisen viestinnän eri alustoilla ja ympäristöissä, mikä parantaa ohjelmistoarkkitehtuurin joustavuutta entisestään.

### 5.1.3 Toteutus

Haastateltujen kehittäjien mukaan koodaus- ja kehitysvaiheessa rajapintastandardit mahdollistavat keskittymisen toiminnallisuuksien toteuttamiseen ilman huolta komponenttien välisen viestinnän toimivuudesta. Standardit tukevat ohjelmistokomponenttien luotettavaa ja tehokasta viestintää tarjoamalla kehittäjille hyvin määritellyt protokollat ja tiedon esitysmuodot, mikä vähentää epäselvyyksiä siitä, miten eri palvelut ja järjestelmät vaihtavat tietoja. Kyselyyn vastanneista viisi kuitenkin kokivat kohtaavansa yhteensopivuusongelmia muiden järjestelmien kanssa (taulukko 5.2). Nämä ongelmat voivat johtua siitä, että kehittäjät tulkitsevat standardien määritykset eri tavoilla.

Yksi standardien käytön merkittävistä eduista kehitysvaiheessa on koodin uudelleenkäytettävyys. Kun kehittäjät noudattavat yhteisiä protokollia, he voivat rakentaa komponentteja, joita voidaan käyttää uudelleen eri järjestelmissä tai projekteissa [63]. Uudelleenkäytettävät komponentit tehostavat kehitystä, vähentävät päällekkäistä työtä ja varmistavat, että kehitystiimit voivat nopeasti koota järjestelmiä modulaarisista, standardoiduista rakennuspalikoista.

Standardointi yksinkertaistaa järjestelmien integrointia ("Improves integration to exis-

ting systems"Q17). Kun kehittäjät noudattavat standardoituja viestintäprotokollia, komponentit voivat kommunikoida turvallisesti ja tehokkaasti, mikä mahdollistaa järjestelmän laajentamisen uusilla toiminnoilla tulevaisuudessa. Tämä varmistaa, että kehitetyt ratkaisut ovat skaalautuvia ja mukautettavia. Yli puolet kyselyyn vastanneista kehittäjistä olivat sitä mieltä, että standardointi huomattavasti parantaa sovellusten laatua ja luotettavuutta (taulukko 5.3.).

Taulukko 5.3: Kysymys Q6

| **Q6: How much do you believe interface standards affect software quality and reliability? (1 = Worsened significantly, 5 = Greatly improved)** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 2 |
| 5 | 5 |

Kyselytulosten perusteella standardisoinnilla on merkittävä vaikutus kehitysajan ja tuottavuuden kannalta ohjelmistokehityksessä. Seitsemän kehittäjää vastasi, että standardien käyttö voi nopeuttaa kehitysprosesseja. Selkeät vaatimukset ja valmiit ratkaisut, joiden avulla kehittäjät voivat välttää “pyörän keksimistä uudelleen”, tehostavat kehitystyötä. Tämä parantaa tuottavuutta erityisesti suurissa projekteissa, joissa on tärkeää noudattaa yhtenäisiä käytäntöjä ja malleja. Standardointi usein parantaa tuottavuutta ja lyhentää kehitysaikaa, sen vaikutus riippuu siitä, kuinka hyvin tiimi on perehtynyt käytettäviin standardeihin ja kuinka selkeät ja joustavat ne ovat toteutukseltaan.

### 5.1.4 Testaus ja laadunvarmistus

Kyselytulosten mukaan standardointi helpottaa ohjelmistojen testausprosessia. Seitsemän kahdeksassta vastaajista totesi, että standardit tarjoavat selkeät puitteet ja määrittelyt, joiden avulla testaus voidaan suorittaa johdonmukaisesti ja tehokkaasti (Taulukko 5.4). Esimerkiksi standardoidut tietomuodot ja viestintäprotokollat vähentävät virheiden määrää ja yksinkertaistavat yhteensopivuuden testaamista eri järjestelmien välillä. Tämä nopeuttaa testausvaihetta, koska kehittäjien ei tarvitse luoda testausympäristöä tyhjästä. Näin ollen standardoinnin vaikutus testaukseen on pääosin myönteinen, mutta sen onnistuminen riippuu standardien selkeydestä ja käytännön soveltuvuudesta.

| **Q11: Does standardization make testing easier?** | **Vastausten määrä** |
|---|---|
| Yes | 7 |
| No | 0 |
| Maybe | 1 |

Kuten artikkelissa [65] mainitaan, yleinen yhteentoimivuuden testauskehys tarjoaa kattavan tavan varmistaa, että viestintästandardien mukainen tietojenvaihto tapahtuu luotettavasti ja oikein. Testaus- ja laadunvarmistusvaiheessa viestintärajapintastandardit varmistavat, että eri komponenttien ja järjestelmien välinen viestintä toimii odotetusti. Yhteentoimivuuden saavuttaminen edellyttää, että järjestelmät noudattavat standardoituja protokollia, mikä helpottaa testauksen automatisointia ja integraatiotestien suorittamista. Tämä auttaa tunnistamaan yhteentoimivuusongelmat jo varhaisessa kehitysvaiheessa, vähentäen virheitä ja tehostaen integraatiota.

Automaattisen testauksen merkitys korostuu erityisesti, kun järjestelmien monimutkaisuus kasvaa. Käyttämällä standardoituja viestintäprotokollia, testikehykset voivat automatisoida useita testitapauksia ja vähentää manuaalisen testauksen tarvetta. Tämä paitsi nopeuttaa testausta, myös parantaa testien kattavuutta ja toistettavuutta. HL7-pohjaisille järjestelmille kehitetty testauskehys [66] korostaa, kuinka automaatiolla voidaan varmistaa järjestelmien johdonmukainen toiminta kaikilla tasoilla, ja varmistaa samalla järjestelmän sääntöjenmukaisuus ja luotettavuus. Tämä tukee myös standardoitujen järjestelmien tehokasta skaalautumista ja laajentamista tulevaisuudessa.

### 5.1.5 Käyttöönotto ja toiminta

Kuusi kyselyyn vastanneista olivat sitä mieltä, että standardit helpottavat koodin ylläpitoa (Taulukko 5.5). Helposti ylläpidettävä koodi voi myös vähentää käyttöönottoon liittyviä ongelmia, kuten integraatiohaasteita. Yksi vastaajista huomautti, että standardien epäjohdonmukaisuudet tai niiden väärinymmärrykset voivat aiheuttaa ongelmia käytännön toteutuksessa ja käyttäjäkokemuksessa, "Although developing with standards, it is needed to make sure everyone has the same understanding of the standards."(Q19) Vaikka standardointi yleensä edistää tehokasta käyttöönottoa ja järjestelmien käyttöä, sen onnistuminen edellyttää huolellista suunnittelua ja kaikkien osapuolten sitoutumista noudattamaan yhteisiä käytäntöjä.

Taulukko 5.5: Kysymys Q12

| **Q12: Does standardization make code more difficult to maintain?** | **Vastausten määrä** |
|---|---|
| Yes | 0 |
| No | 6 |
| Maybe | 2 |

Standardoidut viestintäprotokollat tukevat myös käyttöönoton skaalautuvuutta. Kun järjestelmien koko kasvaa ja tarvitaan lisää resursseja, viestintästandardit varmistavat, että uusia komponentteja voidaan lisätä häiritsemättä olemassa olevia palveluita [67]. Tämä on erityisen hyödyllistä pilviympäristöissä, joissa palvelut skaalautuvat dynaamisesti kysynnän mukaan. Viestintästandardien käyttö mahdollistaa järjestelmien helpon integroinnin kolmannen osapuolen palveluihin tai ulkoisiin rajapintoihin [68] [69]. Tämä vähentää käyttöönoton monimutkaisuutta ja mahdollistaa sujuvammat päivitykset ja laajennukset ajan myötä.

Jatkuvan integroinnin (CI, eng. continuous integration) ja jatkuvan käyttöönoton (CD, eng. continuous deployment) putkistot hyötyvät merkittävästi viestintästandardeista. Vakiintuneiden viestintäprotokollien avulla nämä standardit helpottavat käyttöönottoprosessien automatisointia, varmistaen, että uudet ominaisuudet tai päivitykset voidaan ottaa käyttöön ilman katkoksia palveluissa [70]. Standardointi lisää myös järjestelmien luotettavuutta tarjoamalla selkeät ohjeet viestinnän ylläpitämiseen palveluiden välillä. [71]

## 5.2 Vaikutukset käytettävyyteen

Seuraavaksi esitellään yhteenveto tutkimuskysymykseen TK3 (Millä tavoin rajapintastandardit voivat parantaa käytettävyyttä?) saaduista vastauksista. Tutkimuksen perusteella voidaan todeta, että rajapintastandardit parantavat käytettävyyttä luomalla yhdenmukaisia käytäntöjä ja rakenteita. Käytettävyysstandardit varmistavat, että järjestelmän suunnittelu ottaa huomioon loppukäyttäjien tarpeet, mikä vähentää oppimiskynnystä ja parantaa käyttäjäkokemusta. Lisäksi rajapintastandardit, jotka tukevat selkeitä ja johdonmukaisia tietomuotoja, vähentävät virheiden mahdollisuutta ja helpottavat tiedon tulkintaa. Kehittäjäkokemukseen liittyen standardit tarjoavat valmiit ratkaisut, jotka nopeuttavat kehitystä ja yksinkertaistavat monimutkaisten järjestelmien käyttöä. Näitä aiheita tarkastellaan syvällisemmin seuraavissa osioissa.

Käytettävyys ja käyttäjäkokemus (UX) ovat kriittisiä tekijöitä minkä tahansa ohjelmistojärjestelmän menestykselle. Nämä seikat määrittävät, kuinka helposti ja tehokkaasti käyttäjät voivat olla vuorovaikutuksessa järjestelmän kanssa ja saavuttaa tavoitteensa. Kun järjestelmät suunnitellaan helposti käytettäviksi, ne minimoivat virheitä ja parantavat käyttäjien tyytyväisyyttä.

Suuret organisaatiot kaipaavat laajoja resursseja, todistettua osaamista sekä tiukkaa turvallisuutta omissa IT-palveluissa ja kehityksessä. Tästä syystä usein suuret yritykset voittavat kilpailutukset ja pääsevät tekemään yhteistyötä niiden kanssaan. Vaikka näissä yrityksissä on paljon osaamista monelta osa-alueelta, se saattaa silti aiheuttaa ongelmia toimivan käyttöliittymän suunnittelussa. [51]

Standardien ja isojen yritysten yhdessä asettamat vaatimukset tekevät käyttöliittymän suunnittelusta jäykkää ja haastavaa. [51] Kun käytettävyys tunnustetaan yhä tärkeämmäksi tekijäksi ohjelmistojärjestelmien menestyksessä, on kehitetty useita standardeja varmistamaan, että käyttöliittymät eivät täytä vain teknisiä vaatimuksia. Niiden tulee olla myös intuitiivisia ja käyttäjäystävällisiä. Nämä käytettävyysstandardit tarjoavat rakenteellisia ohjeita ja kriteerejä, jotka auttavat suunnittelijoita ja kehittäjiä luomaan järjestelmiä, jotka ovat tehokkaita, toimivia ja käyttäjille miellyttäviä. Niiden käyttö ei kuitenkaan takaa hyvää lopputulosta. [72]

Kyselytulosten perusteella standardisoinnilla on vaikutuksia käytettävyyteen ja käyttäjäkokemukseen (Taulukko 5.6). Useimmat vastaajat kokivat, että standardit parantavat käyttäjäkokemusta (Taulukko 5.7). Yhtenäiset käytännöt ja rakenteet, jotka helpottavat järjestelmien ymmärrettävyyttä ja käytettävyyttä, parantavat käyttökokemusta. Tämä voi erityisesti näkyä rajapintojen suunnittelussa, jossa standardit varmistavat yhtenäiset toimintatavat ja selkeän käyttöliittymän. Toisaalta jotkut vastaajat huomauttivat, että standardoinnin jäykkyys voi joskus rajoittaa suunnittelun joustavuutta, mikä saattaa vaikuttaa negatiivisesti käyttäjäkokemukseen, jos standardit eivät ole riittävän joustavia vastaamaan loppukäyttäjien tarpeisiin, "If developers are forced to follow overly strict rules, it prevents them from experimenting with new approaches or using creative solutions to problems"(Q17).

Taulukko 5.6: Kysymys Q13

| **Q13: Does standardization affect usability and UX design?** | **Vastausten määrä** |
|---|---|
| Yes | 4 |
| No | 2 |
| Maybe | 2 |

Taulukko 5.7: Kysymys Q14

| **Q14: Does standardization make for better UX?** | **Vastausten määrä** |
|---|---|
| Yes | 6 |
| No | 2 |
| Maybe | 0 |

Vastaajat pitivät hyviä suunnittelijoita ja selkeitä suunnitteluprotokollia tärkeimpinä tekijöinä toimivan käyttäjäkokemuksen (UX eng. User Experience) luomisessa (Taulukko 5.8). Vaikka standardit tarjoavat hyödyllisen pohjan, laadukas UX edellyttää suunnittelijoilta taitoa soveltaa käytettävyysstandardeja käyttäjien tarpeiden mukaisesti. Erityisesti yleiset heuristiikat, kuten Nielsenin 10 heuristiikkaa, ja suunnittelun parhaat käytännöt nähtiin hyödyllisinä työkalupakkeina, jotka varmistavat järjestelmien selkeyden, käytettävyyden ja käyttäjien tyytyväisyyden.

Taulukko 5.8: Kysymys Q16

| **Q16: How do you ensure good UX?** | **Vastausten määrä** |
|---|---|
| With the same, or another standard | 2 |
| General guidelines and heuristics | 3 |
| Good designers | 4 |
| Other (Measuring success) | 1 |

Kyselyn ja kirjallisuuden avulla saadaan vastaus tutkimuskysymykseen TK3. Rajapintastandardit varmistavat suunnittelun yhdenmukaisuuden, mikä vähentää käyttäjän kognitiivista kuormitusta ja tekee järjestelmistä helpommin ymmärrettäviä ja käytettäviä. Käyttämällä käyttäjäkeskeisen suunnittelun periaatteita, järjestelmiä voidaan mukauttaa tehokkaasti käyttäjien tarpeisiin. Standardit parantavat myös kehittäjäkokemusta tarjoamalla selkeät ohjeet ja protokollat, yksinkertaistaen kehitysprosessia ja varmistaen käytettävyysvaatimusten täyttymisen. Tämä johtaa järjestelmiin, jotka ovat sekä toiminnallisia että käyttäjäystävällisiä, tarjoten saumattoman ja tehokkaan käyttökokemuksen.

### 5.2.1 Suunnittelijoiden rooli

Pelkät standardit eivät voi taata positiivista käyttäjäkokemusta. Koulutetut UX-suunnittelijat, jotka toteuttavat näitä periaatteita yhdessä käyttöliittymästandardien kanssa, voivat tarjota ohjelmistoa, joka on sekä toiminnallista että käyttäjäystävällistä.

Esimerkiksi esteettinen ja minimalistinen muotoilu näyttelee merkittävää roolia tarpeettoman monimutkaisuuden vähentämisessä, kun taas joustavuus ja tehokkuus palvelevat sekä aloittelijoita että asiantuntijoita tarjoamalla oikopolkuja ja mukautettavia työnkulkuja.

### 5.2.2 Kehittäjäkokemus

Kehittäjäkokemus (DX eng. Developer Experience) on olennainen tekijä ohjelmistokehityksen onnistumisessa, erityisesti kun työskennellään rajapintastandardien kanssa. Hyvä kehittäjäkokemus voi varmistaa, että integraatioprosessi sujuu tehokkaasti ja joustavasti. Standardien tavoitteena on, että kehittäjät voivat keskittyä toiminnallisuuden rakentamiseen ilman, että heidän tarvitsee jatkuvasti käsitellä järjestelmien välisiä yhteysongelmia. [73]

Kyselyssä korostui, että standardit voivat sekä helpottaa että hankaloittaa kehittäjäkokemusta riippuen niiden selkeydestä ja toteutuksesta. Yksi vastaaja mainitsi, että hyvin määritellyt standardit yksinkertaistavat kehitystyötä tarjoamalla valmiita ratkaisuja, mikä vähentää kehittäjien työkuormaa ja parantaa tuottavuutta ("From a positive perspective, standards can promote interoperability, improve quality, and accelerate development, as developers can leverage common patterns and interfaces instead of reinventing the wheel. This can free up time and resources to focus on new ideas and development directions"Q17). Toisaalta kehittäjät kohtasivat myös haasteita, kuten standardien tulkinnanvaraisuutta ja monimutkaisuutta, jotka voivat aiheuttaa sekaannusta ja lisätä kehitystyön vaativuutta (Taulukko 5.2). Vastausten mukaan standardien hyödyllisyys kehittäjäkokemuksen parantamisessa riippuu suuresti niiden käytön helppoudesta ja siitä, kuinka hyvin ne on sovitettu projektin tarpeisiin.

Yksi keskeinen tekijä, joka parantaa kehittäjäkokemusta, on selkeä ja kattava dokumentaatio [73]. Viestintästandardeja käytetään usein eri alustoilla, ja kehittäjät tarvitsevat yksityiskohtaista ja helposti saatavilla olevaa dokumentaatiota, jotta he voivat toteuttaa ne oikein. Tehokas dokumentaatio tarjoaa selkeät ohjeet integroinnista, virheenkäsittelystä ja standardien käytöstä, mikä helpottaa oppimista ja vähentää virheiden mahdollisuutta. Kun standardit ovat johdonmukaisia ja hyvin dokumentoituja, kehittäjät voivat hyödyntää aiemmissa projekteissa opittua, mikä nopeuttaa uusien standardien omaksumista.

Toinen tärkeä tekijä hyvän kehittäjäkokemuksen varmistamisessa on työkalujen ja tuen

saatavuus, jotka helpottavat viestintästandardien toteutusta ja kehitysympäristön sujuvuutta. Fagerholmin ja Münchin mukaan [73] kehitystyökalut ja ohjelmistokehykset eivät pelkästään lisää tehokkuutta, vaan myös vähentävät kognitiivista kuormitusta, jolloin kehittäjät voivat keskittyä ongelmanratkaisuun ja koodin laatuun. Työkalut, jotka tukevat testaus-, virheenkorjaus- ja integraatioprosesseja, auttavat varmistamaan tehokkaan kehitysprosessin. Lisäksi aktiiviset yhteisöt, keskustelufoorumit ja muut tuen muodot tarjoavat tärkeää apua kehittäjille ongelmien ratkaisemiseen ja tiedon jakamiseen, mikä parantaa yhteistyötä ja rohkaisee innovaatiota. Kun viestintästandardeja tukevat vahvat työkalut ja selkeät ohjeistukset yhdistetään yhteisölliseen tukeen, kehittäjät voivat saavuttaa parempia tuloksia ja kohdata vähemmän haasteita.

Yksi haastateltavista kehittäjistä korosti, että standardien päivittäminen vaikuttaa järjestelmien suunnitteluun ja kehitykseen ja niiden tulisi säilyttää yhteensopivuus aiempien versioiden kanssa, jotta vältytään 'tekniseltä velalta' ("If standards are updated, it would be wise to consider their impact and incorporate them into the planning and design phases. This is one of the points where "technical debt"can accumulate if not properly addressed"Q18). Lisäksi taaksepäin yhteensopivuus ja versiointi ovat keskeisiä hyvän kehittäjäkokemuksen ylläpitämisessä, erityisesti tilanteissa, joissa standardit kehittyvät tai muuttuvat ajan myötä. Samalla on tärkeää, että päivitykset otetaan huomioon jo suunnitteluvaiheessa, jotta mahdolliset vaikutukset voidaan arvioida ajoissa. Selkeä versiointistrategia, joka mahdollistaa uusien ominaisuuksien sujuvan käyttöönoton ja vähentää teknisen velan kertymistä, on olennaista. Tämä auttaa kehittäjiä pysymään ajan tasalla standardien muutoksista ja takaa järjestelmien vakauden kehityksen kaikissa vaiheissa.

## 5.3 Vaikutus järjestelmän luotettavuuteen

Tämä luku tarjoaa osittaisen vastauksen tutkimuskysymykseen TK4. Rajapintastandardit parantavat tiedonlaadun ylläpitoa tarjoamalla yhteiset säännöt tiedonsiirrolle, mikä parantaa tiedon tarkkuutta ja luotettavuutta. Tiedon tarkkuus ja luotettavuus ovat keskeisiä järjestelmän luotettavuuden osatekijöitä, joita rajapintastandardit tukevat. Toisaalta standardien eri tulkinnat voivat aiheuttaa riskejä, kuten datan epäjohdonmukaisuutta tai virheitä, jotka voivat heikentää järjestelmän toimintavarmuutta. Näiden näkökulmien yksityiskohtainen tarkastelu jatkuu seuraavissa osioissa, joissa syvennytään niiden vaikutuksiin järjestelmien luotettavuuteen.

Tietojärjestelmien tiedonlaadulla (IQ, eng. Information Quality) on keskeinen merkitys ohjelmistojen ja järjestelmien toiminnallisuudessa. Tiedonlaatu määrittää, kuinka tark-

kaa, johdonmukaista ja käyttökelpoista tieto on erilaisissa käyttötarkoituksissa. Laadukas tieto mahdollistaa paremman päätöksenteon, tehokkaammat prosessit ja sujuvamman käyttäjäkokemuksen. Informaation tallentaminen ja sen hoitaminen ei kuitenkaan ole triviaalia. Kuten Caballero mainitsee artikkelissaan [74], sen ylläpitäminen ja hyödyntäminen vaatii paljon resursseja.

Josyula [75] korostaa rajapintastandardien tärkeää asemaa tiedonlaadun ylläpitämisessä. Ne luovat yhteiset säännöt tietojen vaihdolle, mikä parantaa tiedon johdonmukaisuutta ja tulkittavuutta. Noudattamalla standardoituja protokollia järjestelmät voivat vaihtaa tietoa luotettavasti ja tehokkaasti, mikä tukee tietojen eheyttä ja yhtenäisyyttä. Tämä johdonmukaisuus vähentää virheiden mahdollisuutta tiedonsiirrossa, mikä lisää tiedon laatua ja luotettavuutta. Lisäksi standardien tuoma yhteentoimivuus ja tiedon uudelleenkäytettävyys helpottavat tiedon yhdistelyä ja hyödyntämistä eri alustoilla.

Esimerkiksi yritykset, kuten VISA Inc., asettavat vaatimuksia tuotteidensa liittyvien sovellusten toimintaan ja käyttäjäkokemukseen. Maksutapahtumissa siirrettävän tiedon laatu vaikuttaa suoraan asiakkaan kokemukseen, ja virheet tai viivästykset tiedonsiirrossa voivat heikentää asiakastyytyväisyyttä.[76]

### 5.3.1 Hyödyt ja riskit

D'Amore et al. [77] korostavat, että tallennettu tieto on olennainen osa ohjelmistojärjestelmien menestystä ja luotettavuutta, vaikuttaen suoraan päätöksentekoon, toiminnan tehokkuuteen ja käyttäjätyytyväisyyteen. Kun tieto on tarkkaa, johdonmukaista ja hyvin jäsenneltyä, se mahdollistaa organisaatioille paremman päätöksenteon, trendien tunnistamisen ja prosessien optimoinnin. Esimerkiksi terveydenhuollossa tarkat potilastiedot voivat parantaa diagnoosien tarkkuutta ja johtaa parempiin hoitotuloksiin. Samoin Josyula [75] tuo esille, että rahoitusalalla tarkka tieto transaktioista ja tileistä varmistaa säädösten noudattamisen ja parantaa tehokkuutta, käytettävyyttä ja yhteentoimivuutta. Tiedonlaatu ei ainoastaan ohjaa tehokasta liiketoimintaa, vaan se vähentää myös tarvetta laajalle tiedon tarkistamiselle ja virheenkorjaukselle, mikä alentaa kustannuksia ja parantaa järjestelmän tehokkuutta.

Kyselytutkimuksessa vastaajista viisi kertoi kohtaavansa yhteentoimivuusongelmia (Taulukko 5.2). Jos näitä ongelmia ei havaita ajoissa, se voi johtaa virheisiin, väärintulkintoihin ja tehottomuuteen. Kriittisillä toimialoilla, kuten terveydenhuollossa, epätarkat tiedot voivat johtaa virheellisiin diagnooseihin tai väärään hoitoon, mikä vaarantaa potilastur-

vallisuuden [77]. Rahoitusalalla virheellinen tieto voi aiheuttaa säädösten noudattamiseen liittyviä ongelmia, taloudellisia tappioita tai mainehaittoja [75]. Tiedonlaadun ylläpitäminen on ratkaisevan tärkeää riskien vähentämiseksi ja ohjelmistojärjestelmien toiminnan varmistamiseksi monilla eri sovellusalueilla.

### 5.3.2 Tiedon ylläpitäminen

Vastaajista seitsemän uskoo, että standardeilla on merkittävä vaikutus sovellusten luotettavuuteen ja laatuun (Taulukko 5.4), mikä korostaa standardien roolia johdonmukaisen ja tarkan tiedon käsittelyn varmistamisessa. Kuitenkin yhteensopivuusongelmat ovat edelleen haaste monille kehittäjille, mikä osoittaa, kuinka tärkeää on noudattaa yhteisiä standardeja estääkseen poikkeavuuksia tiedon käsittelyssä. Seitsemän vastaajaa oli sitä mieltä, että standardointi helpottaa testausprosessia, mikä varmistaa, että järjestelmät täyttävät johdonmukaiset laatuvaatimukset (Taulukko 5.3). Vaatimusten täyttäminen parantaa tiedon laatua ja luotettavuutta koko järjestelmässä.

Korkealaatuisen tietojen paikkansapitävyyden ja johdonmukaisuuden ylläpitäminen on olennainen osa tietojen luotettavuutta ja käyttökelpoisuutta järjestelmien välillä. Yksi tärkeimmistä keinoista varmistaa tietojen laatu on ottaa käyttöön tehokkaat validointi- ja varmennusmekanismit [77]. Tietojen validointi voidaan suorittaa syöttövaiheessa, jolloin järjestelmät tarkistavat, että tiedot noudattavat määriteltyjä muotoja, arvorajoja tai sääntöjä. Varmennusprosessit, jotka usein integroidaan osaksi järjestelmään, mahdollistavat tietojen tarkistamisen suhteessa vakiintuneisiin vertailuarvoihin tai historiatietoihin, mikä varmistaa tietojen paikkansapitävyyden pitkällä aikavälillä. Nämä mekanismit ovat erityisen arvokkaita korkean riskin aloilla, joilla tarkka ja virheetön tieto on välttämätöntä operatiivisen turvallisuuden ja sääntöjenmukaisuuden kannalta.

Virheenkäsittelyprotokollat ja automaattiset valvontatyökalut auttavat ehkäisemään ja korjaamaan tietojen epäjohdonmukaisuuksia. Virheenkäsittelymekanismit, kuten automaattiset hälytykset ja korjausprosessit, mahdollistavat tietojen poikkeamien nopean tunnistamisen ja korjaamisen [77]. Automaattiset työkalut, jotka jatkuvasti seuraavat tietovirtoja, voivat havaita poikkeavuuksia tai standardeista poikkeavia tietoja reaaliajassa, mikä vähentää virheiden huomaamatta jäämisen riskiä. Näiden strategioiden avulla voidaan varmistaa, että tiedot pysyvät tarkkoina, johdonmukaisina ja luotettavina, mikä tukee järjestelmän toimivuutta ja käyttäjien luottamusta.

## 5.4 Vaikutukset kyberturvallisuuteen

Kirjallisuuden perusteella voidaan vastata tutkimuskysymykseen TK4. Rajapintastandardit eivät takaa turvallista järjestelmää. Ne muodostavat kriittisen osan järjestelmää, ja kuten mikä tahansa osa, ne voivat olla haavoittuvia hyökkäyksille. Tämän vuoksi rajapintastandardit tulee aina yhdistää turvallisuusprotokolliin ja -standardeihin järjestelmän suojaamiseksi.

Vuonna 2011 Xie et al. [78] tekivät useita haastatteluja, ja heidän tutkimuksensa paljasti puutteita ohjelmistoturvallisuuden käsitteiden ja niiden soveltamisen välillä kehittäjien ammatillisissa rooleissa. Myös tätä tutkimusta varten suoritetuissa haastatteluissa kävi ilmi, että kehittäjät eivät tiedä standardoinnin merkitystä kyberturvallisuuteen.

Nykyisessä digitaalisessa ympäristössä kyberturvallisuuden merkitys kasvaa, kun yritykset ja sovellukset pyrkivät yhä yksinkertaisempaan ja nopeampaan järjestelmien integrointiin ja käyttöönottoon. Standardointi on tässä keskeisessä roolissa, sillä se luo yhtenäisiä protokollia ja suojamekanismeja, jotka vahvistavat tiedon suojausta ja järjestelmien vastustuskykyä. Rajapintastandardit suojaavat ja salaavat lähetettyä dataa joko sisäänrakennettujen turvaprotokollien avulla tai hyödyntämällä salausstandardeja, jotka takaavat tietojen luottamuksellisuuden ja eheyden. [79]

HL7 järjestelmien haavoittuvuudet voivat altistaa terveydenhuoltojärjestelmät merkittäville tietomurroille, mikä on erityisen vakavaa, kun otetaan huomioon terveystietojen arkaluontoisuus. Jos HL7-toteutuksia ei suojata kunnolla, se voi johtaa merkittäviin turvallisuusaukkoihin, jotka voivat vaikuttaa miljooniin potilaisiin maailmanlaajuisesti [80]. Samoin ISO 20022 hallitsee tietojenvaihtoa rahoituslaitosten välillä ja varmistaa turvalliset maksutapahtumat. Standardi hyödyntää salausta, petosten havaitsemista ja henkilöllisyyden todentamista, mikä vahvistaa kyberturvallisuutta. Mahdolliset haavoittuvuudet voivat kuitenkin aiheuttaa maailmanlaajuisia vaikutuksia, mikä saattaa vaarantaa koko rahoitusverkostot.

Kaikki tässä tutkielmassa esitellyt rajapintastandardit hyödyntävät muita standardeja ja protokollia, jotta tietoturva pysyy korkealla tasolla [79] [81] [82] [83]. Näin rajapintastandardit toimivat yhdessä tietoturvaprotokollien kanssa varmistaakseen tietojen turvallisuuden. Tällaiset yhteiset käytännöt helpottavat myös sääntelyvaatimusten ja toimialakohtaisten vaatimusten noudattamista. Huolellisesti toteutettujen tietoturvastandardien avulla organisaatiot voivat vahvistaa kyberturvallisuuttaan, vähentää riskejä ja varmistaa tietojensa eheyden ja luottamuksellisuuden. [79]

### 5.4.1 Suojausprotokollat ja -standardit

Yhteensopivien suojausstandardien käyttö vähentää haavoittuvuuksia ja tukee luotettavuutta erityisesti moniorganisaatiollisissa järjestelmissä, joissa jaetaan arkaluonteista tietoa, kuten terveydenhuollon tai rahoitusalan sovelluksissa. Standardit myös yksinkertaistavat turvallisuusmekanismien käyttöönottoa tarjoamalla selkeitä ja testattuja käytäntöjä. Suojausprotokollat ja -standardit varmistavat, että tietojen siirto on suojattua, eheyden säilyttävää ja yhteensopivaa kaikissa rajapinnoissa.

Standardoidut suojausratkaisut, kuten PKI ja AES, varmistavat viestien eheyden ja luottamuksellisuuden sekä tarjoavat tehokkaan tavan salata tiedot niiden siirron aikana, mikä ehkäisee luvattoman käytön ja tietojen muuttamisen. Lisäksi OAuth, joka on erityisesti API käytössä laajasti käytetty standardi, tukee turvallista pääsynhallintaa määrittelemällä, kuka ja miten eri osapuolet voivat käyttää tietoja. Näiden protokollien ja standardien avulla järjestelmät voivat säilyttää datan turvallisuuden ja luotettavuuden myös monimutkaisissa ja hajautetuissa ympäristöissä.

### 5.4.2 Turvallinen viestien käsittely

Tehokas viestinkäsittely on olennaista tiedon eheyden, luottamuksellisuuden ja järjestelmävaatimusten noudattamisen kannalta. Kun järjestelmät vaihtavat tietoja, on tärkeää varmistaa, että jokainen viesti seuraa tiettyjä turvallisuusprotokollia. Tämä estää luvattoman pääsyn, manipuloinnin ja tietojen katoamisen.

Turvattoman viestinkäsittelyn laiminlyönti tuo mukanaan huomattavia riskejä. Jos tietoja lähetetään ilman salausta tai asianmukaista validointia, ne ovat alttiita sieppaukselle luvattomien osapuolten toimesta. Tämä voi johtaa siihen, että arkaluonteiset tiedot altistuvat tai niitä manipuloidaan, mikä vaikuttaa tietojen tarkkuuteen ja luotettavuuteen. Lisäksi turvaton viestinkäsittely voi tehdä järjestelmistä haavoittuvia man-in-the-middle -hyökkäyksille, joissa hyökkääjät sieppaavat ja mahdollisesti muuttavat lähetettyjä tietoja. Terveydenhuoltojärjestelmissä, man-in-the-middle -hyökkäykset voivat aiheuttaa vakavia seurauksia. Hyökkääjä voisi esimerkiksi siepata ja muuttaa laboratoriotuloksia, muuttamalla normaalit arvot vakavan sairauden osoittaviksi. Tämä manipulointi voisi johtaa lääkäreitä tekemään virheellisiä diagnooseja ja mahdollisesti antamaan vaarallisia hoitoja, mikä vaarantaisi potilaan hengen. [84]

### 5.4.3 Haavoittuvuuksien hoitaminen

Säännöllisten haavoittuvuustarkastusten ja turvallisuusauditointien suorittaminen on olennaista järjestelmien heikkouksien tunnistamiseksi ennen niiden hyödyntämistä. Tunkeutumistestauksen kaltaiset tekniikat simuloivat kyberhyökkäyksiä paljastaen mahdolliset sisääntulopisteet ja arvioiden järjestelmän kestävyyttä. Rutiiniauditoinnit varmistavat tietoturvastandardien noudattamisen ja osoittavat alueet, joilla tarvitaan lisäturvatoimenpiteitä. Korjaamalla nämä haavoittuvuudet ajoissa organisaatiot voivat vähentää altistumistaan kyberuhille ja parantaa tietoturvatasoaan. [79]

Tehokas päivitysten hallinta on tärkeää järjestelmän tietoturvan ylläpitämiseksi, sillä ohjelmistopäivitykset korjaavat usein sovellusten ja käyttöjärjestelmien tunnettuja haavoittuvuuksia [84]. Jäsennelty päivitysten hallintastrategia mahdollistaa organisaatioille päivitysten nopean käyttöönoton, varmistaen, että tietoturvaheikkoudet ratkaistaan viipymättä. Tämä vähentää riskiä, että hyökkääjät hyödyntävät vanhentuneita komponentteja, ja minimoi järjestelmän käyttökatkoksia.

Käyttöoikeuksien hallinta rajoittaa käyttäjien käyttöoikeuksia ja auttaa suojaamaan arkaluonteisia tietoja rajoittamalla pääsyä käyttäjän roolin ja vastuiden perusteella. Käyttöoikeushallintaa täydentää jatkuva verkonvalvonta, joka havaitsee epäilyttävän tai luvattoman toiminnan reaaliajassa. Näiden toimintojen kirjaaminen ja analysointi tarjoavat arvokasta tietoa tietoturvaloukkauksista, mahdollistaen organisaatioille nopean reagoinnin potentiaalisiin uhkiin. Yhdessä käyttöoikeushallinta ja valvonta luovat ennakoivan lähestymistavan haavoittuvuuksien hallintaan ja vahvistavat sekä tietoturvaa että järjestelmän eheyttä. [79]

# 6 Yhteenveto

Tutkimus selvittää, miten rajapintastandardit vaikuttavat ohjelmistokehityksen eri vaiheisiin, kuten suunnitteluun, ohjelmistoarkkitehtuuriin, toteutukseen, testaukseen ja käyttöönottoon. Keskeiset havainnot osoittavat, että standardit tehostavat kehitysprosesseja tarjoamalla johdonmukaiset ohjeet, vähentämällä teknistä epäselvyyttä ja edistämällä modulaarisia sekä uudelleenkäytettäviä ratkaisuja. Toisaalta esiin nousee myös haasteita, kuten joustavuuden väheneminen, kasvava monimutkaisuus ja yhteensopivuusongelmat eri standardien välillä.

Tutkielmassa tarkastellaan toimialakohtaisia sovelluksia, jotka havainnollistavat standardien kykyä vastata erityisiin tarpeisiin. Terveydenhuollossa HL7 mahdollistaa potilastietojen tehokkaan vaihdon ja järjestelmien yhteentoimivuuden. Finanssialalla ISO 20022 modernisoi transaktioiden käsittelyä, mikä varmistaa tarkkuuden ja nopeuden globaalissa kaupankäynnissä. Koulutussektorilla standardit, kuten SCORM ja xAPI, tukevat digitaalisten oppimistyökalujen integrointia, parantaen oppimiskokemusta. Asioiden internetissä standardointi mahdollistaa monenlaisten laitteiden turvallisen tiedonsiirron ja palveluiden välisen yhteydenpidon.

Kehittäjien haastattelujen ja kyselyiden perusteella tuodaan esiin näkökulmia standardointiin. Vastaajat tunnustavat standardien hyödyt innovaation edistämisessä ja luotettavuuden ylläpitämisessä, mutta mainitsevat myös haasteita, kuten standardien hitaan kehityksen ja vanhojen järjestelmien sopeuttamisen nykyaikaisiin vaatimuksiin.

Tutkielmassa korostetaan myös standardien roolia järjestelmien luotettavuuden ja kyberturvallisuuden tukemisessa. Vaikka rajapintastandardit eivät suoraan ratkaise kyberturvallisuushaasteita, ne tarjoavat vakaan pohjan, jolle turvallisuusprotokollat voidaan rakentaa. Tutkimus käsittelee myös vanhentuneiden standardien ja heikkojen toteutusten aiheuttamia riskejä, ja painottaa ajantasaisen järjestelmän ylläpidon merkitystä.

Johtopäätöksenä työ osoittaa, että vaikka rajapintaviestintästandardit tuovat mukanaan sekä mahdollisuuksia että haasteita, niiden huolellisella soveltamisella voidaan parantaa ohjelmistojärjestelmien laatua, turvallisuutta ja toiminnallisuutta. Tulokset tarjoavat arvokasta tietoa kehittäjille, organisaatioille ja päättäjille, jotka pyrkivät tasapainottamaan standardoinnin ja innovaation kasvavassa digitaalisen yhteydenpidon maailmassa. Standardointi on teoriassa toimiva, mutta ei aina käytännössä.

TK1: Rajapintastandardit ratkaisevat yhteentoimivuus- ja tiedonsiirtohaasteita tarjoa-

malla selkeät säännöt, mikä parantaa kehitysprosessien laatua ja tehokkuutta. Ne kuitenkin voivat tuoda joustavuuden vähenemistä ja monimutkaisia integraatioita, jolloin huolellinen toteutus ja ylläpito ovat välttämättömiä.

TK2: Rajapintastandardit parantavat ohjelmistokehityksen prosesseja tarjoamalla selkeät suuntaviivat, jotka tukevat suunnittelua, modulaarista arkkitehtuuria, tehokasta toteutusta ja automatisoitua testausta. Ne nopeuttavat käyttöönottoa ja ylläpitoa, mutta voivat rajoittaa joustavuutta ja vaatia tarkkaa dokumentaatiota.

TK3: Rajapintastandardit parantavat käytettävyyttä tarjoamalla selkeät ja johdonmukaiset käytännöt, jotka vähentävät käyttäjän kognitiivista kuormitusta ja virheiden mahdollisuutta. Ne ohjaavat intuitiivisten ja käyttäjäystävällisten järjestelmien suunnittelua, tehostavat kehittäjien työtä ja varmistavat yhtenäisen käyttökokemuksen, vaikka voivat toisinaan rajoittaa luovuutta.

TK4: Rajapintastandardit parantavat järjestelmien luotettavuutta ja kyberturvallisuutta varmistamalla tiedon eheyden ja johdonmukaisuuden sekä tukemalla suojausprotokollia, kuten salauksia ja haavoittuvuuksien hallintaa. Oikein toteutettuna ne vähentävät virheitä ja kyberuhkia, mutta väärä tulkinta voi altistaa järjestelmät riskeille.

Tutkimuksessa voitaisiin syventyä tarkemmin kehittyvien teknologioiden, kuten tekoälyn ja kvanttilaskennan, vaikutuksiin rajapintastandardointiin. Näiden teknologioiden erityisvaatimukset saattavat edellyttää täysin uusia standardeja ja rajapintojen suunnittelumalleja. Tällaiset jatkotutkimukset voisivat tarjota arvokasta tietoa sekä standardointiorganisaatioille että kehittäjille, jotka pyrkivät navigoimaan nopeasti muuttuvassa teknologisessa ympäristössä.

Tekoälyä voitaisiin hyödyntää erilaisten tulkintaongelmien ratkaisemiseen. Tässä tutkimuksessa viisi kahdeksasta kyselyyn vastaajasta kokee yhteensopivuusongelmia muiden järjestelmien integraatiossa. Luottamalla määrittelyssä tekoälyyn, ihmisten väliset tulkintaerot voitaisiin minimoida. Tämä helpottaisi ja nopeuttaisi entisestään eri järjestelmien välistä integraatiota.

Tätä tutkimusta voisi parantaa toteuttamalla tarkemmin suunniteltuja ja kohdennettuja kyselyitä sekä haastatteluja. Lisäksi osallistujamäärää voitaisiin kasvattaa, mikä lisäisi tutkimuksen kattavuutta. Lisäksi voitaisiin verrata eri standardeja toisiinsa. Tämän avulla opittaisiin, mitkä standardit ovat toimivia ja mitkä aiheuttavat ongelmia.

Jos jatkaisin tutkimusta, tekisin yhteistyötä standardien luojien kanssa. Selvittäisin, mitä

vaaditaan täysin standardien mukaisen järjestelmän toteuttamiseen. Onko se edes mahdollista inhimillisten virheiden ja jatkuvien muutosten vuoksi?

# Lähteet

[1] Branden Hookway. *Interface*. The MIT Press, Cambridge, Massachusetts, 2014 - 2014.

[2] Interfaces (the java tutorials learning the java language interfaces and inheritance). `https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html`. [Sivulla käyty: 17.09.2024].

[3] Thomas A Wadlow. The xerox alto computer. *Byte Magazine*, 6(9):58–68, 1981.

[4] Tom Fellmann, Manolya Kavakli, et al. A command line interface versus a graphical user interface in coding VR systems. In *Proceedings of Second IASTED International Conference on Human Computer Interaction*, 2007.

[5] Harini Sampath, Alice Merrick, and Andrew Macvean. Accessibility of command line interfaces. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–10, 2021.

[6] Bernard J Jansen. The graphical user interface. *ACM SIGCHI Bulletin*, 30(2):22–26, 1998.

[7] Susan B Barnes. User friendly: A short history of the graphical user interface. *Sacred Heart University Review*, 16(1):4, 2010.

[8] Wendy L Martinez. Graphical user interfaces. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2):119–133, 2011.

[9] Maxime Lamothe, Yann-Gaël Guéhéneuc, and Weiyi Shang. A systematic review of API evolution literature. *ACM Computing Surveys (CSUR)*, 54(8):1–36, 2021.

[10] Adeel Ehsan, Mohammed Ahmad ME Abuhaliqa, Cagatay Catal, and Deepti Mishra. RESTful API testing methodologies: Rationale, challenges, and solution directions. *Applied Sciences*, 12(9):4369, 2022.

[11] Ole Henry Halvorsen and Douglas Clarke. *Universal Serial Bus*, pages 141–172. Apress, Berkeley, CA, 2011.

[12] Stevan Eidson, Brett Gaines, and Paul Wolf. 30.2: HDMI: High-Definition Multimedia Interface. In *SID Symposium Digest of Technical Papers*, volume 34, pages 1024–1027. Wiley Online Library, 2003.

[13] Pierre Bourhis, Juan L Reutter, Fernando Suárez, and Domagoj Vrgoč. JSON: data model, query languages and schema specification. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pages 123–135, 2017.

[14] About health level seven international | hl7 international. `https://www.hl7.org/about/index.cfm?ref=nav`. [Sivulla käyty: 17.09.2024].

[15] The early history of health level 7. `https://www.ringholm.com/docs/the_early_history_of_health_level_7_HL7.htm`. [Sivulla käyty: 13.11.2024].

[16] Hl7 standards product brief - hl7 version 2 product suite | hl7 international. `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=185`. [Sivulla käyty: 17.10.2024].

[17] Hl7 v2.3.1 introduction. `https://www.hl7.eu/HL7v2x/v231/std231/CH1.html#Heading1`. [Sivulla käyty: 17.10.2024].

[18] Michael Anywar, Mário Macedo, Santiago Pazmino, Tobias Bronsch, Benjamin Kinast, Ann-Kristin Kock-Schoppenhauer, and Björn Schreiweis. Challenges and Lessons Learned in Mapping HL7 v2 Data to openEHR: Insights from UKSH Medical Data Integration Center. In *Digital Health and Informatics Innovations for Sustainable Health Care Systems*, pages 1328–1332. IOS Press, 2024.

[19] Duane Bender and Kamran Sartipi. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In *Proceedings of the 26th IEEE international symposium on computer-based medical systems*, pages 326–331. IEEE, 2013.

[20] Frank Oemig. HL7 version 2. x goes FHIR. In *German Medical Data Sciences: Shaping Change–Creative Solutions for Innovative Medicine*, pages 93–98. IOS Press, 2019.

[21] Hl7 standards product brief - hl7 version 3 product suite | hl7 international. `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186`. [Sivulla käyty: 17.09.2024].

[22] Barry Smith and Werner Ceusters. HL7 Rim: An Incoherent Standard. *Studies in Health Technology and Informatics*, 124(Proceedings of MIE 2006):133–138, 2006.

[23] Tim Benson and Grahame Grieve. Principles of health interoperability. *Cham: Springer International*, pages 21–40, 2021.

[24] Hl7 standards product brief - fhir® (hl7 fast healthcare interoperability resources) | hl7 international. `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=491`. [Sivulla käyty: 16.10.2024].

[25] Duane Bender and Kamran Sartipi. HL7 FHIR: An Agile and RESTful approach to healthcare information exchange. In *2013 IEEE 26TH INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS (CBMS)*, pages 326–331, NEW YORK, 2013. IEEE.

[26] Hl7 standards product brief - hl7 clinical document architecture (cda®) r2.0 specification online navigation, edition 2024 | hl7 international. `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=496`. [Sivulla käyty: 11.11.2024].

[27] Catherine Chronaki, P. Lelis, C. Demou, Manolis Tsiknakis, and S.C. Orphanoudakis. An hl7/cda framework for the design and deployment of telemedicine services. volume 4, pages 3504 – 3507 vol.4, 02 2001.

[28] Kanittha Tambunlertchai. Determinants and barriers to financial inclusion in Myanmar: what determines access to financial services and what hinders it? *The Singapore Economic Review*, 63(01):9–26, 2018.

[29] Romina Bandura and Sundar R Ramanujam. *Developing inclusive digital payment systems*. JSTOR, 2022.

[30] Messaging and standards | swift. `https://www.swift.com/about-us/discover-swift/messaging-and-standards`. [Sivulla käyty: 16.10.2024].

[31] Mike Gallaher and Chad Harper. Demystifying ISO 20022: Evaluating the benefits and limitations of new messaging standards. *Journal of payments strategy systems*, 15(4):410–418, 2021.

[32] Farid Makhlouf and Refk Selmi. Do sanctions work in a crypto world? The impact of the removal of Russian Banks from SWIFT on Remittances. 2022.

[33] Quinn Magendanz. *Data Sharing and Traceability: Improving User Trust in Data Management within Open Banking and Beyond*. PhD thesis, Massachusetts Institute of Technology, 2024.

[34] Technical overview - mojaloop. `https://mojaloop.io/how-it-works/technical-overview/`. [Sivulla käyty: 16.10.2024].

[35] Marta Lokhava, Giuliano Losa, David Mazières, Graydon Hoare, Nicolas Barry, Eli Gafni, Jonathan Jove, Rafał Malinowsky, and Jed McCaleb. Fast and secure global payments with stellar. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 80–96, 2019.

[36] Jayanth Varma. Blockchain in finance. *Vikalpa*, 44:1–11, 03 2019.

[37] Jen Sheng Wang. Exploring and evaluating the development of an open application programming interface (Open API) architecture for the fintech services ecosystem. *Business Process Management Journal*, 2024.

[38] Mark Haakman, Luís Cruz, Hennie Huijgens, and Arie Van Deursen. Ai lifecycle models need to be revised: An exploratory study in fintech. *Empirical Software Engineering*, 26(5):95, 2021.

[39] Narcisa Roxana Mosteanu and Alessio Faccia. Fintech frontiers in quantum computing, fractals, and blockchain distributed ledger: Paradigm shifts and open innovation. *Journal of Open Innovation: Technology, Market, and Complexity*, 7(1):19, 2021.

[40] Robby Robson and Avron Barr. Learning technology standards-the new awakening. In *Proceedings of the Sixth Annual GIFT Users Symposium*, volume 6, pages 1–9. US Army Research Laboratory, 2018.

[41] Iraklis Varlamis and Ioannis Apostolakis. The present and future of standards for e-learning technologies. *Interdisciplinary Journal of E-Learning and Learning Objects*, 2(1):59–76, 2006.

[42] Gottfried Vossen and Peter Westerkamp. Why service-orientation could make e-learning standards obsolete. *International Journal of Technology Enhanced Learning*, 1(1-2):85–97, 2008.

[43] Tatyana Ivanova, Valentina Terzieva, and Malinka Ivanova. Intelligent technologies in e-learning: Personalization and interoperability. In *Proceedings of the 22nd International Conference on Computer Systems and Technologies*, pages 176–181, 2021.

[44] Oliver Bohl, Jörg Scheuhase, Ruth Sengler, and Udo Winand. The sharable content object reference model (SCORM)-a critical review. In *International Conference on Computers in Education, 2002. Proceedings.*, pages 950–951. IEEE, 2002.

[45] Charles Severance, Ted Hanss, and Josepth Hardin. Ims learning tools interoperability: Enabling a mash-up approach to teaching and learning tools. *Technology, Instruction, Cognition and Learning*, 7(3-4):245–262, 2010.

[46] Jonathan M Kevan and Paul R Ryan. Experience API: Flexible, decentralized and activity-centric data collection. *Technology, knowledge and learning*, 21:143–149, 2016.

[47] Vangelis Gazis. A Survey of Standards for Machine-to-Machine and the Internet of Things. *IEEE Communications Surveys & Tutorials*, 19(1):482–511, 2016.

[48] Biswajeeban Mishra and Attila Kertesz. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8:201071–201086, 2020.

[49] Dan Dinculeană and Xiaochun Cheng. Vulnerabilities and limitations of MQTT protocol used between IoT devices. *Applied Sciences*, 9(5):848, 2019.

[50] Xi Chen. Constrained application protocol for internet of things. *URL: https://www. cse. wustl. edu/˜ jain/cse574-14/ftp/coap*, 2014.

[51] Jonathan Grudin. Systematic sources of suboptimal interface design in large product development organizations. *Human-computer interaction*, 6(2):147–196, 1991.

[52] 10 usability heuristics for user interface design. `https://www.nngroup.com/articles/ten-usability-heuristics/`. [Sivulla käyty: 3.11.2024].

[53] Blackford Middleton, Meryl Bloomrosen, Mark Dente, Bill Hashmat, Ross Koppel, J. Marc Overhage, S. Rosenbloom, Charlotte Weaver, and Jiajie Zhang. Enhancing patient safety and quality of care by improving the usability of electronic health record systems: Recommendations from amia. *Journal of the American Medical Informatics Association : JAMIA*, 20, 01 2013.

[54] Jiajie Zhang and Muhammad F Walji. Turf: toward a unified framework of ehr usability. *Journal of biomedical informatics*, 44(6):1056–1067, 2011.

[55] Nigel Bevan, James Carter, and Susan Harker. ISO 9241-11 revised: What have we learnt about usability since 1998? In *Human-Computer Interaction: Design and Evaluation: 17th International Conference, HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part I 17*, pages 143–151. Springer, 2015.

[56] Nigel Bevan, Jim Carter, Jonathan Earthy, Thomas Geis, and Susan Harker. New ISO Standards for Usability, Usability Reports and Usability Measures. volume 9731, pages 268–278, 07 2016.

[57] Daniel Fett, Ralf Küsters, and Guido Schmitz. A comprehensive formal security analysis of OAuth 2.0. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1204–1215, 2016.

[58] Joel Weise. Public key infrastructure overview. *Sun BluePrints OnLine, August*, pages 1–27, 2001.

[59] Sandeep Mathur and Ankita Arora. Internet of things (IoT) and PKI-based security architecture. In *Industrial internet of things and cyber-physical systems: transforming the conventional to digital*, pages 25–46. IGI Global, 2020.

[60] Douglas Selent. Advanced encryption standard. *Rivier Academic Journal*, 6(2):1–14, 2010.

[61] Yasir Javed, Adnan Shahid Khan, Abdul Qahar, and Johari Abdullah. Preventing DoS attacks in IoT using AES. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(3-11):55–60, 2017.

[62] Heli Hyvättinen. Interface standards and creating innovation markets—implications on SMEs in a technology programme. *Technovation*, 26(2):262–273, 2006.

[63] Mikel D Petty, Jungyoon Kim, Salvador E Barbosa, and Jai-Jeong Pyun. Software frameworks for model composition. *Modelling and Simulation in Engineering*, 2014(1):492737, 2014.

[64] Jodie LM Anderson. A basic understanding of the impact of interoperability on system's architectures. In *IIE Annual Conference. Proceedings*, page 1672. Institute of Industrial and Systems Engineers (IISE), 2008.

[65] Thomas Rings, Patrick Poglitsch, Stephan Schulz, Luca Serazio, and Theofanis Vassiliou-Gioles. A generic interoperability testing framework and a systematic development process for automated interoperability testing. *International Journal on Software Tools for Technology Transfer*, 16:295–313, 2014.

[66] Tuncay Namli, Gunes Aluc, and Asuman Dogac. An interoperability test framework for HL7-based systems. *IEEE Transactions on Information Technology in Biomedicine*, 13(3):389–399, 2009.

[67] Konstantinos Fysarakis, Ioannis Askoxylakis, Othonas Soultatos, Ioannis Papaefstathiou, Charalampos Manifavas, and Vasilios Katos. Which IoT protocol. In *Comparing standardized approaches over a common M2M application. IEEE Global Communications Conference (GLOBECOM), Washington DC, USA*, 2016.

[68] Philip Scott and Robert Worden. Semantic mapping to simplify deployment of HL7 v3 Clinical Document Architecture. *Journal of biomedical informatics*, 45(4):697–702, 2012.

[69] Petar Rajković, Dejan Aleksić, Andjelija Djordjević, and Dragan Janković. Hybrid software deployment strategy for complex industrial systems. *Electronics*, 11(14):2186, 2022.

[70] Leszek Jaskierny and Leszek Kotulski. A self-adapting iot network configuration supported by distributed graph transformations. *Applied Sciences*, 13(23):12718, 2023.

[71] Teun Hendriks and Kees Wevers. Abstraction conflicts in industrial deployment of model-based interoperability standards. In *Proc. Conference on Systems Engineering Research (CSER)*. Citeseer, 2007.

[72] Rebecca A Meehan, Donald T Mon, Kandace M Kelly, Mitra Rocca, Gary Dickinson, John Ritter, and Constance M Johnson. Increasing EHR system usability through standards: Conformance criteria in the HL7 EHR-system functional model. *Journal of biomedical informatics*, 63:169–173, 2016.

[73] Fabian Fagerholm and Jürgen Münch. Developer experience: Concept and definition. In *2012 international conference on software and system process (ICSSP)*, pages 73–77. IEEE, 2012.

[74] Ismael Caballero, Óscar Gómez, and Mario Piattini. Getting better information quality by assessing and improving information quality management. *ICIQ*, 2004:9th, 2004.

[75] Hari Prasad Josyula. The role of the ISO 20022 messaging standard in improving payment transactions utilising participants' data. *Journal of Payments Strategy & Systems*, 18(2):159–166, 2024.

[76] Joseph Bugajski and Philippe De Smedt. Assuring data interoperability through the use of formal models of visa payment messages. In *ICIQ*, pages 29–37, 2007.

[77] John D'Amore, Omar Bouhaddou, Sandra Mitchell, Chun Li, Russell Leftwich, Todd Turner, Matthew Rahn, Margaret Donahue, and Jonathan Nebeker. Interoperability progress and remaining data quality barriers of certified health information technologies. In *AMIA Annual Symposium Proceedings*, volume 2018, page 358. American Medical Informatics Association, 2018.

[78] Jing Xie, Heather Richter Lipford, and Bill Chu. Why do programmers make security errors? In *2011 IEEE symposium on visual languages and human-centric computing (VL/HCC)*, pages 161–164. IEEE, 2011.

[79] Milena Krumova and Ashish Kataria. Education cybersecurity: Learning management system, data and tools. In *Proceedings of the 16th International Conference on Theory and Practice of Electronic Governance*, pages 318–323, 2023.

[80] Ryan DRAKE, Evan RIDDER, et al. Healthcare cybersecurity vulnerabilities. In *International Conference on Cybersecurity and Cybercrime*, volume 9, pages 49–56, 2022.

[81] NOAH SUNDER RAJ and G MANOJ KUMAR. Secure Messaging–Cryptography And Middleware Queuing Using ISO 20022 Messaging Standards.

[82] Rishi Saripalle, Christopher Runyan, and Mitchell Russell. Using HL7 FHIR to achieve interoperability in patient health record. *Journal of biomedical informatics*, 94:103188, 2019.

[83] Euijong Lee, Young-Duk Seo, Se-Ra Oh, and Young-Gab Kim. A Survey on Standards for Interoperability and Security in the Internet of Things. *IEEE Communications Surveys & Tutorials*, 23(2):1020–1047, 2021.

[84] Christian Dameff, Maxwell Troy Bland, Kirill Levchenko, and Jeffrey Tully. Pestilential Protocol : How Unsecure HL7 Messages Threaten Patient Lives.

# Liite 1: Kysely ja vastaukset

| **Q1: How long have you been working in the IT field?** | **Vastausten määrä** |
|---|---|
| 0–2 years | 0 |
| 3–5 years | 1 |
| 6–10 years | 0 |
| 11+ years | 7 |

| **Q2: What standard you are working or have worked with? (multiple answers allowed)** | **Vastausten määrä** |
|---|---|
| HL7 | 6 |
| ISO 20022 | 2 |
| SWIFT | 1 |
| SCORM | 1 |
| LTI | 0 |
| MQTT | 1 |
| CoAP | 0 |
| Muu | 1 |

| **Q3: Rate your experience level in working with standards. (1 = Beginner, 5 = Expert** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 2 |
| 3 | 2 |
| 4 | 3 |
| 5 | 1 |

| **Q4: How do you feel about standardization in the field of software development? (1 = Hate it, 5 = Love it)** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |
| 4 | 4 |
| 5 | 3 |

| **Q5: Did following a standard affect your productivity? (1 = Worsened significantly, 5 = Greatly improved)?** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 3 |
| 4 | 1 |
| 5 | 4 |

| **Q6: How much do you believe interface standards affect software quality and reliability? (1 = Worsened significantly, 5 = Greatly improved)** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 2 |
| 5 | 5 |

| **Q7: Does standardized interfaces affect development time in your experience? (1 = Worsened significantly, 5 = Greatly improved)?** | **Vastausten määrä** |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |
| 4 | 4 |
| 5 | 3 |

| **Q8: Within your system do you need to take other standards into consideration?** | **Vastausten määrä** |
|---|---|
| Yes | 6 |
| No | 2 |
| Maybe | 0 |

| **Q9: Do you encounter interoperability issues with other systems?** | **Vastausten määrä** |
|---|---|
| Yes | 5 |
| No | 3 |
| Maybe | 0 |

| **Q10: Do you use authentication mechanisms for message integrity?** | **Vastausten määrä** |
|---|---|
| Yes | 3 |
| No | 3 |
| Maybe | 2 |

| **Q11: Does standardization make testing easier?** | **Vastausten määrä** |
|---|---|
| Yes | 7 |
| No | 0 |
| Maybe | 1 |

| **Q12: Does standardization make code more difficult to maintain?** | **Vastausten määrä** |
|---|---|
| Yes | 0 |
| No | 6 |
| Maybe | 2 |

| **Q13: Does standardization affect usability and UX design?** | **Vastausten määrä** |
|---|---|
| Yes | 4 |
| No | 2 |
| Maybe | 2 |

| **Q14: Does standardization make for better UX?** | **Vastausten määrä** |
|---|---|
| Yes | 6 |
| No | 2 |
| Maybe | 0 |

| **Q15: Does standardization make UX design easier?** | **Vastausten määrä** |
|---|---|
| Yes | 4 |
| No | 0 |
| Maybe | 4 |

| **Q16: How do you ensure good UX?** | **Vastausten määrä** |
|---|---|
| With the same, or another standard | 2 |
| General guidelines and heuristics | 3 |
| Good designers | 4 |
| Other (Measuring success) | 1 |

| **Q17: How do you feel about the impact of standardization on innovation in software development?** |
|---|
| "In general, this will certainly increase innovation, especially for smaller systems specialized in one specific area." |
| "Impact on a single project is not so huge, but when there are many, then it makes sense to create a library that any project can use." |
| "I think it's very good, it makes it easier to migrate and run products on different vendors." |
| "From a positive perspective, standards can promote interoperability, improve quality, and accelerate development, as developers can leverage common patterns and interfaces instead of reinventing the wheel. This can free up time and resources to focus on new ideas and development directions. Excessive standardization, however, can stifle creativity and flexibility. If developers are forced to follow overly strict rules, it prevents them from experimenting with new approaches or using creative solutions to problems. Overall, standardization creates a more stable foundation for development, but innovation also requires room for flexibility and creative thinking." |
| "Ease development and increase innovation." |
| "Makes it easier." |
| "Improves integration to existing systems." |

| **Q18: Does future updates to interface standards affect the planning and design phases or is it typically overlooked?** |
|---|
| "Depends on the budget and time allocation. Some parts that are not prioritized in our system will be left out." |
| "It could be, but so far for SCORM I think still SCORM 1.2 is very popular." |
| "If standards are updated, it would be wise to consider their impact and incorporate them into the planning and design phases. This is one of the points where "technical debt"can accumulate if not properly addressed. However, it's also important not to over-anticipate changes, as this can lead to excessive adjustments at the end of the project. A third point to consider is the slow pace of standard development at the national level. Often, companies have already made decisions (standards are not all-powerful), and the development of national standards tends to lag behind, meaning that the solutions in place may not align perfectly." |
| "Affect when needed." |
| "It affects, not overlooked." |
| "Possible updates are good to keep in mind while developing software." |

| **Q19: What challenges or limitations have you encountered due to the standardization of interfaces?** |
|---|
| "Within our system, we primarily use our internal interfaces, which are not HL7 standard. The HL7 standard is complex and takes time to implement." |
| "Missing features." |
| "I think modern ways of collecting data and using them are challenging for standardization." |
| "The loss of flexibility and the difficulty in implementing desired or necessary changes, such as updates to national standards. Incompatibility between different standards (country-specific variations even within the same standard) can increase integration challenges. It's important to remember that the standard a supplier uses may also be a tool to lock in a user base within a chosen business ecosystem. Standards aren't governed by a dictator but by communities, which can make decision-making processes more difficult. The sales pitch for standards is usually 50% true, 40% dreams, and 10% total nonsense. On top of that, every country sneaks in what they consider important, further confusing the whole package. Sometimes, semantics break down accidentally, leading to the exchange of information that isn't what the other party expects. Standards are slow, and as a result, innovation suffers." |
| "Acceptance of change and difference in opinions." |
| "No challenges." |
| "Although developing with standards, it is needed to make sure everyone has the same understanding of the standards." |

| **Q20: In your opinion, how could the current standards in your field be improved to better meet the evolving needs of developers and the industry?** |
|---|
| "I don't have enough experience to answer this question." |
| "Keep a good balance between universality and flexibility and make good use of coming new technologies." |
| "In integrations, there should at least be a dictator/tsar, who is a visionary bringing the necessary flexibility. In a standard, however, I wouldn't focus on an individual but rather enforce a community-based solution, even though that may slow down the development of the standard. On the other hand, the development of standards should be completely removed from publicly funded organizations and shifted entirely to the responsibility of suppliers. This would ensure a more logical development process, without the kind of nonsensical ideas that we are still fixing in the 2020s." |
| "Better standards, for example authentication and authorization between clouds, systems, apps, and users." |
| "They should more clearly say what to show to end users." |