Jere Ojala

Kuorien käyttö yrityksissä

Tietotekniikan pro gradu -tutkielma

12. marraskuuta 2021

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Jere Ojala

Yhteystiedot: jejoojal@student.jyu.fi

Ohjaajat: Ville Tirronen, Jonne Itkonen ja Jorma Kyppö

Työn nimi: Kuorien käyttö yrityksissä

Title in English: Usage of shells in companies

Työ: Pro gradu -tutkielma

Opintosuunta: Tietotekniikka

Sivumäärä: 64+5

Tiivistelmä: Kuoret ovat yksi vanhimmista ja olennaisimmista tietotekniikan innovaatioista, jotka ovat aktiivisessa käytössä nykypäivänäkin. Kuorissa on kuitenkin useita iän tuomia ongelmia, jotka tulevat esiin varsinkin ympärillä kehittyvän teknologian myötä. Tässä tutkielmassa tavoitteena on kartoittaa syitä, mikä saa IT-alan ihmiset käyttämään kuorta varsinkin yritysten sisällä. Samalla katseltiin vaihtoehtoisia innovaatioita kuoren tilalle. Tutkimus suoritettiin kyselytutkimuksena, jossa alan ihmiset saivat vastata keskeisiin kysymyksiin avoimesti. Vastaukset analysoitiin sisällönanalyysilla. Vastauksissa ilmaantuneet maininnat laskettiin ja eniten esiintyneiden mainintojen pohjalta tehtiin päätelmiä verraten niitä kuoren teoriaan.

Avainsanat: kuori, bash, kysely, sisällönanalyysi, teknologian adaptaatio, TAM

Abstract: Shells are one of the oldest and most essential IT innovations that are still in active use today. However, the shells are starting to show their old age and several problems have become apparent with the technology evolving around them. The aim of this study is to map the reasons that make people in the industry use shells, especially within the IT industry. At the same time, innovative alternatives to the shell were looked at. The study was conducted as a questionnaire in which people of the IT field were allowed to answer primary questions of the survey openly. The answers were analyzed by content analysis. The mentions that appeared in the answers were counted and conclusions were drawn from the most common

mentions and compared to surrounding shell theory.

Keywords: bash, shell, survey, content analysis, technology adaptation, TAM

Jyväskylässä 12. marraskuuta 2021

Termiluettelo

Turing-vahvuus	Turing-vahvuus on ominaisuus tietojenkäsittelyjärjestelmille, jotka pystyvät laskemaan minkä tahansa Turing-laskettavan funktion tai pystyvät simuloimaan Turingin koneen.
Turing-laskettava funktio	Funktio on Turing-laskettava funktio, jos sille on olemassa Turing-kone, joka osaa laskea sen.
Turingin kone	Teoreettinen malli tietokoneelle ja sen toiminnalle.
Idempotenssi	Idempotentin metodin vaikutus on sama suoritettiin se kerran tai montaa kertaa.

Kiitokset

Aluksi haluisin kiittää Olli Vänskää Mikrobitiltä artikkelin julkaisemisesta ja sen kautta laajemman aineiston keräämisestä. Artikkelista oli paljon apua gradun aineiston laadun parantamisessa. Kiitokset myös kaikille kyselyyn vastanneista.

Lisäksi kiitokset Ilona Nurmiselle ja Jeri Varjosalolle kaikesta avusta tutkielmaa tehdessä.

Kuviot

Kuvio 1. PaSh (Vasilakis ym. 2021) 9
Kuvio 2. POSH (Raghavan ym. 2020) 9
Kuvio 3. Innovaation käyttöönottajat 17
Kuvio 4. TAM-kaavio 20
Kuvio 5. TAM2-kaavio 21
Kuvio 6. TAM3-kaavio 22
Kuvio 7. Ulkoisten muuttujien välisten suhteiden merkittävyydet (Lee, Kozar ja Larsen 2003) 23
Kuvio 8. Tehtävien selvitys 25

Taulukot

Taulukko 1. Vastaajien vastaamat roolit 29
Taulukko 2. Muut roolit 30
Taulukko 3. Vastaajien vastaamat kuoret 31
Taulukko 4. Vastaajien itsearvioidut kuoren käytön osaamiset 31
Taulukko 5. Haluaisitko käyttää kuoriohjelmointia tehtäviin, joihin et tällä hetkellä käytä sitä? 34
Taulukko 6. Käyttäisitkö kuoriskriptien sijasta jotain muuta kieltä tai teknologiaa? 35
Taulukko 7. Vastaajien ehdottamat vastineet kuoriohjelmoinnille 36
Taulukko 8. Haluaisitko käyttää interaktiivista kuorta tehtäviin, joihin et tällä hetkellä käytä kuorta? 39
Taulukko 9. Käyttäisitkö interaktiivisen kuoren sijasta edellisen kohdan tehtäviin jotain muuta kuin interaktiivista kuorta? 40
Taulukko 10. Helppous 41
Taulukko 11. Nopeus 41
Taulukko 12. Siirrettävyys 41
Taulukko 13. Automaatio 41

Sisällys

1 JOHDANTO 1

2 KUORET 2
2.1 Kuoret 2
2.1.1 Historia 2
2.1.2 Kuoriohjelmointi 3
2.1.3 AWK & sed & grep 3
2.1.4 Bash 4
2.1.5 PowerShell 5
2.1.6 Kuoren suosio 5
2.1.7 Kuoren vahvuudet, heikkoudet ja tulevaisuuden arvio 6
2.1.8 Kuoren vaarat 11
2.2 Muut mahdolliset vaihtoehtoiset teknologiat kuoren sijaan 12
2.2.1 Python 12
2.2.2 Perl 13
2.2.3 Ansible 14

3 TEKNOLOGIAN ADAPTAATIO 16
3.1 Innovaation leviäminen 16
3.2 Technology acceptance model 19

4 TUTKIMUKSEN TOTEUTUS 24
4.1 Kyselytutkimus 24
4.2 Pilotointi 25
4.3 Kyselyn jakelu 26
4.4 Sisällönanalyysi 26

5 TULOKSET 28

6 TULOSTEN POHDINTA 43
6.1 Kuoren tehtävät 43
6.2 Sisäiset syyt 43
6.3 Ulkoiset syyt 46
6.4 Haittapuolet 47
6.5 Adaptaatio 48
6.6 Mahdolliset ristiriidat 48
6.7 Vaihtoehtoiset työkalut 49
6.8 Tulevaisuuden arviointi 50

7 YHTEENVETO 51

LÄHTEET 53

LIITTEET 57

A Kyselylomake 57

1 Johdanto

Kuoret (engl. shell) ovat yksi vanhimmista tietotekniikan alan innovaatioista, jonka juuret yltävät jo 1960-luvulta nykypäivään asti, eikä sen suosio näytä hiipuvan lähitulevaisuudessa. Kuorella on kuitenkin epäkäytännöllisiä piirteitä, ja sen toiminnoille näyttäisi olevan vaihtoehtoisia toteutuksia. Kuoret eivät myöskään vaikuta olevan akateemisen ja teollisen maailman suosiossa, ja ne ovat jääneet ilman näiden tahojen tukea. Näistä seikoista huolimatta kuoret ovat suosittuja työkaluja, joiden käyttöä harjoitetaan päivittäin niin töissä kuin harrastusmielessä. Tässä tutkimuksessa on tarkoitus kartoittaa yksilötasolla syitä, *miksi alalla ja varsinkin sen yrityksissä käytetään yhä kuorta ja kuoriohjelmointia työkaluna*. Lisäksi tarkastellaan niin uusia kuin vanhempiakin vaihtoehtoisia kieliä ja teknologioita sekä niiden hyötyjä ja haittoja.

Tutkimuksen ensimmäisessä osiossa katsotaan teoriaa;aluksi itse kuoreen liittyvää teoriaa ja sitten teknologian adaptaation käsitettä. Kuoriosiossa eli toisessa luvussa alustetaan, mikä kuori on, mitä olennaisia komponentteja siihen kuuluu ja millainen historia ja nykytilanne kuorella on. Lisäksi tarkastellaan vaihtoehtoja kuorelle.

katsotaan, mitä teknologian adaptaatiolla tarkoitetaan ja käydään läpi siihen olennaisesti liittyvää teoriaa. Lisäksi kartoitetaan teknologian adaptaation mallintamiseen luotua yleistä TAM-mallia (Technology Adaptation Model) ja sen eri versioita.

Luvussa neljä käydään läpi tutkimuksen toteutuksen suunnitelma ja sen jälkeen itse toteutus. Tutkimus toteutettiin kyselytutkimuksena, joka sisälsi pääsääntöisesti avoimia kysymyksiä. Kysely jaettiin eri nettisivujen foorumeille, ja vapaaehtoiset vastaajat saivat vastata siihen. Vastaukset analysoitiin perinteisellä sisällönanalyysillä, jonka käsite myös määritellään sen omassa alaluvussa.

Luvussa viisi käydään kerätty ja analysoitu aineisto läpi kysymys kysymykseltä ja raportoidaan sisällönanalyysissä esiintyneet huomiot ja niiden esiintyneisyydet. Luvussa 6 esitetään pohdintoja luvun 5 havaintoihin ja teoriaosioihin verraten.

2 Kuoret

2.1 Kuoret

Kuori (engl. shell) on termi, joka viittaa käyttäjän ja järjestelmän väliseen rajapintaan, jolla on tarkoitus antaa käyttäjän kommunikoida tekstisyötteillä tietokoneen käyttöjärjestelmän ja sen toimintojen kanssa. Kuoret ovat yksittäisiä ohjelmia tietokoneella. Kuoria on olemassa useita, ja käyttäjä voi valita mieleisensä omaan käyttöönsä.

Kuoret kuuluvat nykyäänkin tärkeänä osana IT-alaan. Niitä käytetään pääsääntöisesti järjestelmien organisointiin, dataprosessointiin ja automatisointitehtäviin. Kuoret löytyvät jokaisesta Unix-järjestelmästä, joten ne ovat usein ainoa tai parhaiten saatavilla oleva vaihtoehto mainittuihin tehtäviin. Näin ollen kuoret pitävät erityisasemansa. Jopa useimmat pilvipalveluiden sovellukset vaativat kuorien käyttöä, kuten yleisesti käytetty Docker-konttipalvelu. (Greenberg, Kallas ja Vasilakis 2021.)

Unix-kuoret noudattavat POSIX-standardia. POSIX, eli Portable Operating System Interface, on kokonaisuus IEEE:n määrittelemiä standardeja, jotka tukevat ohjelmistojen siirrettävyyttä. POSIX sisältää yli 20 erilaista standardia ja asiakirjaa. Kuoret määritellään POSIX.2:ssa, joka määrittelee siirrettävän kuoriohjelmoinnin, sekä siirrettävän sovelluskehityksen ja käyttäjäympäristön. (Walli 1995.)

2.1.1 Historia

Ensimmäisenä kuorena voidaan pitää Louis Pouzinin kehittämää RUNCOM-kuorta, joka kehitettiin vuosina 1963-1964 (“The Origin of the Shell”, n.d.). Kuori-termi tulee myös suoraan Pouzinilta, sillä hän koki kyseisen teknologian toimivan tietokoneen sisusten kuorena (“The Internet’s fifth man”, n.d.). Pouzin näki, että tavalliset käskyt CTSS-osituskäyttöjärjestelmälle toimivat isompien käskyjen osasina.

Myöhemmin CTSS:n RUNCOM-komento antoi vaikutteita Multics-osituskäyttöjärjestelmään luodulle kuorelle, jonka suunnitteluun Pouzin osittain osallistui. Samalla Pouzin antoi tälle suunnitteluperiaatteelle nimeksi kuoren. Multicsin kuori toimi lopulta inspiraationa Unix-

järjestelmiä varten luodulle kuorelle. Unix-järjestelmä oli myös ensimmäinen järjestelmä, jossa komentotulkki ei ollut osana itse järjestelmää. Ensimmäinen kehitetty Unix-kuori oli Thompson-kuori[1].

2.1.2 Kuoriohjelmointi

Kuoriskripti (engl. shell script) on joukko peräkkäin kirjoitettuja komentoja, jotka voidaan ajaa kuorista pienehkönä ohjelmana. Osa kuorista on tarpeeksi edistyneitä toimimaan komento- ja ohjelmointikielinä. Kuten muissa ohjelmointikielissä, kuoriskripteissä komennot kirjoitetaan tiedostoon, joka sitten suoritetaan kuoren sisältä suorituskomennoilla. Kuorista löytyy paljon ohjelmoinnin perusominaisuuksia, kuten muuttujat, ehtolauseet ja silmukat[2].

Tietotekniikan piireissä on kiistelty, mitä voidaan kutsua ohjelmointikieleksi ja mitä ei. Kuoriohjelmointi erotellaan usein muista korkeamman tason ohjelmointikielistä, pidettiin sitä sitten ohjelmointikielenä tai ei. (Ray ym. 2014.)

Yksi tärkeimmistä kuoriohjelmoinnin rooleista alan sisällä on tehtävien automatisointi. Tehokkaimmillaan kuoren käsin tehtävät operaatiot, kuten palvelimien määritykset voidaan automatisoida niin, että manuaalisesti puolesta tunnista tuntiin kestävä operaatio voidaan suorittaa automatisoidusti puolessa minuutissa. (Alam, Faizan ja Shaik 2017.)

2.1.3 AWK & sed & grep

Kuoresta käsin pystytään käyttämään ulkoisia työkaluja, jotka kuuluvat osaksi Unix-järjestelmiä. AWK ja sed ovat yleisimpiä työkalut ja niitä käytetään paljon. AWK on Unixin mukana tuleva sovellusaluekielinen ohjelma, joka suodattaa sille annetun tekstisyötteen ja palauttaa käsitellyn tekstin tulosteena. AWK:n koodi hyödyntää säännöllisiä lausekkeita. Aluksi käsiteltävä syöte tarkastellaan säännöllistä lauseketta mukaillen ja sen jälkeen toiminto suoritetaan, jos syöte vastaa säännöllistä lauseketta. (Dougherty ja Robbins 1997.)

Sed, eli *stream editor*, on Unixin mukana tuleva tekstinkäsittelyyn tarkoitettu työkalu, joka käyttää omaa yksinkertaista kieltään. Sed on ei-vuorovaikutteinen ja käsittelee tekstin

[1] http://www.softpanorama.org/People/Shell_giants/introduction.shtml

[2] https://www.macs.hw.ac.uk/h̃wloidl/Courses/LinuxIntro/x945.html

tiedostosta syötevirtana ja palauttaa sen tulostevirtana kohdetiedostoon. AWK:n tavoin sed ymmärtää säännöllisiä lausekkeita. (Dougherty ja Robbins 1997.)

AWK:n ja sedin lisäksi yksi hyödyllisimmistä ja käytetyimmistä kuoren työkaluista on grep. Grep on Unix-komentorivin työkalu, jolla etsitään annetusta tekstistä merkkijonoja, jotka täsmäävät haettuun säännölliseen lausekkeeseen. (Tiloca ja Dini 2016.)

Grep, AWK ja sed kaikki ovat määriteltynä POSIXissa[3].

2.1.4 Bash

Yksi yleisimmistä kuorista on *Bourne Again Shell*, eli yleisemmältä nimeltään Bash. Bash:in tarkoitus oli olla laajennettu versio Bourne-kuoresta, joka oli bash:ia edeltävä kuori Unix-järjestelmille. Se syrjäytti rajoittuneeksi koetun oletus sh-kuoren Unix-järjestelmille ("The A-Z of Programming Languages: Bourne shell, or sh", n.d.). Bash julkaistiin ensimmäistä kertaa vuonna 1989, jonka jälkeen siitä on tullut useiden Linux-järjestelmien vakiokomentokuori. Stack Overflow'ssa kysymyksiä Bash-avainsanalla löytyy noin 139,833. Pelkällä Shell -avainsanalla etsittynä löytyi vain 84,931 ja toisella suositun kuoren avainsanalla PowerShell löytyi n. 97,406[4].

Bash on komentokuoren lisäksi komentokieli, jota voidaan käyttää myös ohjelmointikielenä. Bash on myös Turing-vahva sekä sisältää paljon ominaisuuksia, jotka edesauttavat Bash:in käyttöä ohjelmoinnissa. Bash:in syntaksi on laajennettu versio Bourne Shellin syntaksista ja se sisältää monia ominaisuuksia, jotka on otettu monista muista kuorista. Esimerkiksi Bash on ottanut C-kuorelta käyttöön aaltosulkeilla vaihtoehtoisten yhdistelmien joukon luomisen, jota alkuperäisessä Bourne-kuoressa ei ollut. Bourneen verrattuna myös Bashin saa asetuksilla mukautettua POSIX-määritelmien kanssa yhteensopivaksi[5].

[3] https://pubs.opengroup.org/onlinepubs/9699919799/

[4] https://www.stackoverflow.com (tarkistettu 3.11.2021)

[5] https://www.gnu.org/software/bash/manual/html_node/index.html

2.1.5 PowerShell

Kaikki nykykuoret eivät ole kuitenkaan Bashin lailla Unix-kuoria. PowerShell on vuonna 2006 julkaistu kuori Windows-käyttöjärjestelmille. Se ei ole unix-kuori eikä mukaudu POSIX:iin. PowerShell tulee Windows-käyttöjärjestelmien mukana ja on suhteellisen käytetty kuori. Kuten aiemmin mainittiin, Stackoverflow'ssa PowerShell-hakusanalla löytyy noin 97,406 kysymystä[6].

PowerShellin kehityksen tarkoitus oli saada Windows-käyttöjärjestelmälle mahdollisuus käyttää Unix-työkaluja, koska Windows-käyttöjärjestelmä ei ole oletukselta POSIXia tukeva järjestelmä ja niiden välillä on paljon ristiriitaisia eroja: esimerkiksi Unix-kuorille pystyy syöttämään ainoastaan numeroita merkkijonomuodossa. PowerShellin syntaksi perustuu osittain Korn-kuoreen, jonka syntaksi taas puolestaan perustuu osittain Bourne-kuoreen. PowerShellillä pystyy myös kuoriohjelmoimaan, ja sen ohjelmointi on ottanut vaikutteita Python- ja Perl-ohjelmointikielistä. (Payette 2007.)

2.1.6 Kuoren suosio

Ohjelmointikielten suosion määrittelyyn käytettävän menetelmän arviointi on hankalaa. Arviointitapoja on useita. Näihin kuuluu muun muassa se, kuinka usein kielen nimi esiintyi hakukoneissa, kuinka monessa työpaikkailmoituksessa kieli on mainittu, kuinka monta kieleen liittyvää kirjaa on myyty tai kuinka monta Github- tai muuta kehitysalustaprojektia on käyttänyt kyseistä kieltä. Lisäksi suosion arviointiin on luotu muutama siihen tarkoitettuja luetteloita, kuten esimerkiksi *TIOBE Programming Community Index*[7] ja *RedMonk Programming Language Rankings*[8].

Suositun ohjelmointiin suuntautuneen sivuston Stack Overflow'n 2019 vuoden käyttäjäkyselyssä 36,6 % kaikista vastaajista käytti eniten `Bash/Shell`iä, kuten myös 37,9 % ammattilaiskehittäjistä[9]. `Bash/Shell` oli kuudenneksi käytetyin kieli sekä kaikkien käyttäjien keskuudessa että ammattikehittäjien kesken. Käytetyin kieli kyselyssä oli `JavaScript` ja

[6] https://www.stackoverflow.com (tarkistettu 3.11.2021)

[7] https://www.tiobe.com/tiobe-index/

[8] https://redmonk.com

[9] https://insights.stackoverflow.com/survey/2019

toiseksi käytetyin oli `HTML/CSS`. Vuoden 2020 tilastoihin[10] ei ollut eriteltynä ammattikehittäjiä kaikista vastaajista, mutta ohjelmointikielistä tykätyimmät, inhotuimmat ja halutuimmat oli luokiteltu. Tykätyimpien ja inhotuimpien tilastot määriteltiin sen mukaan, miten kyseistä kieltä käyttävä koki kyseisen kielen. Halutuimpien tilasto taas muodostettiin henkilöistä, jotka eivät olleet vielä käyttäneet kieltä, mutta ilmaisivat haluavansa käyttää sitä. Tykätyimpien tilastoissa Shell/Bash/PowerShell oli saanut käyttäjien kesken 53,7 % kannatusta. Sama kategoria oli vastaavasti saanut inhotuimpien tilastossa 46,3 %. Halutuimpien tilastossa kuoret olivat saaneet vain 3,9 % kannatusta. Suositulla *Github*[11] -nimisellä versionhallintasivustalla kuorilla tehtyjen projektien määrä on 148,512. *RedMonk Programming Language Rankings*[12] -nimisessä ohjelmointikielten suosiota arvioivassa luettelossa vuoden 2021 listauksessa kuoret olivat päässeet sijalle 14. RedMonkin analyysi perustuu muun muassa StackOverflow'hun ja Githubiin. Toisaalta taas Google Trendistä hakuja mittaavan PYPL:n arvioinnissa kuoret eivät olleet päässeet sen listaamaan 28:n parhaan ohjelmointikielen joukkoon. Myös hakujen mukaan luokittelevassa TIOBE:n luettelossakaan kuoret eivät näkyneet edes 50:n parhaan joukossa.

2.1.7 Kuoren vahvuudet, heikkoudet ja tulevaisuuden arvio

Suhtautumista kuoreen on pidetty vaihtelevana alalla. Osa kokee sen erittäin tehokkaana ja hyödyllisenä, osa taas inhoaa sitä vikojensa takia. Vaikka kuoret ovat yhä aktiivisessa käytössä, niitä on kuitenkin vahvasti kritisoitu akateemisessa maailmassa sekä ammattialalla. Vuonna 2021 julkaistussa Greenbergin artikkelissa *Unix Shell Programming: The Next 50 Years* (Greenberg, Kallas ja Vasilakis 2021) väitetään kuorista eroon pääsemisen halun ammattialalla ja akateemisella puolella johtuvan kuoren kolmesta piirteestä:

- Kuoren kielellä pystyy toteuttamaan mielivaltaisia komentoja mielivaltaisilla kielillä, joissa on mielivaltaisia ominaisuuksia. Mikä tahansa komento voi kääntyä execve:ksi, joka voi suorittaa mitä tahansa tiedoston, joka toimii odottamattomasti. Koska semantiikka vaihtelee eri lähdekielten välillä, yhtenäinen analyysi on hankalaa.

[10] https://insights.stackoverflow.com/survey/2020

[11] https://www.github.com (tarkistettu 11.11.2021)

[12] https://redmonk.com/sogrady/2021/03/01/language-rankings-1-21/

- Kuoren suoritukseen liittyy monia dynaamisia komponentteja, kuten nykyinen tiedostosijainti, ympäristömuuttujat ja tiedostojärjestelmän tila. Dynaamisuuden ongelmana on se, että se estää kuorta ajamasta staattisia analyyseja tai staattisia muunnoksia, jotka voisivat joko tarkastaa komentojen oikeellisuuden tai parantaa niiden suorituskykyä.
- Kuoren semantiikkaa sotkee POSIX-kuoren spesifikaatio, joka on määritelty monisivuisessa ja hankalasti luettavassa dokumentaatiossa. Jotta skriptien toimintaa voisi perustella, täytyy ymmärtää operaattorien tarkka käyttäytyminen, ympäristön rooli sekä kuoren tulkin tila. Samassa laitteessa saattaa olla myös enemmän kuin yksi kuori erinäisiä sovellusalueita varten, ja ne laajentavat POSIXia omilla tavoillaan. Helposti ymmärrettävissä olevan semantiikan ja suoraviivaisuuden puutteen ohella tutkimuksesta tulee hankalaa.

Greenberg myös argumentoi, että kaikki nämä ominaisuudet ovat lopulta kaksiteräisiä miekkoja, eivätkä kuoret olisi niin tehokkaita, jos niitä ei olisi (Greenberg, Kallas ja Vasilakis 2021, Greenberg 2018). Tutkimus myös luokitteli kuoren ongelmia, jotka toisin kuin edellä mainitut ongelmat eivät ole "pakollisia pahoja", vaan kuoren kehityspotentiaalia tukahduttavia tekijöitä:

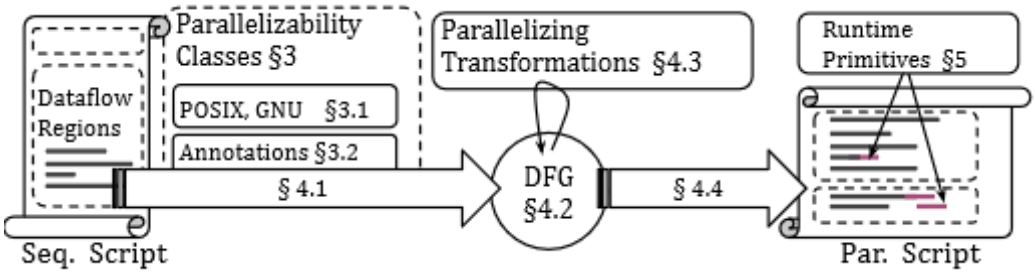
- Virhealttius: Kuorista puuttuu muiden ohjelmointikielten kaltainen suojamekanismi, joka estää vakavia virheitä muodostumasta. Kuorilla on pääsy itse järjestelmään, jossa yhdellä kirjoitusvirheellä pystytään tekemään suurta vahinkoa.
- Skaalautumaton suoritettavuus: Kuoret ovat suunniteltu pääsääntöisesti yksiytimisille asetuksille. Kuoren komentosarjojen suorituskyky perustuu yksittäisten komentojen suorituskyvylle, ja suurin osa niistä ei skaalaudu.
- Tarpeeton uudelleenlaskenta: Pienet muutokset komentosarjojen syötteeseen aiheuttavat kokonaisen uudelleensuorittamisen, joka kuluttaa tarpeettomasti aikaa.
- Ei tukea samanaikaisille käyttöönotoille: Kuoren toiminnat ovat suunniteltu suoritettavaksi samalla koneella. Kuitenkin nykypäivän hajautetuissa järjestelmissä on muitakin kuoren toiminnoille tarkoitettuja ratkaisuja, jotka ovat ristiriidassa kuoren ja näiden ratkaisujen tarpeiden kanssa.

Haitallisten ominaisuuksien lisäksi tutkimus listasi myös kuorien hyötyjä:

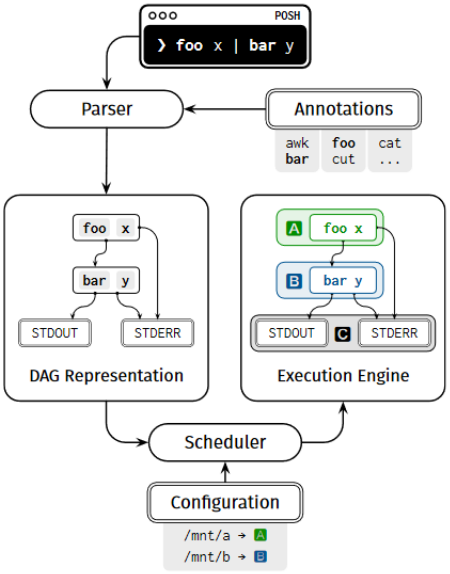
- Universaalisuus. Kuori toimii yleisenä kielenä ja välittäjänä eri kielten ja ohjelmistojen välillä.
- Tietovoiden käsittely. Unix-kuorissa on oma sovellusaluekieli, jolla kuoret pystyvät putkittamaan tietovoiden laskentaa ja vuorottaisuusrutiineja.
- Unix-natiivisuus. Kuoren ominaisuudet ja käsitteet ovat hyvin yhteensopivia Unixin tiedostojärjestelmien ja käsitteiden kanssa.
- Vuorovaikutteisuus. Kuoren ohjelmointiympäristö on erittäin vuorovaikutteinen ja nopea käyttää.

Unix Shell Programming: The next 50 years -artikkeli (Greenberg, Kallas ja Vasilakis 2021) esittää kuoren neljälle pääheikkoudelle kolme mahdollistajaa, jotka pienentävät näiden heikkouksien vaikutusta ja mahdollistavat paremman käytön. Kaksi näistä mahdollistajista on suhteellisen tuoreita, ja yksi on tutkimuksessa tehty uusi ehdotus. Ensimmäiset mainituista tuoreista mahdollistajista ovat Smoosh ja Libdash. Smoosh on Greenbergin luoma kuori, jonka tarkoituksena on formalisoida POSIX-kuoren semantiikkaa. Täysin toimivan vuorovaikutteisen kuoren lisäksi Smoosh toimii lisäksi myös koneellisena POSIX-standardin viitesemantiikkana. Smoosh on kirjoitettu konekielimäisellä Lem-kielellä, joka voidaan kääntää Coq-kielelle todistusta varten ja OCaml-kielelle suorittamista varten. Smooshin toiminnan mahdollistaa sen käyttämä libdash-kirjasto, jolla pystytään jäsentämään skriptit abstrakteiksi syntaksipuiksi ja sen jälkeen jäsentämään ne takaisin skripteiksi. Seitsemään muuhun kuoreen vertailtaessa Smoosh osoittautui vähiten ohjelmointivirheitä sisältäväksi. Se pystyi myös löytämään ohjelmointivirheitä muista testin kuorista. Greenbergin mielestä Smoosh on yksi kuoren päämahdollistajista, koska se auttaa muun muassa kuorenkäyttäjiä tutkimaan POSIX-kuorta ja sen käyttäytymistä sekä auttamaan käyttäjiä ymmärtämään kuorten välisiä eroavaisuuksia. Toiseksi päämahdollistajaksi Greenberg argumentoi PaSh:ia ja POSH:ia. PaSh (Kuvio 1, sivu 9) on järjestelmä, jonka pystyy rinnastamaan kuoriohjelmia automaattisesti (Vasilakis ym. 2021). PaSh on lähteestä-lähteeksi-kääntäjä, joka muuntaa skriptit tietovuokaavioiksi, rinnakkaistaa ne ja muuntaa ne takaisin skripteiksi. Sen lisäksi PaShia voidaan käyttää ajonaikaisena komponenttina, joka käsittelee suoritukseen ja virheisiin liittyviä ongelmia. POSH (Kuvio 2, sivu 9) on kehys, jonka tarkoitus on nopeuttaa komentoriviputkia syöttötulostusperäisillä komponenteilla, joilla on pääsy hajautettuihin verkkolevyjärjestelmiin. POSH keskeyttää putket ja siirtää yksityiset komennot ajettavaksi välipalvelimille.

(Raghavan ym. 2020.) Greenbergin tutkielma luokittelee Pash ja POSH -yhdistelmän toiseksi kuoren päämahdollistajaksi neljästä syystä:



Kuvio 1. PaSh (Vasilakis ym. 2021)



Kuvio 2. POSH (Raghavan ym. 2020)

- PaSH ja POSH kumpikin toimivat kehyksinä, jotka tekevät ohjelmamuunnoksia automaattisesti.
- PaSH ja POSH tarjoavat lähtökohdan suoritusta parantaville työkaluille.
- Kumpikin tarjoavat merkintäkielet, joiden avulla voidaan päätellä mielivaltaisia komentoja.
- PaSh ja POSH tarjoavat tietovuomallin kuoren osajoukoille, mikä mahdollistaa jatkotutkimukset ja muunnokset kuorille.

PaShissa ja POSHissa on kuitenkin Greenbergin artikkelin mukaan kaksi estettä, jotka haittaavat niiden käyttöönottoa: kuoren dynaamisuus ja järjestelmän resurssien rajoitteet. PaSh:in ja POSHin kaltaiset ratkaisut vaativat staattisia ja ennalta suoritettavia toimenpiteitä, mikä on ristiriidassa kuoren dynaamisen luonteen kanssa. Lisäksi resurssien rajallisuus haittaa PaSh:in ja POSH:in toimintaa. PaSh vaatii paljon tallennustehoa ja muistitilaa ajoympäristöltään. POSH taas olettaa, että sitä käytetään lohkostossa, jossa laskennalliset resurssit solmua kohden ovat rajattomia.

Tutkimuksessa mainittu ehdotus kolmanneksi mahdollistajaksi on Jash (just a shell). Jash on Greenbergin ehdottama dynaaminen optimointijärjestelmä, jonka tarkoitus olisi tarkastaa jokainen komento ja katsoa, löytyykö sille parempaa vaihtoehtoa. Dynaamisuutensa ansiosta Jash osaisi ottaa huomioon nykyisen järjestelmän olosuhteet tarkistaakseen, pystyykö optimointia tekemään nykyisessä tilassaan. Tämän Jash toteuttaisi käyttämällä JIT:n kaltaista kokoelmakehystä, jossa kääntäjää kutsutaan mahdollisimman myöhään toimittamaan tarvittavat ajonaikaiset tiedot. Näin kuori siirtyisi edestakaisin tulkinnan ja optimoinnin välillä. Resurssien rajallisuuden Jash sivuuttaisi luomalla resurssitietoisen optimointiproseduurin, joka valmistettaisiin kustannustietoisen tietovuomallin pohjalle. Tämä mahdollistaisi kaavioiden uudelleenkirjoittamisjärjestelmän, joka osaisi luoda annetulla kustannusbudjetin mukaisesti tavoitekohtaisia muunnoksia. Lopulta Jash on yksi kuoren päämahdollistaja, sillä sen pitäisi toimia kaikilla POSIXin mukaisilla kuoriskripteillä, jotka toteuttavat POSIXin dynaamisia standardeja sekä mahdollistaa staattiset operaatiot kuoren dynaamisessa ympäristössä. Greenbergin mukaan kaikki kolme mahdollistajaa yhdessä mahdollistaisivat alan tutkimusmahdollisuuksien laajentumisen dynaamisen kuoren kautta. Tällaisia tulevaisuuden tutkimussuuntia olisivat hajautetut laskennat, inkrementaaliset laskennat, ilmaisuvoimaisuuden tuki ja kuoren työkalut. Greenberg argumentoi artikkelissaan, kuinka hänen ehdottamansa kolme päämahdollistajaa voisi edistää kuoren tutkimuksessa kuoren hajauttamista, jota on yritetty toteuttaa tutkimusalalla jo pitkään. Päämahdollistajien uskotaan olevan yhdessä tarpeeksi yksinkertainen ja kevyt järjestelmä ylittämään osan hajautuksen yhteydessä tulevista ongelmista. (Greenberg, Kallas ja Vasilakis 2021.)

PaShin ja Poshin sekä Jashin yhteisesti luoma kehys saattaisi auttaa vähentämään kuoren nykyistä tarpeettoman uudelleenlaskennan aiheuttamaa ongelmaa, jonka Greenberg argumen-

toi olevan yksi olennaisimmista kuoren ongelmatekijöistä. Kuoriohjelmointi hyötyisi paljon inkrementaalisesta laskennasta, sillä kuoriohjelmointi perustuu iteratiiviseen ohjelmointiin. Lisäksi monet kuoren toiminnat ovat suhteellisen raskaita moneen uudelleen suoritettaviksi. Vaikka inkrementaalisen laskennan eri tyyppejä on tutkittu laajalti, ohjelmointimallia on lähes aina pitänyt laajentaa saadakseentarpeellisten tietojen ulos saamiseksi, mikä yleensä rikkoo vanhemmat komentosarjat. PaSh ja POSH auttaisivat paljastamaan rinnakkaislaskennan käyttöönottoa varten tarpeelliset puuttuvat tiedot. (Greenberg, Kallas ja Vasilakis 2021.)

Tutkimuksen esittämät mahdollistajat myös edistäisivät kuoren heurestista tukea. Kuorten komentojen ja niiden käyttäytymisen päättelyyn PaSh:n ja POSH:n luoma yhteinen komentomäärittelykehys auttaisi määrittelemään komentojen rinnakkaisuuteen liittyviä puolia. Artikkeli myös arvioi Smooshin tarjoavan mahdollisuuden kuoren tukijärjestelmien kehitykselle, jonka artikkelissa nähdään helpottavan kuoren mielivaltaisuutta. (Greenberg, Kallas ja Vasilakis 2021.)

2.1.8 Kuoren vaarat

Kuoren tehokkuudella on omat haittapuolensa. Koska kuoret pääsevät käsiksi suoraan käyttöjärjestelmään, niillä pystyy saamaan paljon tuhoa aikaiseksi. Tämän takia kuorista löydetyt tietoturva-aukot tai turvattomat kuoriohjelmat saattavat koitua erittäin vaarallisiksi. Yksi tunnetuimmista tietoturva-aukoista oli Bashista vuonna 2014 löydetty tietoturva-aukko, jota kutsuttiin muun muuassa nimillä shellshock tai bashdoor. Tietoturva-aukko koski kaikkia versioita versiosta 1.0.3 lähtien versioon 4.3 asti. Shellshockilla pystyttiin etäältä suorittamaan komentoja ympäristömuuttujista. Vaikka Bash ei ole itsessään yhteydessä verkkoon, monet verkossa olevat palvelimet käyttävät ympäristömuuttujia kommunikoidessaan palvelimen käyttöjärjestelmän kanssa. Tämän takia hyökkääjät pystyisivät suorittamaan komentoja etäyhteydellä ympäristömuuttujista käsin. (Mary 2015.)

Toinen esimerkki kuoren haavoittuvuuksien vakavuudesta oli tunnetun Steam-nimisestä pelialustasta löydetty kuoriohjelmointivirhe nimeltä Steam-cleaning, joka saattoi poistaa käyttäjän kaikki tiedostot rekursiivisesti (Greenberg 2017).

2.2 Muut mahdolliset vaihtoehtoiset teknologiat kuoren sijaan

Kuoret ovat usein de facto -työkalu niin sanottuna liimakielenä, eli kielenä, joka toimii muiden kielien ja teknologioiden välittäjänä. Kuoren muihin tehtäviin kuuluvat usein muun muassa automatisointi, asennukset ja käyttöönotot. Kuitenkin kuoren perinteisesti suorittamiin tehtäviin on kehitetty vuosien ajan vaihtoehtoisia menetelmiä. Monet ohjelmointikielet ovat saaneet vaikutteita kuorten luontaisista ominaisuuksista, kuten tulkattavuudesta ja vuorovaikutteisuudesta.

2.2.1 Python

Python on interaktiivinen ja tulkattava yleiskäyttöön tarkoitettu korkean tason ohjelmointikieli. Python tukee monia eri ohjelmoinnin tapoja, kuten olio-ohjelmointia ja funktio-ohjelmointia. Pythonin syntaksia kuvataan usein yksinkertaiseksi ja selkeäksi, ja Python ylipäänsäkin koetaan helposti opittavaksi kieleksi. Python voidaan ja usein luokitellaankin usein komentosarjakieleksi (Prechelt 2003).

Python onnistuu jatkuvasti pääsemään kaikista suosituimpien ohjelmointikielten listojen kärkeen. TIOBE:n[13] luokitelman mukaan Python havittelee vuosi vuodelta kärkiasemaa, asettumalla vuonna 2020 toiseksi suosituimmaksi ja vuonna 2021 kaikkein suosituimmaksi.

Python on vuorovaikuttuneisuutensa ja tulkattavuutensa takia kuoren kaltainen ohjelmointikieli. Eroja kuitenkin Pythonin ja kuorien välillä on. Greenbergin mukaan kuoren yksi neljästä isosta ongelmasta on se, että kuorista uupuu tuki moniydinprosessoreille. Kuorikomennot eivät myöskään skaalaudu, mikä rajoittaa kuoriohjelmien suorituskykyä. Python, joka on korkeamman tason ohjelmointikieli ja optimoitu toimii siis täten kuorta paremmin ja tehokkaammin suurempien datamäärien kanssa. Pythoniin saa mukaan moniydinprosessoinnin muun muassa *Parsl*-kirjastolla (Y. N. Babuji ym. 2018, Y. Babuji ym. 2019).

Noah Gift argumentoi artikkelissaan *Python for Bash scripters: A well-kept secret* Pythonin olevan suosituinta kuorta Bashia tehokkaampi ja vähintään yhtä siirrettävä (Gift ja Jones 2008). Artikkeli kertoo Pythonin saavuttavan kuoren kaltaisen vuorovaikutteisuuden IPython

[13] https://www.tiobe.com/tiobe-index/ (tarkistettu 11.11.2021)

nimisen Python-kuoren avustuksella.

Vaihtoehtoisesti Bashin komennot ja skriptit pystyttäisiin tarvittaessa ajamaan Pythonilla. Bash2Py on kääntäjä, jonka tarkoitus on kääntää Bashilla kirjoitetut skriptit Python-skripteiksi. Bash2Py:ta esittelevässä artikkelissa tehtyjen kokeilujen mukaan Bash2Py onnistui kääntämään 90 % tutkimukseen valituista aidoista avoimen lähdekoodin projekteista löytyneistä skripteistä (Davis ym. 2015).

Pythonin suosio voi osittain johtua myös sen syntaksista. Pythonin syntaksi osoittautui vuonna 2013 ohjelmointikielten syntakseja vertailevassa tutkimuksessa C-kielen syntaksiin perustuvia kieliä intuitiivisemmaksi. C-kieliin perustuvat kielet, joita tutkimuksessa edustivat Perl ja Java eivät kohonneet satunnaisesti generoitua kieltä paremmiksi. (Stefik ja Siebert 2013.)

2.2.2 Perl

Perl on tulkattava ja useita ohjelmointitapoja tukeva ohjelmointikieli, joka on komentosarjakielen kaltainen. Ensimmäinen versio Perlistä julkaistiin vuonna 1987, ja siitä on tullut sen jälkeen useita versioita. Vaikka ”Perl 6” on julkaistu, tämä koetaan alkuperäisestä Perlistä erilliseksi, uudeksi kieleksi, joten uusin versio varsinaisesta Perlistä on Perl 5. Perl 6 nimettiin uudelleen Rakuksi vuonna 2019. Perlin tuleva versio Perl 7 tiedoitettiin vuoden 2020 kesäkuussa[14]. Perl on suhteellisen suosittu ohjelmointikieli, ja se sijoittuikin TIOBE:n 2021 suosittujen kielten indeksissä [15] kuudennelletoista sijalle.

Perl on kirjoitettu C-kielellä, ja se on myös ottanut paljon vaikutteita syntaksiinsa C-kieleltä. Perl on monen ohjelmointimenetelmän kieli, joten se pystyy toimimaan muillakin ohjelmointimenetelmillä kuin proseduuriohjelmointina, esimerkiksi olio-ohjelmointina ja funktio-ohjelmointina. Perl on myös ottanut vaikutteita kuorista ja pystyy toimimaan komentosarjakielenä. Komentosarjakielten tapaan Perl on vuorovaikutteinen sekä tulkattava. Perliin on integroitu kuoressa usein käytettyjen sovellusaluekielityökalujen, tekstinkäsittelyyn tarkoitetun AWKin ja tekstivirrankäsittelyyn tarkoitutetun sedin toiminnallisuudet[16].

[14]https://news.perlfoundation.org/post/perl_7_announced_sawyerx_conference

[15]https://www.tiobe.com/tiobe-index/ (tarkistettu 11.11.2021)

[16]http://history.perl.org/PerlTimeline.html

Komentosarjamaisuutensa takia Perlillä pystyy suorittamaan monia tehtäviä, joihin yleensä käytetään kuorta. Perlissä on joitakin etuja verrattuna kuoriin. Perl on koettu erittäin nopeaksi kieleksi, ja sen suoritusteho on havaittu tehtävästä riippuen kuoria nopeammaksi. Perl-ohjelmat ovat myös jossain määrin itsenäisempiä ja siirrettävämpiä kuin kuoriohjelmat. Tämä näkyy esimerkiksi siinä, että Perl sisältää itsessään monia hyödyllisiä ominaisuuksia, jotka kuoressa riippuisivat ulkoisista työkaluista. Näitä ovat muun muassa AWK ja sed. Ulkoisten työkalujen lisäksi monien kuorien yhtenäisestä syntyperästä huolimatta kuorien väliltä löytyy usein hienovaraisia, mutta merkittäviä syntaktisia eroavaisuuksia.

2.2.3 Ansible

Ansible on vuonna 2012 julkaistu avoimen lähdekoodin automaatiomoottori, joka toimii ensisijaisesti Unixin kaltaisilla järjestelmillä. Ansiblella voi muun muassa automatisoida pilvipalveluita, konfiguraationhallintaa sekä sovellusten käyttöönottoa[17].

Ansible tarvitsee Pythonin ja tarvittavat Pythonin paketit asennetuksi sekä asiakkaalle että palvelimelle. Ansiblen erikoisuus on sen arkkitehtuuri, jota kutsutaan ”agentless architecture”:ksi. Arkkitehtuuri ei siis käytä ohjelmia, jotka suorittavat jatkuvia ja itsenäisiä tehtäviä yksilön tai organisaation puolesta. Sen sijaan Ansible ottaa väliaikaisia etäyhteyksiä asiakkaaseen SSH:lla tai WinRm:llä ja PowerShellin välityksellä. Ansible ei siis vaadi Pythonin, SSH:n tai WinRM:n/PowerShell:n lisäksi sovellusten asentamista asiakaspäätteelle. Väliaikaisten yhteyksien ansiosta Ansible ei käytä turhaan resursseja asiakaskoneelta silloin, kun Ansible ei suorita tehtäviään. Ansible luonnehtii omiksi suunnittelutavoitteekseensa *minimaalisen luonteen*, *johdonmukaisuuden*, *turvallisuuden*, *luotettavuuden* ja *minimaalisen opittavuuden*. Kuoreen verrattuna Ansible tarjoaa luotettavuutta idempotenssin ansiosta[18].

Ansible työkaluna on tarkoitettu käytettäväksi samoissa ja samankaltaisissa tehtävissä kuin kuoria käytetään. Näihin tehtäviin kuuluu muun muassa automatisointi, konfiguraatio ja ohjelmistojen käyttöönotot. Tämän takia Ansiblea on ehdotettu kuoren korvikkeeksi tiettyihin kuorille tarkoitettuihin tehtäviin, pääsääntöisesti automaatioon[19].

[17] https://www.ansible.com/

[18] https://www.ansible.com/resources/whitepapers/ansible-in-depth

[19] https://www.ansible.com/

Ansiblen yksi tehokkaimmista ominaisuuksista on sen kyky ajaa rinnakkain kuorikomentoja monelle eri koneelle ad-hoc -tilassa[20]. *Agentless Architecture* -tyylisen arkkitehtuurinsa ansiosta Ansible on kuoria tietoturvallisempi, sillä se ei silloin tarvitse erillisiä ohjelmistoja etäyhteydensä päässä, mikä pienentää mahdollista hyökkäyspintaa. Ainoat ohjelmat mitä Ansible tarvitsee etäyhteyteen ovat OpenSSH tai WinRM -palvelinprosessit, joiden on koettu kuuluvan turvallisimpiin ohjelmistoihin.

Vaikka Ansiblessa on omat tietoturvavahvuutensa kuoreen verrattuna, siinä on kuitenkin omia tietoturvariskejään. Vuonna 2021 julkaistussa tutkimuksessa ”Security Smells in Ansible and Chef Scripts: A Replication Study” (Rahman ym. 2021) Ansiblesta löydettiin kuusi potentiaalista tietoturvariskiä. Ensimmäinen riski Ansiblessa oli mahdollisuus käyttää tyhjiä salasanoja, mikä yleensä merkitsee Ansiblen sallivan heikkojen, helposti arvattavien salasanojen käytön. Toinen havaittu riski oli kovakoodatut salasanat, käyttäjätunnukset tai salausavaimet. Kolmantena potentiaalisena riskinä oli Ansiblessa oli nettilatauksista puuttuvat tarkistussummien tarkastukset tai muut eheyden tarkastukset. Eheyden tarkastus estäisi järjestelmää lataamasta mahdollisen hyökkääjän korruptoimaa turvatonta materiaalia. Neljäs Ansiblen riski oli epäilyttävät kommentit, eli Ansible antaa koodin kommenteihin virheisiin tai puuttuvaan koodeihin viittavia avainsanoja, kuten BUG, HACK tai TODO. Ansiblessa löydetty viides mahdollinen tietoturvallisuusriski oli IP-osoitteiden rajoitusten puute, eli Ansible antaa käyttäjän asettaa tietokantapalvelimien IP-osoitteeksi 0.0.0.0. Kuudes ja viimeinen Ansiblen tietoturvallisuusriski oli HTTP:n käyttö ilman TLS:ää tai SSL:ää. (Rahman ym. 2021.)

[20]https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html

3 Teknologian adaptaatio

3.1 Innovaation leviäminen

Teknologian adaptaation elinkaari on malli, jolla kuvataan uuden teknologian tai innovaation käyttöönottoa sosiologisesta näkökulmasta. Emerett Rogers on kirjassaan *Diffusion of innovations* tehnyt laajamittaisen tutkimuksen innovaation leviämisestä, mitä on myös käytetty monessa aiheeseen liittyvässä tutkimuksessa (Rogers 2010).

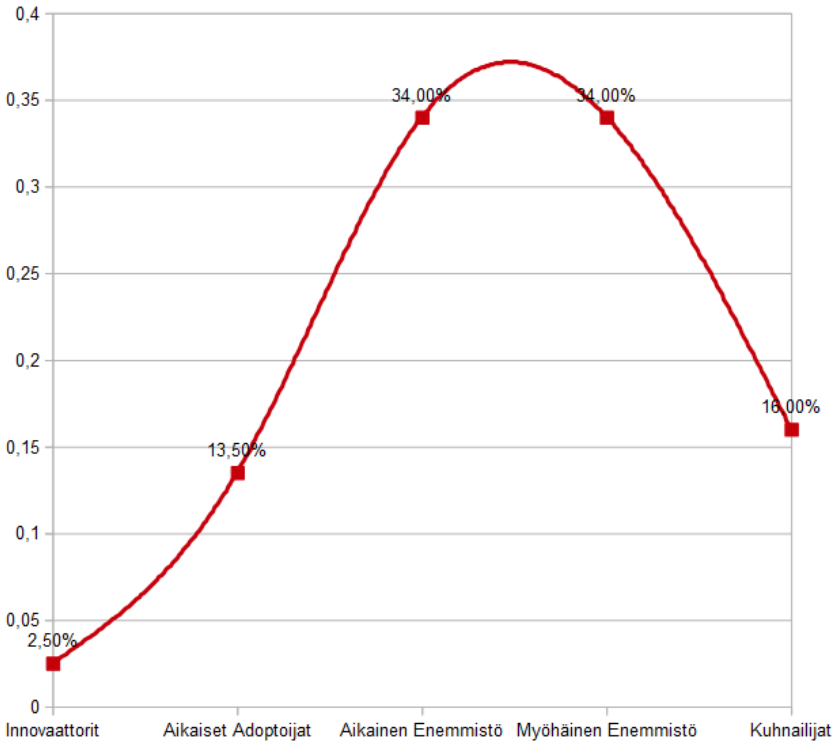
Innovaation leviämisen neljä pääelementtiä ovat *innovaatio*, *viestintäkanavat* (engl. communication channels), *aika* sekä *yhteiskuntajärjestelmät* (engl. social systems). Innovaatioksi luetaan ideat, käytännöt ja keksinnöt, jotka yksilö tai jonkin sortin yksikkö kokee uutuudeksi. Innovaation ei tarvitse itsessään olla uusi, vaan sen on oltava uusi ainoastaan yksilön omasta näkökulmasta. Esimerkiksi kymmenjärjestelmä on Euroopassa vanha innovaatio, mutta se olisi Yhdysvalloissa uusi. (Rogers 2010.)

Kaikki innovaatiot eivät ole samanarvoisia ja innovaatioille on määriteltävissä ominaisuuksia, jotka nopeuttavat niiden käyttöönottoa. Rogers luokittelee viisi ominaisuutta innovaatiolle. Nämä ovat *relativiinen etu*, *yhteensopivuus*, *monimutkaisuus*, *kokeiltavuus* ja *havaittavuus*. *Relatiivisella edulla* tarkoitetaan uuden innovaation tarjoamaa hyötyä verrattuna aikaisempaan tilanteeseen. *Yhteensopivuus* viittaa kuinka hyvin innovaatio on yhdenmukainen jo olemassa olevien arvojen, kokemuksien ja käyttöönottajien mahdollisten tarpeiden kanssa. *Monimutkaisuus* on luonnollisesti innovaatiossa havaittu käytön hankaluus tai ymmärtämisvaikeus. *Kokeiltavuudella* mitataan, kuinka paljon innovaatiota pystytään testaamaan tai kokeilemaan ennen käyttöönottoa. *Havaittavuudella* tarkoitetaan innovaation tuottamien tulosten näkyvyyttä muille. (Rogers 2010.)

Viestintäkanava on kahden yksilön välinen yhteys, jonka kautta innovaatio voi levitä sen tietävältä yksilöltä tietämättömälle yksilölle. Joukkoviestintävälineet ovat yleensä kaikkein tehokkaimpia viestintäkanavia. Näitä ovat esimerkiksi televisio, radio, lehdet tai Internet. Ihmistenväliset viestintäkanavat taas ovat kahden tai useamman yksilön välinen viestintäkanava. (Rogers 2010.)

Innovaation leviäminen tapahtuu ajan myötä eri vaiheissa. Ensimmäisessä innovaation päätösprosessissa yksilö joko hylkää tai ottaa innovaation käyttöön. Tämä prosessi tapahtuu viidessä vaiheessa, jotka tulevat ajan myötä eri viestintäkanavien kautta yhteisön jäsenille. Ensimmäisessä vaiheessa yksilö tai muu päätöksistä huolehtiva yksikkö tulee *tietoiseksi* kyseisen innovaation olemassaolosta, mahdollisesti ilman syvempää käsitystä tai kiinnostusta aiheeseen. Seuraavaksi tulee *suostuttelun* vaihe, jolloin yksilölle tai yksikölle muodostuu suotuisa tai epäsuotuisa käsitys innovaatiosta, jonka jälkeen tehdään *päätös* innovaation käyttöönotosta. Neljännessä vaiheessa innovaatio *käyttöönotetaan*. Lopuksi yksilö tai yksikkö *vahvistaa* käyttöönotetun innovaation, ellei sitä vastustavaa tietoa ilmaannu. (Rogers 2010.)

Emerett Rogers luokittelee innovaation käyttöönottajat innovaattoreihin, aikaisiin adoptoijiin, aikaiseen enemistöön, myöhäiseen enemistöön ja kuhnailijoihin (Rogers 2010). Tämä luokituksen käyttöönottajien jakauma muodostaa Gaussin kellokäyrän(Kuvio 3, sivu 17).



Kuvio 3. Innovaation käyttöönottajat

Innovaattorit (engl. innovators) ovat pieni vähemmistö, jotka ottavat uuden innovaation käyt-

töön riskeistä huolimatta. Innovaattoreihin kuuluu 2,5 % käyttöönottajista. *Aikaiset adoptoijat* (engl. early adopters) ovat yhteisön *johtajia*, jotka ottavat uuden innovaation käyttöön hieman aiemmin kuin muut yhteisön jäsenet. Aikaiset adoptoijat ovat kaikista käyttöönottajista 13,5 % osuus. *Aikainen enemmistö* (engl. early majority) ottaa uuden innovaation käyttöön aikaisemmin kuin keskitason yhteisön jäsenet. *Myöhäinen enemmistö* (engl. late majority) koostuu myöhäisistä käyttöönottajista, jotka ovat yleisesti muita skeptisempiä uusia innovaatioita kohtaan. Aikainen ja myöhäinen enemmistö muodostavat kummatkin erikseen 34 % käyttöönottajista. *Kuhnailijat* (engl. laggards) ovat viimeisiä, jotka ottavat innovaation käyttöön, usein silloin, kun kyseinen innovaatio on jo ehtinyt vanhentua. Innovaation käyttöönottajista 16 % on kuhnailijoita. (Rogers 2010.)

Yhteiskuntajärjestelmäksi lasketaan minkä tahansa kokoinen yhteisö, jonka jäsenet yhtenäisesti ratkovat ongelmia saavuttaakseen yhteisen tavoitteensa. Esimerkiksi yhteiskuntajärjestelmä voi olla vaikka pieni kylä tai kaikki valtion kuluttajat yhteensä. Yhteiskuntajärjestelmä toimii innovaation leviämisen rajoina ja yhteiskuntajärjestelmän rakenne myös vaikuttaa siihen, miten innovaatio leviää itsensä sisällä. Esimerkiksi yhteiskuntajärjestelmän sisäiset normit ja arvot vaikuttavat innovaation leviämiseen. (Rogers 2010.)

Yhteiskuntajärjestelmässä kaikki yksilöt eivät myöskään ole tasavertaisia innovaation levittäjiä. Usein kaikkein innovatiivisin järjestelmän jäsen koetaan muista poikkeavaksi ja hänen asemansa saatetaan kyseenalaistaa, mikä rajoittaa hänen rooliaan innovaation levittäjänä innovatiivisuudesta huolimatta. Sen sijaan yhteiskuntajärjestelmässä on yksilöitä, joilla on niin sanottu mielipidejohtajan rooli. Mielipidejohtajat ovat yksilöitä, jotka pystyvät vaikuttamaan ja muuttamaan muiden yksilöiden asenteita ja käytöstä järjestelmässä epävirallisen auktoriteetin mukaan. Kun mielipidejohtajat ovat usein innovatiivisia, innovaatio mukautuu hyvin järjestelmän normien kanssa. (Rogers 2010.)

Artikkeli *An Empirical Study of Programming Language Trends* (Chen ym. 2005) luokittelee ohjelmointikielten adaptaation sisäisiin ja ulkoisiin tekijöihin, joista sisäiset tekijät ovat kielen vakiona pysyvät ominaisuudet, ja ulkoiset jatkuvasti muuttuvat yhteisöllisemmät vaikuttajat. Sisäiset tekijät kyseisessä artikkelissa ovat *yleisluonteisuus*, *ortogonaalisuus*, *luotettavuus*, *huollettavuus*, *tehokkuus*, *yksinkertaisuus*, *laiteriippumattomuus*, *toteuttamiskelpoisuus*, *laajennettavuus*, *ilmaisuvoimaisuus* ja *vaikutus*. Näiden sisäisten tekijöiden välillä

neljäksi tärkeimmäksi tekijäksi osoittautuivat järjestyksessä *laiteriippumattomuus*, *laajennettavuus*, *yleisluonteisuus* ja *luotettavuus*.

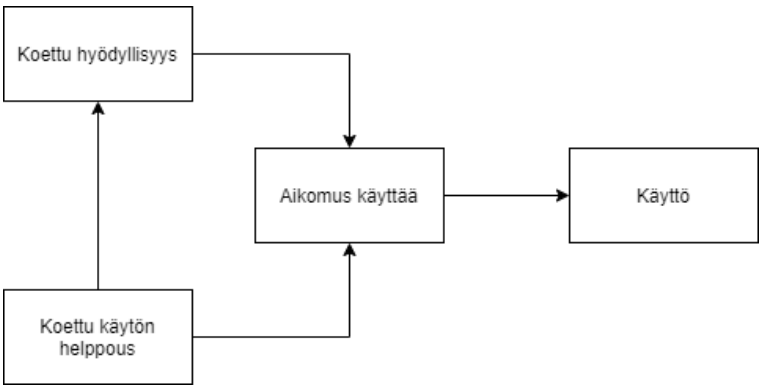
Ulkoiset tekijät taas valittiin seuraavasti: *institutionaalinen tuki*, *teollinen tuki*, *valtiollinen tuki*, *organisaatioiden tuki*, *ruohonjuuritason tuki* ja *teknologian tuki*. Tutkimuksessa ei korreloitu ulkoisia tekijöitä erikseen keskenään, vaan ne korreloitiin lukien mukaan sisäiset tekijät. Tutkimuksessa löydettiin vain vähän korrelaatiota näiden kaikkien tekijöiden välillä ja vain osalla löydettiin jotain merkittävää. Kaksi johtopäätöstä havainnoista kuitenkin luotiin. Ensimmäiseksi *institutionaalinen tuki* nousi tärkeäksi tekijäksi kaikkien tekijöiden välillä, mikä voi johtua instituutioissa oppineiden siirtymisestä teollisuuden työtehtäviin, tuoden mukanaan instituutioiden suhtautumiset. Toinen havainto oli *teknologian tuki*, joka vaikutti huomattavasti kaikkiin tekijöihin. Tämä saattoi myös merkitä, kuinka kielen menestys vaikutti sen saamaan tukeen, kuin tuen vaikuttavan kielen menestykseen. (Chen ym. 2005.)

Vuonna 2013 tehdyssä tutkimuksessa *Empirical Analysis of Programming Language Adoption* nimensä mukaisesti empiirisesti tutkittiin ohjelmointikielten adaptaatiota ja niiden yleistymiseen vaikuttavia tekijöitä (Meyerovich ja Rabkin 2013). Tutkimuksessa katsottiin aluksi, minkälaisia käyttäytymismalleja kielen suosio noudattaa. Osoittautui, että kielen suosio laskee potenssilakien mukaan. Suosittuja kieliä käytetään kaikentasoisilla alueilla, kun taas vähemmän suositut tippuvat omiin erikoispiireihin. Kielten käyttöönottoon eivät myöskään vaikuta kielen lingvistiset ominaisuudet, vaan käyttöyhteydet ja sen piirit. Toiseksi tutkimus selvitti, mitkä tekijät vaikuttivat kehittäjiin projektia aloitettaessa. Tutkimuksessa kielivalintaan vaikutti valmiin kielellä kirjoitetun koodin olemassaolo, tottumus kieleen ja kielelle toteutetut avoimen koodin kirjastot. Kirjastoihin panostaminen koettiin varsinkin olennaisena kielen käyttöönottoa puoltaessa.

3.2 Technology acceptance model

Technology acceptance model eli lyhennetyltä nimeltä TAM, on malli, jolla mallinnetaan, miten käyttäjät ottavat uuden teknologian tai tietojärjestelmän käyttöönsä. TAM:n tarkoitus on jäljittää ulkoisten tekijöiden vaikutus sisäisiin asenteisiin ja uskomuksiin. (Davis, Bagozzi ja Warshaw 1989.)

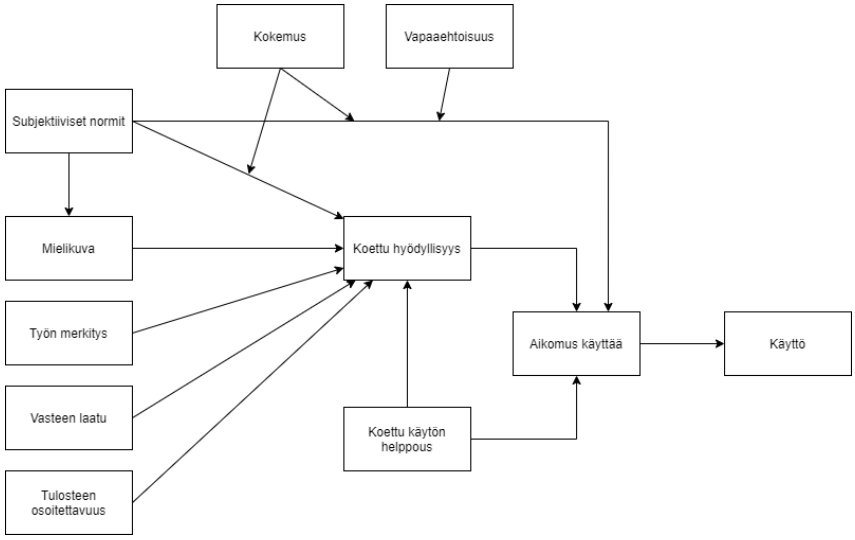
TAM:in (Kuvio 4, sivu 20) keskeiset asenteiden muuttujat ovat ”koettu hyödyllisyys” ja ”koettu käytön helppous”, jotka TAM:ssa koetaan tärkeimmiksi tietotekniikan teknologioihin liittyviksi asenteiksi. Koetulla hyödyllisyydellä tarkoitetaan käyttäjän subjektiivista havaintoa tehokkuuden noususta. Koetulla käytön helppoudella mitataan, kuinka vaivattomaksi käyttäjä havainnoi uuden tietojärjestelmän. Nämä kaksi muuttujaa vaikuttavat ”aikomukseen käyttää” teknologiaa, mikä taas johtaa ”todelliseen käyttöönottoon”. Tämän lisäksi mallin mukaan ”koettu käytön helppous” vaikuttaa suoraan ”koettuun hyödyllisyyteen”. (Davis, Bagozzi ja Warshaw 1989.)



Kuvio 4. TAM-kaavio

Laajennetussa TAM2-mallissa (Kuvio 5, sivu 21) TAMiin lisätään lisää teoreettisia rakenteita (Venkatesh ja Davis 2000). Nämä jaetaan kahteen luokkaan, jotka ovat sosiaaliset vaikutusprosessit ja kognitiiviset mittausprosessit. Sosiaalisiin vaikutusprosesseihin kuuluvat subjektiiviset normit, vapaaehtoisuus ja myöntyvyys sosiaalisiin vaikutuksiin ja mielikuva. Kognitiiviset mittausprosessit ovat työn merkitys, vasteen laatu, tulosten osoitettavuus sekä koetun käytön helppous.

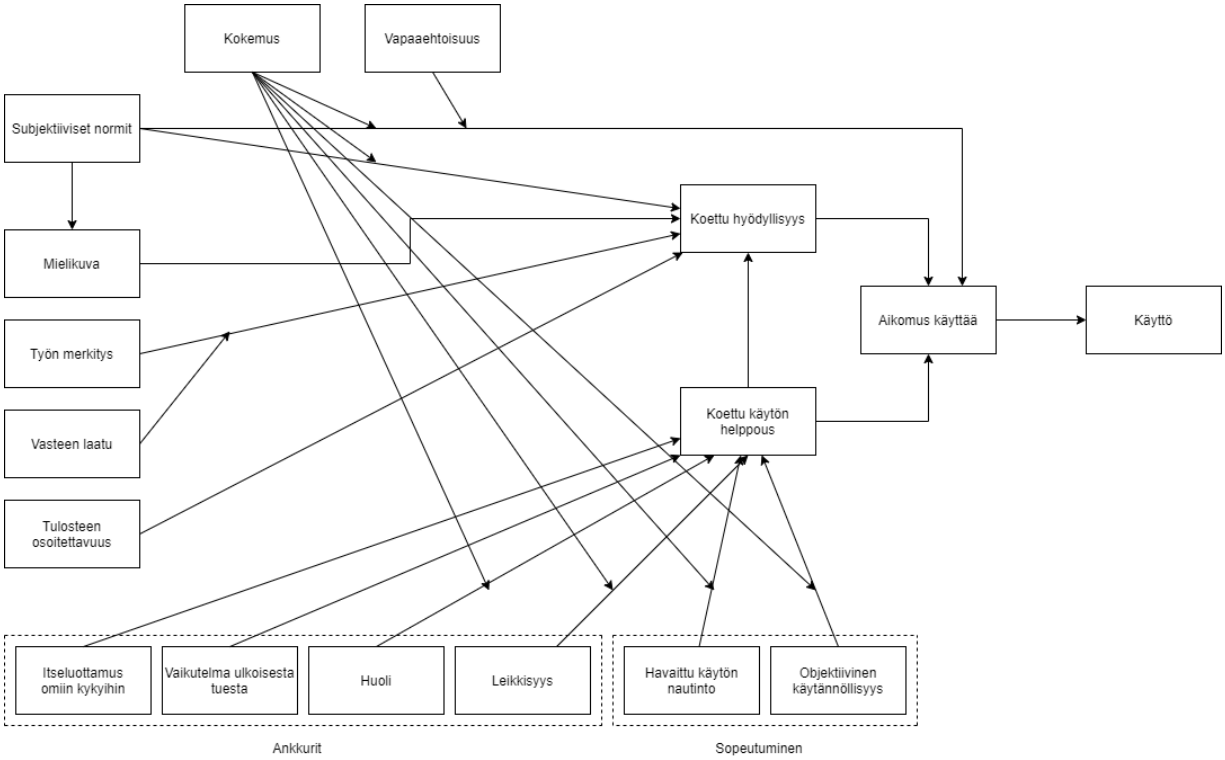
TAM3-malli (Kuvio 6, sivu 22) laajentaa vielä ulkoisia muuttujia kuudella, sekä lisää vanhoihin muuttujiin uusia suhteita ulkoisiin muuttujiin ja TAM-mallin päämuuttujiin. Lisätyt kuusi ulkoista muuttujaa perustuvat ankkurointivaikutukseen. Nämä muuttujat jakaantuvat neljään ankkurimuuttujaan ja kahteen sopeutusmuuttujaan. Ulkoiset muuttujat, jotka luokiteltiin ankkureiksi TAM3-mallissa, olivat luottamus omiin tietokoneen käyttökykyihin, yksilön oma vaikutelma ulkoisesta tuesta, huoli tietokoneiden käytöstä, sekä leikkisyys. Kaksi uutta sopeutumiseen liittyvää ulkoista muuttujaa olivat havaittu käytön nautinto ja objektii-



Kuvio 5. TAM2-kaavio

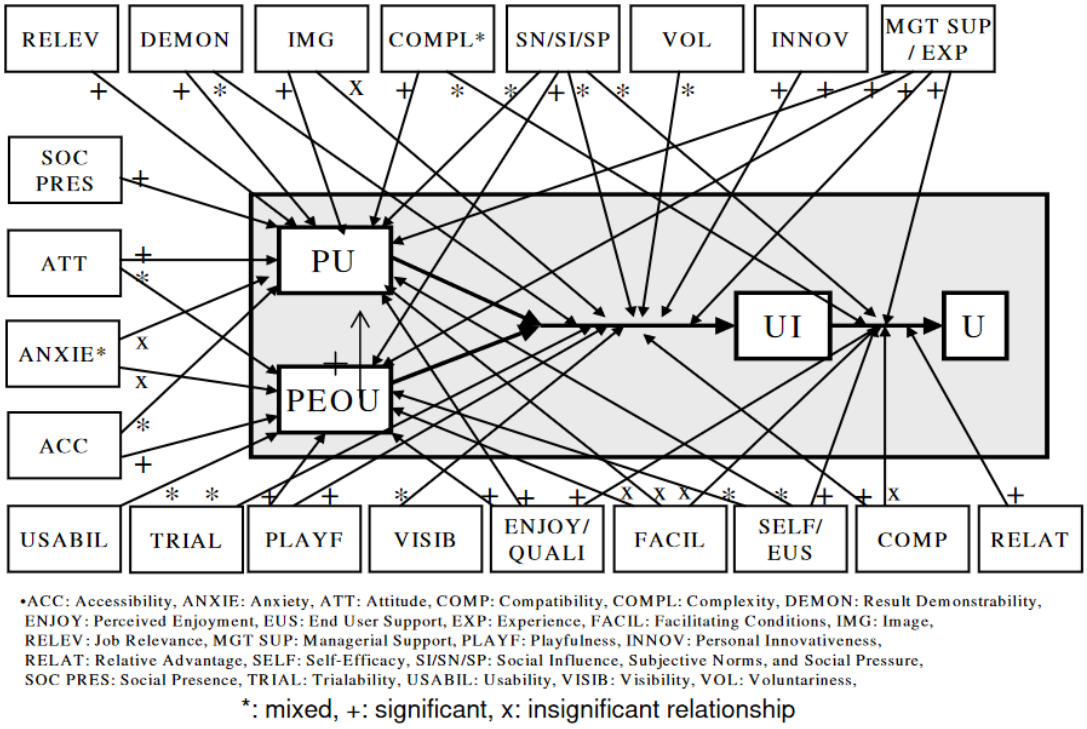
vinen käytännöllisyys. (Venkatesh ja Bala 2008.)

Vuonna 2003 tehdyssä TAM-mallia (Kuvio 7, sivu 23) arvioivassa tutkimuksessa *The Technology Acceptance Model: Past, Present, and Future* (Lee, Kozar ja Larsen 2003) kerättiin tietoa useista eri käytetyistä TAM-malleista ja koottiin niissä esiintyviä ulkoisia muuttujia tutkimuksessa tarkasteltavaan TAM-malliin. Tutkimus keräsi yhteensä 21 ulkoista muuttujaa eri malleista. Tutkimus tutki ja arvioi näiden ulkoisten muuttujien suhteiden merkitystä TAM-mallin päämuuttujiin, luokitellen suhteiden merkitykset merkittäviin, merkityksettömiin tai sekalaisiin. Kyseisessä tutkielmassa esitetyt ulkoiset muuttujat olivat *saavutettavuus*, *huoli tietokoneiden käytöstä*, *asenne*, *innovaation yhteensopivuus aiempien arvojen ja tarpeiden kanssa*, *innovaation monimutkaisuus*, *tulosteiden todistettavuus*, *havaittu nautinnollisuus*, *loppukäyttäjän tuki / itseluottamus omiin kykyihin*, *johtotuki / kokemus*, *käyttöönottoa edistävät olosuhteet*, *mielikuva*, *työn merkitys*, *leikkisyys*, *yksilön henkilökohtainen innovatiivisuus*, *innovaation koettu suhteellinen paremmuus entiseen teknologiaan*, *yhteisöllinen vaikutus*, *normit ja paineet*, *yhteisön läsnäolo*, *koetettavuus*, *käytettävyys*, *näkyvyys*, sekä lopuksi vielä *vapaaehtoisuus*. (Lee, Kozar ja Larsen 2003.) Lisäksi kyseinen tutkimus tarkastelee tutkimuksissa TAM-mallin käyttämisen rajoituksia ja heikkouksia. Yleisin rajoitus TAM-mallia käyttävissä tutkimuksissa oli itse raportoitu käyttö todellisen käytön mittaamisen sijaan, mikä usein aiheuttaa raportoinnissa vääristyneitä tuloksia. Toiseksi yleisin ongelma on vain yhden tietojärjestelmän käyttäminen liian tasakoosteiselle koehenkilöryh-



Kuvio 6. TAM3-kaavio

mälle vain yhdellä koetehtävällä, aiheuttaen yleistysongelman. Tutkimuksessa myös ilmeni liian monen TAM-tutkimuksen olevan poikittaistutkimuksia, joissa tutkimus suoritettiin pienellä aikavälillä. Tämä rajoittaa tutkimusta sen takia, että yksilön käsitykset ja aikomukset vaihtelevat ajan myötä. Lisäksi moni tutkimus ei käytä kuin ennalta arvioidun määrän eri ulkoisia muuttujia, eikä ota huomioon mallista puuttuvia ulkoisia muuttujia, jotka saattaisivat olla vaikuttavia tekijöitä.



Kuvio 7. Ulkoisten muuttujien välisten suhteiden merkittävyydet (Lee, Kozar ja Larsen 2003)

4 Tutkimuksen toteutus

Tutkimuksen tarkoituksena oli selvittää syitä kuoriohjelmoinnin yhä jatkuvalle käytölle tietoteknikka-alan yrityksissä. Tutkimus toteutettiin kyselytutkimuksena, jossa aineisto kerätään kyselyllä. Kysely tehtiin luomalla kyselylomake Webropol-sivustolla[1]. Lukuunottamatta taustatietoihin liittyviä kysymyksiä, kyselyn kysymykset olivat avoimia kysymyksiä, jotka jakaantuivat vuorovaikutteiseen kuoreen liittyviin kysymyksiin ja kuoriohjelmointiin liittyviin kysymyksiin. Sen jälkeen aineistot analysoitiin sisällönanalyysillä. Vastauksista nsotettiin eri teemoja ja mainintoja, jotka sitten luokiteltiin omiin luokitelmiin. Nämä maininnat sen jälkeen laskettiin niiden luokitelmia kohden.

4.1 Kyselytutkimus

Tämän tutkielman aineisto kerättiin kyselyllä. Kyselyn pääkysymykset kuudennesta kysymyksestä 17:sta kysymykseen olivat *avoimina kysymyksiä*. Kysymykset 1-5 olivat monivalintoja, joissa oli myös avoimia kohtia, kuten työnkuvasta vapaasti kirjoitettava *Muu*-vaihtoehto.

Tässä tutkimuksessa käytettiin kyselytutkimusta, jotta tutkimukseen saataisiin laajempi otanta alalla työskentelevistä ihmisistä ja jotta saataisiin vastauksia tutkimuskysymykseen alalla työskentelevien näkökulmasta. Koska kysymykset olivat avoimia kysymyksiä, niistä pystyttiin saamaan suoraa palautetta vastaajien syistä kuoren käyttämiseen, kuorella suoritettavista tehtävistä, sen hyödyistä ja sen haitoista. Samalla voitiin hiukan arvioida asenteita ja suhtautumisia kuoreen vastaajien vastausten sävyjen perusteella. Vastaukset voivat myös antaa itsestään uusia näkökulmia, joiden avulla voidaan tarkastella muitakin vastauksia sekä saada lisätietoa aiheesta.

Kyselytutkimuksen yhteydessä luonnollisesti tulee myös haittoja ja vaikeuksia, jotka pitää huomioida tuloksissa. Vastauksien todenmukaisuutta on hankala arvioida, eikä niistä välttämättä tiedä, kuinka huolellisesti tai luotettavasti niihin on vastattu. Kuitenkin avoimien kysymysten kohdalla täytyy nähdä jonkin verran vaivaa vastauksen kirjoittamiseen, minkä

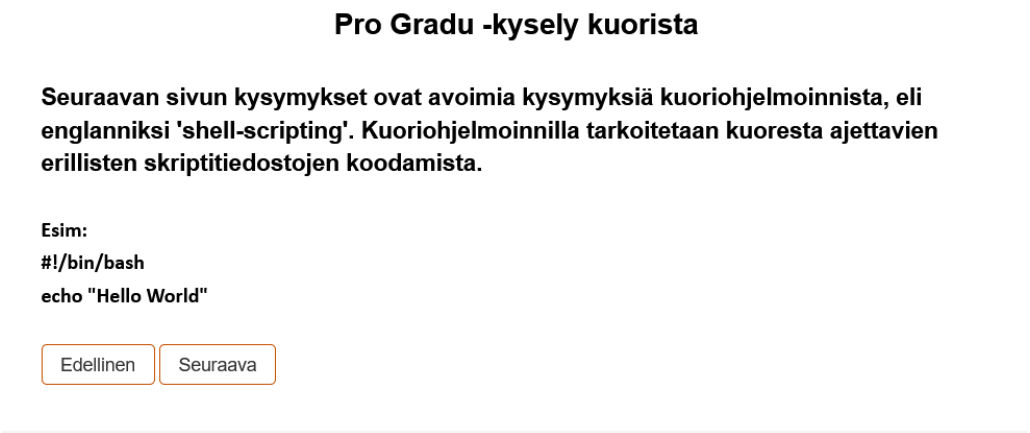
[1] https://new.webropolsurveys.com

voidaan katsoa lieventävän joidenkin tutkimuksen vastausten virhealttiutta. Suhteellisen pitkässä ja perusteellisessa kyselylomakkeessa haitaksi koituu usein suhteellisesti korkeampi vastaamattomuus kysymyksiin. Tässä tutkielmassa kokonaan tyhjäksi tai puolityhjäksi jätettyjä vastauksia tuli paljon, mutta siitä huolimatta sisällöllisiä vastauksia tuli tarpeeksi. Webropolista pystyi näkemään, kuinka moni oli jättänyt kyselyn kokonaan kesken, eli jättänyt lähettämättä ollenkaan. Näitä havaittiin myös suhteellisen paljon.

4.2 Pilotointi

Pilotointi ja kyselyn testaus aloitettiin Webropol-kyselyn pilottiversiolla. Testaajia oli kokonaisuudessaan kuusi. Kaksi ensimmäistä testaajaa olivat ohjaajan puolesta hankittuja. Loput neljä olivat gradun laatijan hankkimia. Suurin osa testaajista oli Jyväskylän yliopistolla työskenteleviä tai opiskelevia henkilöitä.

Ensimmäisissä kahdessa testissä testaajat eivät huomanneet tehtävien selvityksiä, eivätkä siten ymmärtäneet vastata kysymyksiin. Tämän takia selvitykset siirrettiin omille sivuilleen (Kuvio 8, sivu 25). Loput neljä osasivat vastata kysymyksiin, jotka vaativat aiemmilta vastaajilta parempaa selvitystä.



Kuvio 8. Tehtävien selvitys

4.3 Kyselyn jakelu

Vuoden 2021 tammikuun 12. päivä kysely jaettiin kolmelle suomenkieliselle tietotekniikka-aiheiselle keskustelupalstalle. Keskustelupalstat olivat *ohjelmointiputka*[2], *Suomi24*:n[3] tietotekniikan ja ohjelmoinnin keskusteluosio sekä *Mikrobitti*-tietotekniikkalehden[4] keskustelupalsta. Helmikuun 9. päivä kyselypyyntö jaettiin uudelleen *Suomi24*:ään, ja kahdelle muulle keskustelupalstalle, *TechBBS*:ään[5] ja *MuroBBS*:n[6] ohjelmointiosastolle. Kyselypyyntö poistettiin saman päivän aikana *TechBBS*:stä. Helmikuun loppuun mennessä kyselyyn oli vastannut 43 henkeä. Maaliskuun alussa *MikroBitti*[7] -lehti julkaisi artikkelin kyselystä ja pro gradu -tutkielmasta. Vastauksia tuli huhtikuun kolmanteen päivään asti, jonka jälkeen kysely suljettiin. Vastauksia kerääntyi kokonaisuudessaan 153.

4.4 Sisällönanalyysi

Tässä tutkimuksessa aineiston analyysissa hyödynnettiin sisällönanalyysin tekniikkaa. Tarkemmin ottaen tutkimuksessa hyödynnettiin sisällönanalyysin tyyppiä, jota kutsutaan *perinteiseksi sisällönanalyysiksi*. Kuitenkin sisällönanalyysin tekniikassa otettiin tiettyjä vapauksia. Kvalitatiivinen sisällönanalyysi voidaan luokitella kolmeen eri tyyppiin: *perinteiseen sisällönanalyysiin*, *suunnattu sisällönanalyysiin* ja *summatiivinen sisällönanalyysiin* (Hsieh ja Shannon 2005). *Perinteisessä sisällönanalyysissä* koodaukset luodaan läpilukemisen aikana ja koodaukset perustuvat suoraan aineistoon. *Suunnatussa sisällönanalyysissä* koodaukset luodaan ennen läpikäyntiä sekä sen aikana, ja koodaukset perustavat teoriaan. *Summatiivisessa sisällönanalyysissä* luodaan koodauksien sijaan avainsanoja, jotka perustuvat kirjallisuuteen tai tutkijan kiinnostuksiin, ja avainsanat tunnistetaan ennen läpikäyntiä ja sen aikana. Tässä tutkimuksessa koodaukset luotiin läpilukemisen aikana jakamalla vastauksissa esille tulleet maininnat luokiksi.

[2] https://www.ohjelmointiputka.net/

[3] https://keskustelu.suomi24.fi/tiede-ja-teknologia/tietotekniikka/ohjelmointi

[4] https://keskustelu.mikrobitti.fi/

[5] https://bbs.io-tech.fi/

[6] https://murobbs.muropaketti.com/forums/

[7] https://www.mikrobitti.fi/uutiset/kaytatko-shell-ohjelmointia-tyossasi-vastaa-kyselyyn/91a4f76d-b1c4-4965-b232-49bed6ecb092

Vaikka tutkimus on laadittu laadullisen sisällönanalyysin tyyliin, tutkimuksen sisällönanalyysissä on myös määrällisen sisällönanalyysin piirteitä. Vastauksien luokitukset ja toistuvat maininnat laskettiin niiden esiintyvyyksien mukaan ja korostettiin sen mukaan, mitä yleisemmin ne esiintyivät.

5 Tulokset

Kyselyn alussa käytiin läpi vastaajien taustatietoja, kuten tämän roolia yrityksessä tai kuoreen liittyvää osaamista. Tutkimuksen pääkysymykset olivat avoimia kysymyksiä. Avoimia kysymyksiä oli yhteensä kaksitoista, joista kuusi liittyivät kuoriohjelmointiin. Toiset kuusi olivat samankaltaisia, mutta perustuivat vuorovaikutteiseen kuoreen. Kyselyssä kysyttiin, mihin käyttäjät käyttävät kuoria, minkä takia niitä käytettiin ja niiden hyödyt ja hankaluudet. Lopuksi kyseltiin, haluaisiko vastaaja lisätä tai vähentää niiden käyttöä, ja millä teknologialla kuoret pitäisi korvata, jos niin edes haluttaisiin tehdä.

Aineisto käytiin läpi lukemalla kaikki vastaukset läpi kysymyskohtaisesti. Vastauksista poimittiin niissä esiintyvät maininnat ja teemat, jotka sitten luokiteltiin ja laskettiin.

Kysymykset 1-5: Taustatiedot

Kokonaisuudessaan kyselyyn vastasi 153 ihmistä. Lopulliseen otokseen karsittiin yksitoista vastausta, jonka jälkeen vastauksia jäi jäljelle 142. Lopullisesta otoksesta karsitut vastaukset olivat joko tyhjiä tai ne koettiin asiattomiksi. Kyselyn ensimmäiset kysymykset, eli kysymykset 1-5 olivat taustatietoihin liittyviä kysymyksiä. Taustatietojen kysymyksistä ensimmäinen tiedosteli vastaajalta, työskentelivätkö he yksityisellä vai julkisella sektorilla, vaiko eivätkö työskentele kummallakaan. Vastaajia kysymykseen oli 140. Näistä 140:stä vastaajasta 111 eli 79,3 % toimi yksityisellä sektorilla. Vastaavasti julkisella sektorilla toimi 23 vastaajaa eli 16,4 %. Loput 6 (4,3 %) olivat vastanneet etteivät toimi kummallakaan.

Seuraavassa monivalintakysymyksessä kysyttiin, missä kaikissa työrooleissa henkilö työskentelee. Vastaajat, joita oli 141, olivat vastanneet tähän kohtaan jonkin vaihtoehdoista ja valittuja vaihtoehtoja oli yhteensä 343, eli keskiarvoisesti yksi vastaaja valitsi noin 2,4 vaihtoehtoa (Taulukko 1, sivu 29).

Monivalinnassa Muu -kohtaan oli mahdollista kirjoittaa oma vastaus. Kohtaan vastasi 16 henkilöä (Taulukko 2, sivu 30).

Kolmannessa kysymyksessä kysyttiin, käyttääkö vastaaja kuorta työssään. Kysymykseen

Rooli	n	%
Avoimen lähdekoodin ohjelmistojen kehittäjä	31	22
Ohjelmistoarkkitehti	49	34,8
Ohjelmistokehittäjä	74	52,5
Ohjelmistotestaaja	11	7,8
Opettaja	14	9,9
Suunnittelija	32	22,7
Tietotekniikan harrastaja	63	44,7
Ylläpitäjä	53	37,6
Muu	16	11,3

Taulukko 1. Vastaajien vastaamat roolit

vastasi 137 henkilöä. Heistä 97,1 % käytti kuorta ja loput 2,9 % ei käyttänyt.

Neljännessä monivalintakysymyksessäpyydettiin luettelemaan ne kuoret, joita vastaaja käyttää työssään. Vastaajia oli 138, ja vastauksia oli yhteensä 244, eli keskiarvoisesti yksi vastaaja oli valinnut 1,8 kuorta (Taulukko 3, sivu 31).

Seuraavaksi vastaajaa pyydettiin arvioimaan omaa kuoren käytön osaamista asteikolla 1-5 erittäin kokemattomasta erittäin kokeneeseen. Vastaajia oli 142. Suurin osa vastaajista (35,9 %) olivat arvioineet omaksi kuoren kokeneisuudekseensa 4. Toiseksi suurin osa (29,6 %) oli valinnut kaikkein korkeimman, eli 5 (Taulukko 4, sivu 31). Vähiten vastauksia oli sen sijaan saanut kaikkein alin kokemustaso eli 1, johon oli vastannut vain 2,8 %. Kokemustasojen vastausten keskiarvo oli 3,8 ja mediaani oli 4.

Kysymys 6. Mitkä / Minkälaiset tehtävät kuoriohjelmoit (engl. shell-scripting)?

Kuudennessa kysymyksessä, eli ensimmäisessä avoimessa kysymyksessä kysyttiin, minkälaisiin tehtäviin vastaaja käyttää kuoriohjelmointia. Kysymykseen vastasi 127 henkilöä. Vastauksissa oli havaittavissa monimuotoisuutta, eikä mikään erityinen piirre noussut valtaosaksi. Noin 36:ssa 127:stä vastauksessa esiintyi toistettavien tai usein suoritettavien tehtävien tekeminen kuoriohjelmoinnilla. Toistuvien tehtävien lomassa oli myös isolla osalla vastauk-

Muu	Muu
Tutkimusasiantuntija	Tutkija
service desk	devops / test engineer
Järjestelmäarkkitehti	tietoturva-asiantuntija
Tekninen asiantuntija	helpdesk-asiakaspalvelija
IT huoltoteknikko	Sr. Dir, Cloud Engineering
DevOps	SCM/CI
Tutkija	Data engineer
Tietoliikenneinsinööri	HAKKERI

Taulukko 2. Muut roolit

sessa maininta tehtävien automatisoinnista. Mainintoja erilaisten tehtävien automatisoinnista oli noin 29. Tiedostojen käsittelyyn ja ylläpitoon liittyvät tehtävät näkyivät selvästituloksissa, ja kyseiseen kategoriaan luokiteltiin 14 esiintymää.

Muita huomattavia havaintoja olivat ylläpitoon ja asennuksiin viittaavat tehtävät, sekä maininnat kuoren käyttämisestä tehtäviin, jotka eivät olleet pieniä, mutteivat toisaalta myöskään liian suuria. Asennukset sai noin 11 mainintaa, ylläpito sai arviolta 10 mainintaa, ja niin sanotut keskikokoisten tehtävien suorittamiset saivat 11 tämän kaltaista vastausta.

Erillisiä vähemmän havaittuja esiintyviä olivat muun muassa putkitus, cron, docker, varmuuskopiointi, haut, käynnistykset, oneliner, pilvi, päivitykset ym. Vastaukset vaihtelivat tässä kysymyksessä niin runsaasti, ettei ihan kaikkia erillisiä mainittuja esiintymiä mainita tässä osiossa.

Kuudennessa kysymyksessä kysyttiin vastaajilta, minkälaisiin tehtäviin he käyttivät työssään kuoriohjelmointia. Vastauksissa ilmeni monimuotoisuutta: vastaajlla oli runsas määrä erilaisia tehtäviä, joihin he käyttivät kuoriohjelmointia. Yleiskatsauksessa tehtävissä ilmenivät pääasiassa erilaiset rutiinitehtävät, jotka vaativat toistamista tai automatisointia. Monet tehtävät luonnollisesti viittasivat eri järjestelmän eri toimintoihin, kuten asennuksiin, ylläpitoon ja tiedostojen käsittelyyn. Ilmeni myös, että kuoriohjelmointia suoritetaan tehtäviin, jotka eivät ole laajuudeltaan liian vaativia.

Kuori	n	%
Bash	123	89,1
C Shell	8	5,8
Korn Shell	4	2,9
PowerShell	39	28,3
fish	6	4,3
zsh	42	30,4
cmd	9	6,5
tsch	3	2,2
ash	8	5,8
Muut	5	3,6

Taulukko 3. Vastaajien vastaamat kuoret

Taso	n	%
5	42	29,6
4	51	35,9
3	37	26,1
2	8	5,6
1	4	2,8

Taulukko 4. Vastaajien itsearvioidut kuoren käytön osaamiset

Kysymys 7. Miksi käytät edellisessä kohdassa mainittuihin tehtäviin erityisesti kuoriohjelmointia?

Seitsemännessä kysymyksessä pyydettiin vastaajaa tarkentamaan syitä sille, minkä takia vastaajat käyttävät juuri kuoriohjelmointia edellisessä kysymyksessä esittämiinsä tehtäviin. Kysymykseen vastasi 127 henkilöä. Vastauksissa esiintyi useita teemoja. Kolme yleisimmin toistuvaa teemaa vastauksissa olivat nopeus, helppous, sekä siirrettävyys tai ympäristösitoutumattomuus. Nopeudesta löydettiin noin 41 mainintaa, siirrettävyyteen tai ympäristösitoutumattomuuteen liittyviä vastauksia oli noin 34 ja helppouteen liittyviä mainintoja löydettiin noin 32.

Neljästätoista viiteentoista mainintaa saaneita esiintymiä olivat automatisointi(14), tottumus tai ”laiskuus”(15) sekä vastaukset, jossa ilmaistiin yksittäisistä komennoista siirtyminen tai kokoaminen kokonaisiksi skripteiksi (myös 15 kappaletta). Muita esiintymiä olivat tehokkuus, käytännöllisyys, joustavuus tai muokattavuus, monipuolisuus, käyttövalmius, virheiden vähentäminen, yksinkertaisuus, varmuus, ei tarvetta kirjastoille, ei graafisuutta, Unix, syntaksin selkeys, ei vaihtoehtoja yms.

Seitsemännessä kysymyksessä kysyttiin syitä kuoren käyttöön edellisessä kysymyksessä mainituissa tehtävissä. Vastauksissa tuli ylivoimaisesti esille kolme ominaisuutta: helppous, no-

peus ja siirrettävyys. Siirrettävyyteen laskettiin vastaukset, joissa mainittiin kuoren olevan ympäristösitoutumaton ja toimivan kaikilla järjestelmillä ilman erillisiä vaatimuksia. Vastauksissa tuli esille myös skriptien kokoaminen vuorovaikutteisuuden avulla yksittäisistä komennoista, jota pidettiin yleisenä kuoriohjelmoinnin hyötynä. Monet viittasivat myös tottumukseensa tai omaan ”laiskuuteensa”. Kuitenkin suurin osa syistä viittasivat jollain tavalla kuoren sisäisiin ominaisuuksiin ja hyötyihin.

Kysymys 8. Mitkä toiminnot / ominaisuudet koette hyödyllisinä kuoriohjelmoinnissa?

Kahdeksannessa kysymyksessä tiedusteltiin vastaajalta, mitkä ominaisuudet he kokevat hyödyllisinä kuoriohjelmoinnissa. Kysymys sai 119 vastausta. Vastauksissa esiintyi useita ominaisuuksia, eikä mikään yksittäinen ominaisuus noussut merkittävästi muiden yli. Osa ominaisuuksista sai kuitenkin enemmän huomiota. Yleisimmät ominaisuudet, jotka saivat huomiota, olivat siirrettävyys tai ympäristöriippumattomuus, nopeus ja helppous. Siirrettävyyteen tai ympäristösitoutumattomuuteen liittyviä mainintoja esiintyi noin 21 kertaa, helppouteen liittyviä mainintoja vastauksissa esiintyi noin 19 kertaa ja nopeuteen viittaavia mainintoja esiintyi noin 15 kertaa. Tulosteen ohjaukseen tai tarkasteluun liittyviä vastauksia tuli arvioilta 12. Muita yleisiä hyödyllisiksi koettuja ominaisuuksia olivat erilaiset kielelliset ominaisuudet, kuten putkitus, joka mainittiin noin 12 kertaa ja silmukat, jotka mainittiin 11 kertaa. 10 vastaajaa olivat viitanneet kuoren ulkopuoliseen tukeen, kuten ratkaisujen löytämisen helppouden tai ulkopuoliset työkalut. Kymmenen vastausta olivat saaneet myös syntaksin selkeys sekä automatisointi kuoriohjelmoinnilla.

Muita vähemmän lueteltuja hyötyjä olivat yksinkertaisuus, keveys, tehokkuus, ohjelmoitavuus, ajastukset, liput, Unix, dokumentaatio, yhteydet, datan käsittely, Linux, erinäiset komennot, tiedostojen käsittely, ympäristömuuttujat, ehtorakenteet, muuttujat, monipuolisuus, muistettavuus ja opetettavuus, integroitavuus, sekä komentojen kokoaminen kokonaiseksi skripteiksi.

Kahdeksannessa kysymyksessä kysyttiin vastaajilta heidän hyödyllisiksi kokemiaan ominaisuuksia kuoriohjelmoinnissa. Vastauksissa nousi esille useita asioita, jotka koettiin hyödyllisiksi, eikä mikään aihepiiri saanut huomattavasti suurempaa kannatusta. Siirrettävyys tai

ympäristösitoutumattomuus nousi kaikkein hyödyllisimmäksi tekijäksi. Helppoutta ja nopeutta pidettiin myös yleisinä syinä. Monet kuoren ominaispiirteistä, kuten putkitus, syötteet ja vasteet erottuivat joukosta.

Kysymys 9. Mitkä tehtävät koette olevan hankalia tai epäkäytännöllisiä kuoriohjelmoinnissa?

Yhdeksännessä kysymyksessä taas kysyttiin, minkä vastaajat kokivat kuorissa hankalaksi tai epäkäytännölliseksi. Kysymykseen vastattiin 119 kertaa. Kolme asiaa nousi esille yli muiden. Suurien tai monimutkaista logiikkaa tai datan manipulaatiota vaativat tehtävät olivat yleisin hankaluuksien aihe monissa vastauksissa. Noin 27:ssä vastauksessa vastaajat olivat ilmoittaneet laajempien tai monimutkaisempien tehtävien olevan hankalia suorittaa kuoriohjelmoinnilla. Toiseksi yleisimpänä esiintyivät virheisiin, virheidenkäsittelyyn tai muuten debuggaukseen liittyvät maininnat vastauksissa, joihin viitattiin arviolta 24 vastauksessa. Kolmanneksi yleisimmäksi asiaksi nousi monien mukaan syntaksin ja joskus semantiikankin vaikeaselkoisuus tai sekavuus, jotka puolestaan mainittiin 23 kertaa.

Muita teemoja tai esiintymiä, joita havaittiin 10-14 kertaa olivat eri kuorten väliset eroavaisuudet (10 vastausta), merkkijonoihin tai tekstinkäsittelyyn liittyvät hankaluudet (11 vastausta) sekä aritmeettiset hankaluudet (10 vastausta). Neljässätoista eri vastauksissa havaittiin vastaajien vertaavan kuoren rajoittuneisuutta varsinaisiin ohjelmointikieliin. Muita vähemmän esiintyviä kuoren heikkouksia vastauksissa olivat muuttujat, verkkoliikenteen haastavuudet, poikkeustapaukset, tiedostonimet, silmukat, funktiot, tyypitys, kirjastot, ehdot ja kuvankäsittely.

Kysymyksessä kysyttiin, mitkä seikat käyttäjät kokivat kuoriohjelmoinnissa haastavana tai epäkäytännöllisenä. Kolme epäkäytännöllistä asiaa nousi merkittävästi esille: syntaksi, virhealttius ja suuremman logiikan tai skaalaltaan laajempien toteutusten hankaluus. Tietyt operaatiot, kuten aritmeettiset laskutoimitukset ja merkkijonojen käsittely tuottivat myös hankaluuksia vastausten mukaan. Eroavaisuudet kuorten välillä nousivat myös esille. Monet olivat myös vertailleet ohjelmointikieliä ja kuoria.

Kysymys 10. Haluaisitko käyttää kuoriohjelmointia tehtäviin, joihin et tällä hetkellä käytä sitä?

Kymmenessä kysymyksessä kysyttiin, haluaisiko vastaaja käyttää kuoriohjelmointia tehtäviin, joihin hän ei vielä käytä sitä. Kysymys sai 117 vastausta. Kysymyksestä ei poimittu erillisiä mainintoja tai teemoja, vaan se lajiteltiin vastauskohtaisesti kokonaisuudessaan. Kieltävästi olivat vastanneet 117:sta vastaajasta 67 vastaajaa eli 57,3 %. Vain 15 henkilöä eli 12,8 %, olivat vastanneet myöntävästi. Vastauksissa 18 vastaajaa eli 15,4 %:a eivät olleet ilmaisseet suoraviivaisesti kieltäytymistä tai myöntymistä, vaan kuorelle kerrottiin olevan oma paikkansa ja käyttötarkoituksensa, tai vastaajat ilmaisivat pystyvänsä käyttämään vapaasti haluamiaan työkaluja. Kolme vastaajaa ilmaisivat haluavansa lisää ominaisuuksia kuoreen, tai halua vaihtaa kuorta toiseen, jotta sitä pystyisi käyttämään useampaan tehtävään (Taulukko 5, sivu 34).

Kohdan 10 kysymyksessä kysyttiin vastaajien kiinnostusta käyttää kuorta muuhunkin kuin olemassaoleviin käyttötarkoituksiin. Yli puolet vastanneista olivat vastanneet kieltävästi, ja moni vastaus ilmaisi kuoren löytäneen oman paikkansa. Myöntäviä vastauksia oli vähän.

Vastaus	n	%
En	67	57,3
Kyllä	15	12,8
En tiedä	6	5,1
Ehkä	2	1,7
Tarpeen mukaan?	18	15,4
Kuoren parannus?	3	2,6
En ymmärrä kysymystä	1	0,9
Muut	5	4,3

Taulukko 5. Haluaisitko käyttää kuoriohjelmointia tehtäviin, joihin et tällä hetkellä käytä sitä?

Kysymys 11. Käyttäisitkö kuoriskriptien sijasta jotain muuta kieltä tai teknologiaa?

Yhdennessätoista kysymyksessä vastaajilta kysyttiin, käyttäisivätkö he jotain muuta kieltä tai teknologiaa kuoriohjelmoinnin sijaan. Kysymykseen vastattiin 123 kertaa. Kysymyksistä ei aluksi poimittu erillisiä mainintoja tai teemoja, vaan ne lajiteltiin vastauskohtaisesti kokonaisuudessaan. Ehdotetut kielet tai teknologiat laskettiin mainintakohtaisesti. Kielteisesti suoraan vastasi 17 vastaajaa 123:sta ja taas 13 vastaajaa vastasivat suoraan myönteisesti. Kuitenkin 60 henkilöä, eli 48,8 % vastaajista oli kertonut yhden tai useamman vaihtoehtoisen teknologian vastauksessaan (Taulukko 6, sivu 35). Yhteensä 23 eri teknologiaa esiintyi vastauksissa: *Python*, *Ansible*, *Perl*, *C*, *JavaScript*, *Node.js*, *Jenkins*, *PHP*, *Go*, *Ruby*, *Rust*, *Saltstack*, *C#*, *C++*, *Terraform*, *Java*, *Scala*, *Julia*, *TLC*, *Groovy*, *R*, *F#* ja *Clojure* (Taulukko 7, sivu 36). Yleisimmin mainittu vaihtoehto oli Python, joka mainittiin 44 kertaa.

Yhdennessätoista kysymyksessä pyydettiin vastaajaa kertomaan, käyttäisikö tämä jotain muuta kieltä tai teknologiaa kuoren sijasta. Suurin osa oli antanut oman ehdotuksensa, ja osa vastasi myöntävästi. Osa vastaajista ilmaisi myös olevansa avoin uusille ideoille.

Vastaus	n	%
Ehdotus	60	48,8
En	17	13,8
Kyllä	13	10,6
Ehkä	2	1,6
En tiedä	1	0,8
Tarpeen mukaan	10	8,1
Jos olisi	10	8,1
Käytän jo	10	8,1

Taulukko 6. Käyttäisitkö kuoriskriptien sijasta jotain muuta kieltä tai teknologiaa?

Kysymys 12. Mihin / minkälaisiin tehtäviin käytät interaktiivista kuorta?

Kahdennessatoista kysymyksessä siirryttiin vuorovaikutteista kuorta koskeviin kysymyksiin. Kuudennen kysymyksen tavoin kahdennessatoista kohdassa kysyttiin, mihin tehtäviin vastaaja käyttää vuorovaikutteista kuorta. Kysymykseen vastasi 120 vastaajaa. Vastaajien kes-

Ehdotettu teknologia/kieli	n
Python	44
Ansible	8
Perl	7
C	6
JavaScript	5
Node.js	4
Jenkins	3
PHP	3
Go	2
Ruby	2
Rust	2
Saltstack	2

Ehdotettu teknologia/kieli	n
C#	2
C++	2
Terraform	2
Java	2
Scala	1
Julia	1
TLC	1
Groovy	1
R	1
F#	1
Clojure	1

Taulukko 7. Vastaajien ehdottamat vastineet kuoriohjelmoinnille

kuudessa yleisin vastaus oli vuorovaikutteisen kuoren käyttö yleiseen tietokoneen käyttöön, tai ”kaikkeen”, jota esiintyi 32 kertaa. Tiedostojen käsittely oli toiseksi yleisin vastaus, ja se esiintyi 31 kertaa. Kolmanneksi yleisin vastaus oli palvelimien ylläpito, jonka mainitsi 24 vastaajaa.

Muitakin tehtäväalueita mainittiin kyselyssä. Välillä 10-15 vastausta saaneita teemoja olivat yhteyksiin ja versionhallintaan liittyvät, joista kummatkin tehtävät saivat erikseen 15 vastausta. Vastauksessa 12 vastaajaa olivat ilmaisseet käyttävänsä kuorta yksittäisiin komentoihin tai maininneet suoraan käyttämänsä komennon (esim. grep). Myös kansionhallintaan tai navigointiin liittyvät tehtävät saivat 12 mainintaa. Vianetsintään tai muuhun järjestelmän tilan tarkkailuun liittyviä vastauksia oli 11. Ohjelmien ajaminen esiintyi 10:ssä vastauksessa. Muita kyselyyn vastaajien luettelemia tehtävätyyppejä olivat testaus, asennukset, prosessit, käynnistykset, lokit, datankäsittely, tekstin editointi, docker, tietokannat, skriptit ja pilvi.

Kahdennessatoista kysymyksessä kysyttiin, minkälaisiin tehtäviin vastaaja käyttää vuorovaikutteista kuorta. Vastauksissa tuli esille useita tehtäviä tai tehtävätyyppejä, mutta kolmen havaittiin esiintyvän muita enemmän. Moni oli ilmaissut käyttävänsä vuorovaikutteista kuorta

kaikkeen tietokoneen käyttöön. Muita yleisiä käyttötarkoituksia olivat ylläpito ja tiedostojen käsittely.

Kysymys 13. Miksi käytät edellisessä kohdassa mainittuihin tehtäviin erityisesti kuorta?

Kolmannessatoista kysymyksessä kysyttiin, miksi vastaaja käyttää vuorovaikutteista kuorta. Kysymys sai 116 vastausta. Noin 44 vastausta sisälsi maininnan parempien vaihtoehtojen puutteesta tai graafisten käyttöliittymien epäkäytännöllisyydestä verrattuna kuoreen. Noin 24 vastaajaa vetosi vuorovaikutteisen kuoren nopeuteen syynä sen käytölle.

Vähemmän havaittuihin käytön syihin nousi kolme vähemmän yleistä syytä. Yhteydenottoon liittyvät vastaukset, kuten kuoren ainutlaatuisuus tai sen asema parhaana vaihtoehtona saivat 11 vastausta. Tehokkuuteen viittavia vastauksia oli myös 11. Helppoudesta kertovia vastauksia oli noin 10. Muita harvemmin mainittuja syitä vuorovaikutteisen kuoren käyttöön olivat muun muassa Linux, siirrettävyys, vaihtoehtojen puute, varmuus, automaatio, virheilmoitukset ja testaaminen yleensä, tottumus, ulostulon hallinta, hiiren käytön puute, dokumentaatio, käytännöllisyys ja joustavuus.

Kolmannessatoista kysymyksissä kysyttiin, miksi vastaajat käyttivät vuorovaikutteista kuorta kahdennessatoista kysymyksessä kertomiinsa tehtäviin. Yleisimmäksi syyksi nousi se, ettei ole sopivaa graafista sovellusta tai muuta vaihtoehtoa kuoren käyttöliittymälle. Moni oli myös kokenut vuorovaikutteisen kuoren nopeaksi.

Kysymys 14. Mitkä toiminnot / ominaisuudet koet interaktiivisessa kuoressa hyödyllisinä?

Neljännessätoissa avoimessa kysymyksessä kysyttiin ominaisuuksia, jotka vastaajat kokivat hyödyllisiksi vuorovaikutteisessa kuoressa. Kysymykseen vastattiin 116 kertaa. Neljä yleisimmiksi noussutta havaintoa hyödyllisistä ominaisuuksista olivat komentohistoria, automaattinen tekstin täydennys, komentojen putkitus ja nopeus. Komentohistoria esiintyi vastauksissa 30 kertaa ja täydennys esiintyi 25:ssä vastauksessa. Putkitus/ketjutus oli 18 henkilön vastauksessa ja nopeus esiintyi 17 eri vastauksessa. Harvinaisempia havaintoja olivat

laajennettavuuteen liittyvät, joita oli noin 10. Vastaajat olivat myös luetelleet yksittäisiä komentoja, jotka he kokivat erityisen hyödyllisiksi. Noin 10 henkilöä mainitsivat vastauksessaan yhden tai useamman komennon. Erinäisiä harvinaisempia teemoja löytyi tähän vastauskohtaan useita. Varsinkin aikaisemmissa kysymyksissä esiintyviä aiheita, kuten tehokkuutta ja siirrettävyyttä esiintyi kyseisen kysymyksen vastauksissa. Tiedonohjauskin esiintyi osassa vastauksista. Monet olivat listanneet erinäisiä ohjelmointikielen ominaisuuksia, kuten silmukat ja muuttujat.

Neljännessätoista kysymyksessä kysyttiin, mitkä ominaisuudet vastaaja kokee hyödyllisiksi vuorovaikutteisessa kuoressa. Neljä ominaisuutta mainittiin muita useammin. Kaksi yleisintä olivat vuorovaikutteisen kuoren komentojen kirjoitusta suoraviivaistavat ominaisuudet, eli komentojen täydennys Tab-näppäimellä ja komentohistoria. Lisäksi putkitus oli näiden jälkeen yleinen. Nopeusmaininnat nousivat tässäkin vastauksessa esille.

Kysymys 15. Mitkä tehtävät koet olevan hankalia tai epäkäytännöllisiä interaktiivisessa kuoressa?

Viidennessätoista avoimessa kysymyksessä kysyttiin ominaisuuksia, jotka vastaajat kokivat hankalina tai epäkäytännölllisinä vuorovaikutteisessa kuoressa. Kysymykseen vastasi 102 vastaajaa. Vastausten teemat olivat jakaantuneet suhteellisen tasaisesti. Yleisimpiä teemoja, jotka nousivat esille olivat ongelmat isompien tai monimutkaisempien skriptien kanssa, joita esiintyi noin 15:ssa eri vastauksessa. Lisäksi mainittiin tiedostojen nimiin tai käsittelyyn liittyvät ongelmat, jotka mainittiin 13 vastauksessa. Esille nousi paljon myös graafisiin toimenpiteisiin, kuten kuvankäsittelyyn liittyviä ongelmia. Mainintoja graafisista tai visuaalisista toimenpiteistä esiintyi 11 kertaa. Näistä erillisenä kuvankäsittelyyn ja kuvien katseluun liittyviä heikkouksia esiintyi 10 kertaa. Verkkoselailusta ja verkkoyhteyksistä kertovia vastauksia oli 11. Yhdeksän vastaajaa kertoivat vuorovaikutteisen kuoren hankaluudeksi muistettavuuden tai opettavuuden ongelmat. Hakemistojen navigoinnin hankaluuksista olivat kertoneet seitsemän vastaajaa. Muita harvinaisempia hankaluuksia olivat rajallisuus, tekstieditorit, videoiden editointi tai katselu, virheilmoitukset tai debuggaus, JSON, syntaksi, Git, asetukset. Kahdeksan vastaajaa kertoivat myös, etteivät he koe vuorovaikutteisessa kuoressa olevan mitään hankaluuksia.

Viidennessätoista kysymyksessä kysyttiin vuorovaikutteisen kuoren epäkäytännöllisyyksiä tai hankaluuksia. Yleisimmin ilmi tullut asia oli monimutkaisten tehtävien suorittaminen vuorovaikutteisella kuorella. Monet viittasivat erilaisiin graafisiin operaatioihin, kuten vastaajien kuoren vertaaminen graafisiin käyttöliittymiin tai kuoren vertaamiseen graafiin käyttöliittymiin tai videoiden katsomiseen. Visuaalisuuteen liittyvien vastausten lisäksi tiedostojen nimiin liittyvät hankaluudet koettiin yleisiksi.

Kysymys 16. Haluaisitko käyttää interaktiivista kuorta tehtäviin, joihin et tällä hetkellä käytä kuorta?

Kuudennessatoista avoimessa kysymyksessä kysyttiin, haluaisivatko vastaajat käyttää vuorovaikutteista kuorta tehtäviin, joihin he eivät tällä hetkellä käytä sitä. Vastauksia tuli 105. Miltei puolet vastanneista vastasivat kielteisesti. Suoria myönteisiä vastauksia, joita ei tarkennettu oli noin viisitoista sadasta vastaajasta. Yleisimmäksi tarkennetuksi asiaksi nousivat verkkoon liittyvät tehtävät tai asiat (Taulukko 8, sivu 39).

Vastaus	n	%
En	47	44,8
Kyllä	17	16,2
En tiedä	6	5,7
Ehkä	5	4,8
Käytän jo kaikkeen	7	6,7
Ehdotus?	15	14,3
Osaisinpa?	4	3,8
Muut	4	3,8

Taulukko 8. Haluaisitko käyttää interaktiivista kuorta tehtäviin, joihin et tällä hetkellä käytä kuorta?

Kysymys 17. Käyttäisitkö interaktiivisen kuoren sijasta edellisen kohdan tehtäviin jotain muuta kuin interaktiivista kuorta?

Seitsemännessätoista kysymyksessä kysyttiin vielä, haluaisiko vastaaja käyttää vuorovaikutteisen kuoren sijaista jotain toista teknologiaa. Vastauksia oli 97. Vastaajista 32, eli noin kolmasosa vastasi kieltävästi ja vain 13 vastasi myönteisesti. Vastaajista 34 olivat ilmaisseet joko käyttävänsä tai haluavansa käyttää jotain muuta teknologiaa vuorovaikutteisen kuoren tehtäviin sen sijaan. Graafinen käyttöliittymä vastattiin 15 kertaa (Taulukko 9, sivu 40).

Vastaus	n	%
En	32	33
Kyllä	13	13,4
En tiedä	6	6,2
Ehkä	1	1
Riippuu	4	4,1
Ehdotus?	34	35,1
En ymmärrä kysymystä	2	2,1
Muut	5	5,2

Taulukko 9. Käyttäisitkö interaktiivisen kuoren sijasta edellisen kohdan tehtäviin jotain muuta kuin interaktiivista kuorta?

Vastausten yhteenveto

Aiemmissa kohdissa katsottiin tuloksia yksittäisten vastausten kohdalla. Tietyt teemat kuitenkin nousevat esiin kaikista vastauksista. Valtaosa vastauksista käsitteli kuoren sisäisiä ominaisuuksia niin hyvässä kuin huonossa mielessä, mutta suurin osa antoi kokonaisuudessaan positiivisen vaikutelman. Lähes kaikissa kuoren hyötyihin kohdistuvissa kysymyksissä kolme kuoriohjelmointiin, ja vähemmissä määrin vuorovaikutteiseen kuoreen, liittyvää ominaisuutta korostui: *helppous* (Taulukko 10, sivu 41), *nopeus* (Taulukko 11, sivu 41) ja *siirrettävyys* (Taulukko 12, sivu 41).

Helppouden käsite esiintyi niin kuoriohjelmoinnin kysymyksissä kuin myös vuorovaikutteisen kuoren kysymyksissä, tosin useammin kuoriohjelmointiin liittyvissä kysymyksissä.

Vastaus	n
Kysymys 7	32
Kysymys 8	19
Kysymys 13	10
Kysymys 14	1
Yhteensä	62

Taulukko 10. Helppous

Vastaus	n
Kysymys 7	41
Kysymys 8	15
Kysymys 13	24
Kysymys 14	17
Yhteensä	97

Taulukko 11. Nopeus

Vastaus	n
Kysymys 7	34
Kysymys 8	21
Kysymys 14	6
Yhteensä	61

Taulukko 12. Siirrettävyys

Vastaus	n
Kysymys 6	29
Kysymys 7	14
Kysymys 8	10
Kysymys 13	9
Kysymys 14	4
Yhteensä	66

Taulukko 13. Automaatio

Nopeus esiintyi myös kummassakin kyselyn puoliskoissa.

Siirrettävyys esiintyi eniten kuoriohjelmoinnin kysymyksissä, tosin sitä esiintyi myös pienissä määrin vuorovaikutteisen kuoren osiossa.

Helppouden, nopeuden ja siirrettävyyden lisäksi koko kyselyssä ilmeni muitakin yleisiä teemoja. Automatisointiin ja automaatioon liittyviä vastauksia esiintyi paljon kuoriohjelmoinnin osiossa. Automaatio esiintyi sen lisäksi myös vuorovaikutteisen kuoren kysymysten vastauksissa, mutta vuorovaikutteisen kuoren kysymyksissä automaatio yhdistettiin toistettavien tehtävien ja toistoon liittyvien vastausten kanssa yhdeksi luokitelmaksi. Samalla automaation lisäksi toistuvuuteen viittavia vastauksia esiintyi usein samoissa yhteyksissä. Toistuvuus esiintyi kyselyn kummassakin osiossa (Taulukko 13, sivu 41).

Eri puolilla kyselyä esiintyi vastaajien mainintoja ohjelmointikielistä, mikä näkyi erityisesti kuoren vertailuna ohjelmointikieliin. Ylivoimaisesti yleisin mainittu kieli oli Python. Python-sana esiintyy vastauksissa noin 100 kertaa. Luonnollisesti suurin osa Pythonin-mainin-

noista löytyi vaihtoehtoisten teknologioiden kysymyksistä. Siitä huolimatta Python esiintyi muissakin vastauksissa. Osa Pythonin-maininnoista oli kuitenkin Pythonin puutteiden kritiikkiä.

Unix-kuoresta oletuksena löytyvät työkalut AWK ja sed löytyivät useasta vastauksesta. Sekä AWK että sed mainittiin erikseen yhteensä 15 kertaa. AWK:n ja sedin lisäksi grep-työkalukomento esiintyi laajalti vastauksissa. Grep esiintyi koko kyselyssä yhteensä 20 kertaa, eli enemmän kuin AWK ja sed.

6 Tulosten pohdinta

Tutkimuksen tavoitteena oli ottaa selvää, miksi yritysten sisällä käytetään kuoria. Teoriaosiossa käytiin läpi kuoren käsitettä, sen asemaa ohjelmoinnissa, aiempaa tutkimusta aiheesta ja esiteltiin vaihtoehtoja kuorille. Lisäksi käytiin läpi teknologian adaptaatioon liittyviä käsitteitä. Tässä luvussa tarkastellaan saatuja tuloksia ja pohditaan niiden merkitystä suhteessa teoriaan.

6.1 Kuoren tehtävät

Avoimien kysymysten ensimmäisen osion ensimmäinen kysymys pyysi vastaajia kertomaan, mihin he käyttävät kuoriohjelmointia työssään, ja samaan tapaan toisen osion ensimmäinen kysymys pyysi vastauksia vuorovaikutteisen kuoren tehtäviin. Kuoriohjelmoinnin kohdalla mainittiin kirjava määrä erilaisia tehtäviä, joihin vastaajat käyttivät kuoriohjelmia.

Kuoren ja varsinkin kuoriohjelmoinnin käyttötarkoituksista yleisimmäksi nousivat toistuvat tehtävät. Toinen yleisesti mainittu tehtävä oli automatisointi. Nämä tehtävät eivät ole missään mielessä yllättäviä, sillä automaatio ja muunlainen rutiinien suoraviivaistaminen on yksi kuoren erikoisaloista. Vasta viime aikoina muitakin automaatiotyökaluja, kuten Ansible, Jenkins tai Trraform on alkanut ilmaantua enemmän. Kuori on silti tähän asti toiminut luonnollisimpana vaihtoehtona johtuen sen pitkästä historiasta ja saatavuudesta.

Kuoren tehtäville esiintyi monenkaltaisia eri perustehtäviä automaatioon ja toistuvuuteen nojaavia perustehtäviä, kuten asennukset ja tiedostojen käsittely. Kuori toimii ohjelmistojen välillä eräänlaisena välittäjänä, ja tämä näkyi useissa vastauksissa, joissa kuorta esimerkiksi kutsuttiin ”liimaksi”, ”purukumiksi” tai ”linkkuveitseksi”.

6.2 Sisäiset syyt

Kyselyssä kartoitettiin vastaajien syitä käyttää kuorta työssään ja mitkä ominaisuudet kuoressa he kokivat hyödyllisiksi. Minkä tahansa kielen tai teknologian ominaisuudet jaetaan usein sisäisiin ja ulkoisiin ominaisuuksiin tai tekijöihin. Sisäisiin ominaisuuksiin luokitellaan

ominaisuudet, jotka ovat itsessään ja luontaisesti löydettävissä itse kielestä tai teknologiasta, kun taas ulkoiset ominaisuudet viittaavat enemmän sosiaalisiin vaikuttajiin tai teknologian yhteydestä muihin teknologioihin tai vaikuttajiin. Vastaajien syyt käyttää kuorta viittasivat enimmäkseen kuoren sisäisiin ominaisuuksiin.

Tuloksissa yhdeksi yleisimmistä sisäisistä syistä kuoren käyttöön ja sen hyötyihin nousi helppous. Helppous nousi esille joko vastaajien ilmoittaessa, mitä he kokivat helpoksi kuoressa tai sitten he olivat ilmaisseet helppouden vain kuoren hyödyksi ilman täsmällisempää selvennystä. Yksi TAM-mallin päämuuttujista, joka esiintyy keskeisessä osassa jokaisessa TAM-mallissa, on *käytön koettu helppous*. Myöhempiin malleihin (TAM2 ja TAM3) on lisätty ulkoisia muuttujia, joiden on koettu vaikuttavan mallin päämuuttujiin. *Käytön koettuun helppouteen* on viimeisimmässä TAM3-mallissa on lisättynä ankkurimuuttujia ja sopeutumisen muuttujia. Ankkuroinnin muuttujat ovat TAM3-mallissa *itseluottamus omiin tietokoneen käyttökykyihin*, *yksilön oma vaikutelma ulkoisesta tuesta*, *huoli tietokoneiden käytöstä*, sekä *leikkisyys*. Sopeutumisen muuttujia ovat taas *havaittu käytön nautinto* ja *objektiivinen käytännöllisyys*. *The Technology Acceptance Model: Past, Present, and Future* -niminen artikkeli tutki TAM-mallin ja sen ulkoisten muuttujien yhteyksien merkittävyyksiä. Kyseisessä tutkimuksessa käytetyn TAM-mallin ulkoiset muuttujat erosivat jonkin verran muiden edeltävien TAM-mallien ulkoisista muuttujista. *Havaittuun käytön helppouteen* oli merkitty kymmenen eri yhteyttä, joista viisi oli koettu merkittäväksi, kolme sekalaiseksi ja kaksi viimeistä merkityksettömiksi. Nämä merkityksettömät ulkoisten muuttujien yhteydet havaittuun käytön helppouteen olivat *huoli tietokoneiden käytöstä* ja *käyttöönottoa edistävät olosuhteet*. Sekalaisen merkityksen muuttujat taas olivat *käytettävyys*, *loppukäyttäjän tuki / itseluottamus* ja *asenne*. Varsinaisesti merkittäviksi todetut muuttujien väliset suhteet olivat *saavutettavuus*, *leikkisyys*, *havaittu nautinnollisuus*, *yhteisöllinen vaikutus* ja *johtotuki / kokemus*. (Lee, Kozar ja Larsen 2003.)

Saavutettavuudella tarkoitetaan kahta eri tyyppistä saavutettavuutta: fyysistä ja informatiivistä. Fyysinen saavutettavuus tarkoittaa yksilön pääsyä fyysisesti käsiksi laitteistoon, joka mahdollistaa järjestelmän käytön. Informatiivinen saavutettavuus on kyky kerätä haluttua tietoa järjestelmästä. Kuorten fyysistä saavutettavuutta voidaan havainnoida POSIX:n perusteella. Unix-kuoret kuuluvat jokaiseen POSIXin mukaiseen järjestelmään, joten kuoret ovat

saavutettavana lähes kaikista POSIX- ja Unix-ympäristöistä. Lisäksi siirrettävyys oli yksi kolmesta yleisimmistä havainnoista läpi kaikkien kyselyn vastausten kesken. (Karahanna ja Limayem 2000.)

Leikkisyydellä viitataan kognitiiviseen spontaanisuuteen tietokoneen kanssa vuorovaikuttaessa. *Havaittu nautinnollisuus* on käytön aikaista nautittavuutta itsessään, huolimatta mahdollisista käytön seurauksista ja tuloksista. Vuorovaikutteisen kuoreen liittyvissä kysymyksissä ilmeni paljon vastauksia, joissa vastaajat olivat ilmaisseet käyttävänsä kuorta kaikkeen tietokonetta käyttäessään. Moni oli myös ilmaissut samalla, ettei koe vuorovaikutteisessa etteivät koe vuorovaikutteisessa kuoressa olevan minkään sortin hankaluuksia tai epäkäytännöllisyyksiä. Toisekseen useassa vastauksessa ilmaistiin halua käyttää vuorovaikutteista kuorta tietokoneen perustoimintoihin, joita ei toistaiseksi pysty kunnolla hoitamaan kuorella, kuten kuvien ja videoiden katseluun. Kuoren laajan käytön ja halun käyttää sitä voidaan katsoa esittävän tietokoneen käytön leikkisyyttä tai havaittua nautinnollisuutta, jonka on monissa tutkimuksissa todettu vaikuttavan havaittuun käytön helppouteen (Karahanna ja Straub 1999, Webster ja Martocchio 1992).

Yhteisöllinen vaikutus, yhdistettynä yhteisön normeihin sekä yhteisöllisiin paineisiin merkitsevät yksilön käsitystä siitä, mitä hänelle tärkeiden ja merkittävimpien henkilöiden mielestä hänen täytyisi tehdä ja mitä ei (Lee, Kozar ja Larsen 2003).

Johtotuki on johtajien järjestelmälle kohdennettua tukea resursseilla, mikä mahdollistaa suotuisamman ympäristön tietojärjestelmän menestykselle (Igbaria ym. 1997). *Kokemuksella* luonnollisesti tarkoitetaan aiempaa kokemusta aiheesta. Kyseisessä mallissa johtotuki ja kokemus oli yhdistetty samaksi muuttujaksi. Kyselyn taustatieto-osiossa oli pyydetty vastaajia arvioimaan omaa kuoriosaamistaan. Suurin osa vastaajista oli arvioinut oman osaamisensa 4-tasolle ja toiseksi suurin osa 5-tasolle. Pienin osa oli arvioinut oman osaamisensa 1-tasolle, ja toiseksi pienin 2-tasolle. Kysymyksen vastausten mediaani oli 4. Vastaajia voidaan näin ollen sanoa huomattavasti kokeneemmiksi kuoren käyttäjiksi. Vastaajien korkean kokemustason voidaan hyvin katsoa olevan yksi vaikuttavista muuttujista heidän vastauksissaan esiintyvään *havaittuun helppouteen*.

Nopeus oli yleinen sisäiseksi ominaisuudeksi luokiteltu ominaisuus, joka saanut paljon mai-

nintoja vastaajilta. Nopeuteen laskettiin mukaan niin yleinen suorasti vastattu ”nopeus” kuin vastaukset, jotka erittelivät, mikä on nopeaa tehdä tai minkä asian vastaajat osasivat tehdä kuorella. Vuonna 2021 julkaistussa tutkimusartikkelissa *Unix Shell Programming: The Next 50 Years* lueteltiin Unix-kuoren vahvuuksia ja heikkouksia (Greenberg, Kallas ja Vasilakis 2021). Yksi artikkelissa mainituista kuoren vahvuuksista oli kuoren suhteellisen tehokas suorituskyky. Tämän tehokkuutensa kuori saavuttaa tietovoiden käsittelyllä, joille löytyy kuoresta oma sovellusaluekielensä. Komentoputkitetun tietovoiden käsittelykykynsä ansiosta kuori voi myös saavuttaa tehokkaan suorituskyvyn aluslaitteesta liikaa riippumatta. Toisaalta Greenbergin mukaan yksi kuoren mahdollisesti korjattavissa oleviin heikkouksiin kuului myös *tarpeettomat uudelleenlaskennat*, joihin Greenberg väittää kuluvan liikaa aikaa. Kuitenkaan vastauksista ei kyseistä heikkoutta löytynyt, minkä lisäksi *nopeuden* korostuminen vastauksissa vaikuttaisi olevan jokseenkin ristiriidassa artikkelin väittämän kanssa.

Suhteellisen epämääräisesti määriteltyjen sisäisten ominaisuuksien, nopeuden ja helppouden lisäksi siirrettävyys oli yleinen suosittu ominaisuus kuoressa, mikä voidaan kokea selkeäksi käsitteeksi. Siirrettävyydeksi lasketaan tutkimuksessa myös sellaiset käsitteet, kuten laiteriippumattomuus tai ympäristösitoutumattomuus. Siirrettävyydeksi mielletään kuoren löytyminen jokaisesta Unix-järjestelmästä tai Unix-kaltaisesta järjestelmästä, minkä takia sen kuoriohjelmia voidaan siirtää koneelta koneelle käytettäviksi.

Osassa vastauksista oli myös tullut esille, että kuoren kyky koostaa pienemmistä komennoista hiljalleen pidempiä kuoriskriptejä on hyödyllinen ominaisuus. Oletettavasti tämän voi päätellä kertovan kuoren vuorovaikutteisuuden ominaisuudesta, joka yksi kuoren tärkeimmistä ominaisuuksista.

6.3 Ulkoiset syyt

Vaikka siirrettävyys voidaan kokea sisäiseksi ominaisuudeksi, siinä on olemassa myös ulkoinen tekijä; loppujen lopuksi kuoren siirrettävyys ja laiteriippumattomuus johtuu POSIX:iin mukautumisesta. Kuori löytyy jokaiselta Unixin kaltaisesta käyttöjärjestelmältä, eikä se vaadi erillistä asentamista eri koneille. Tämä on yksi kuoriohjelmien saavuttaman siirrettävyyden syistä.

Vuonna 2005 tehdyssä *An empirical study of programming language trends*-nimisessä tutkimuksessa laadittiin kielten yleistymistä ennustavaa mallia. Tutkimuksessa aluksi tarkasteltiin sen ajan kielten suuntauksia arvioimalla kielten sisäisten ja ulkoisten tekijöiden merkittävyyttä kielen suosioon nähden. Sisäisistä tekijöistä, eli kielen ominaisuuksista, tärkeimmäksi ominaisuudeksi osoittautui laiteriippumattomuus. (Chen ym. 2005.)

Kuoren siirrettävyyden ominaisuus esiintyy myös *Unix Shell Programming: The Next 50 Years* -artikkelissa. Greenberg argumentoi artikkelissaan Unix-natiivisuuden olevan yksi kuoren olennaisimmista vahvuuksista, jonka takia myös kuoret ovat pysyneet yhtenä tietokoneen status quo -työkaluista. (Greenberg, Kallas ja Vasilakis 2021.)

Vaikka suurimmassa osassa vastauksia suhtautuminen kuoreen oli positiviinen, jonkin verran negatiiviselta vaikuttavaa suhtautumista oli havaittavissa. Joissakin vastauksissa kerrottiin kuoren käyttämisestä vaihtoehtojen puutteen vuoksi. Tämä voidaan myös mieltää kuoren kokemiseksi ylivoimaisesti parhaimmaksi ratkaisuksi.

6.4 Haittapuolet

Kyselyn kuoriohjelmointia ja vuorovaikutteista kuorta käsittelevissä osioissa oli kummassakin kysymys 9., jossa vastaajaa pyydettiin kertomaan, mitä ominaisuuksia he kokivat kuoriohjelmoinnissa tai vuorovaikutteisessa kuoressa epäkäytännöllisiksi tai hankaliksi. Kuoriohjelmoinnissa oli kolme eniten mainintoja kerännyttä heikkoutta. Nämä kolme ominaisuutta olivat skaalautumattomuus, virhealttius ja hankala syntaksi.

Skaalautumattomuuteen luokiteltiin vastaukset, joissa vastaaja oli vastannut kuoren heikkoudeksi liian suurten tai monimutkaisten operaatioiden suorittamisen. Greenbergin artikkelissa puhuttiin kuoren heikkouksista. Artikkeli oli luokitellut yhdeksi suurimmaksi kuoren heikkoudeksi sen skaalautumattomuuden. Kuori on pääsääntöisesti suunniteltu yksiytimisille asetuksille. (Greenberg, Kallas ja Vasilakis 2021.)

Kuoren virhealttiudesta oltiin kerrottu monissa vastauksissa, tai muihin virheisiin liittyvissä yhteyksissä, kuten virheiden etsimisessä ja löytämisessä (ns. debuggaaminen) sekä myös virheiden laatuun tai haitallisuuteen liittyvissä tapauksissa. Monet olivat vastauksissaan ker-

toneet kuoriohjelmoinnissa kuoren syntaksin olevan epäselvää tai hankalaa luettavaa. Yleisimmiksi kuoriksi näin vastanneiden keskuudessa ilmenivät Bash ja zsh, jotka olivat myös kaksi yleisintä kuorta kokonaisuudessaan. Kuitenkaan nämä kaksi eivät olleet ainoat mainitut kuoret. Greenberg listasi tutkimuksessaan kuoren virhealttiuden johtuvan osittain kuorien syntakseista (Greenberg, Kallas ja Vasilakis 2021). Osassa vastauksista oli löytynyt vastaajien yksilökohtaisia kertomuksia siitä, kuinka he tai heidän tuntemansa henkilöt olivat aiheuttaneet merkittävää vahinkoa kuoren komennoilla. Tämä tosin ei johdu pelkästään kuoren syntaksista, vaan myös sen virhealttiudesta ja kuoren tehokkuudesta.

6.5 Adaptaatio

Pieni, mutta huomattava osa vastauksista ilmaisi kuoren käytön syyksi oman tottumuksen tai ”laiskuuden”. Tottumukseen laskettiin myös vastaukset, jotka kertoivat yrityksessä olevien järjestelmien olevan vanhempaa perintöä. Alalla tällaiset vanhemmat perinnölliset järjestelmät ja koodit ovat yleensä haasteellisia korvata uusilla innovaatioilla. Tämä johtuu siitä, että vuosien varrella kertyneiden vanhempien koodien uudistamisesta tai kokonaan uudelleen laatimisesta kertyy liian suuria ajallisia ja taloudellisia kustannuksia yrityksille, eivätkä ne välttämättä kata vanhan perintökoodin tuottamia mahdollisia kustannuksia (Weide, Heym ja Hollingsworth 1995).

6.6 Mahdolliset ristiriidat

Empirical analysis of programming language adoption -artikkelin (Chen ym. 2005) mukaan tärkein asia, mihin pitäisi panostaa uuden kielen adaptaatiossa on valmiiden kirjastojen tai jossain määrin myös valmiin esimerkkikoodin saatavuus. Koko kyselyn aikana kirjastoihin liittyviä mainintoja oli vajaat kahdeksan. Osa oli maininnut kirjastoista riippumattomuuden yhdeksi kuoren hyödyistä, kun taas pari oli maininnut niiden puutteen kuoren heikkoudeksi. Toisaalta Greenberg argumentoi kuoren itsessään toimivan de facto -komponenttikirjastona enimmissä ohjelmissa.

Greenberg oli avannut artikkelissaan monia kuoren hyötyjä, hyödyistä johtuvia haittoja, sekä vielä niistä riippumattomia erillisiä ongelmia. Suurin osa Greenbergin tuomista kuoren omi-

naisuuksista tulivat suoraan tai epäsuorasti esille kyselyssä. Greenberg viittasi yhdeksi kuoren perusteellisemmaksi haitaksi sen liiallista dynaamisuutta, jonka Greenberg katsoi haittaavan staattisia analyyseja. Tässä tutkimuksessa toteutettu kysely oli pääsääntöisesti suunnattu tietotekniikan alalla työskenteleville henkilöille. Tästä syystä voidaan olettaa, ettei liikadynaamisen ongelma tullut esille vastauksissa, sillä siitä on suoranaista haittaa enimmäkseen vain kuoren tutkimuksessa. (Greenberg, Kallas ja Vasilakis 2021.)

Greenberg oli myös luonnehtinut kuoren erilliseksi ongelmaksi nykyaikaisten käyttöönottojen tuen puutteen, kuten aikaisemmassa luvussa mainittiin. Kyselyssä kysyttiin ensimmäisenä avoimena kysymyksenä, mihin vastaajat käyttivät kuorta työssään. Noin neljä vastaajaa oli maininnut kuoren käytön käyttöönottotehtävissä. Lisäksi pari käyttöönottoon liittyvää mainintaa löytyi muista vastauksista. Toisin kuin Greenbergin artikkelissa, jossa korostettiin käyttöönottojen ongelmallisuutta, käyttöönoton hankaluuksista ei löytynyt vastauksista paljoa kritiikkiä. Ainoastaan yksi vastaaja kommentoi yrityksensä siirtyneen pois kuoren käytöstä siinä. (Greenberg, Kallas ja Vasilakis 2021.)

6.7 Vaihtoehtoiset työkalut

Kyselyssä myös tiedusteltiin vastaajilta, haluaisivatko he käyttää kuorta enemmän vaiko vähemmän. Ihmiset, jotka halusivat käyttää kuorta useampiin tehtäviin tai muuten vaan useammin kuuluivat huomattavaan vähemmistöön. Moni oli vastannut kysymykseen kuoriohjelmoinnin löytäneen oman paikkansa IT -alalta, ja osa oli ilmaissut saavansa tai pystyvänsä käyttämään kuorta vapaasti oman tarpeensa mukaan. Monet positiivisesti kuoreen suhtautuvat vastaukset olivat vaihtoehtoisia esityksiä kuorelle.

Eri puolilla vastauksia vastaajat olivat ottaneet puheeksi Python-kielen. Enimmäkseen Python luonnollisesti esiintyi kysymyksessä, jossa kysyttiin, käyttäisikö vastaaja kuoren sijasta muuta teknologiaa tai kieltä tehtäviinsä. Python oli myös ylivoimaisesti yleisin vastaus kuoren vaihtoehtoisiin kieliin verrattuna; toiseksi yleisin mainittu vaihtoehto kuorelle: toiseksi yleisin Ansible mainittiin vain kahdeksassa vastauksessa verrattuna Pythonin 44:ään mainintaan. Python on suosittu kieli ohjelmointikielten joukossa ja Pythonin kokeminen vaihtoehtona kuorelle johtuu ainakin osittain siitä, että Pythonilla on kuoren kanssa monia samoja

ominaisuuksia, kuten esimerkiksi kummatkin ovat tulkattavia. Pythonille on myös erikseen rakennettu oma kuori, IPython (McKinney 2012).

Ansible-automaatiomoottori on työkalu, jolla pystyy suorittamaan monia tehtäviä, jotka suoritetaan yleensä kuorilla, kuten tehtävien automatisoinnit. Kyselyssä kuoren korvaavien työkalujen joukossa Ansible oli saanut jonkin verran näkyvyyttä.

Perl oli myös yksi hiukan yleinen ehdotus kyselyssä vaihtoehtona kuorille, ja muutenkin eri osissa kyselyä, tosin usein myös kriittisestä näkökulmasta. Perl on kuoren tapaan tulkattavissa oleva ohjelmointikieli.

6.8 Tulevaisuuden arviointi

vastauksista havaituista pääosin myönteisistä asenteista, tulevaisuutta arvioivasta teoriasta ja kuoren POSIX-tuesta päätelleen kuoret tulevat pysymään käytössä vielä pitkälle lähitulevaisuuteen. Lisäksi kuorista tehdään vielä paljon tutkimusta, ja kuoria kehitetään jatkuvasti eteenpäin erilaisilla lisäyksillä. Vaikuttaisi kuitenkin siltä, että joitain kuoren tehtävistä pystytään korvaamaan tulevaisuudessa ja nykyäänkin uusilla innovaatioilla, joista alan ihmiset ovat tulleet tietoisiksi ja heidän suhtautumisensa niihin vaikuttaisi olevan suhteellisen positiivinen. Näistä huolimatta kuoren asema vaikuttaisi pysyvän vähintäänkin tietokoneen ohjelmistoja yhdistävänä välityökaluna.

7 Yhteenveto

Tämän tutkimuksen tarkoituksena oli kartoittaa syitä kuoren käyttöön tietotekniikan toimialalla. Tutkielmassa määriteltiin kuoren käsitteitä, alustavaa teoriaa kuoren nykyiseen käyttöön, ja esiteltiin mahdollisia vaihtoehtoisia teknologioita tai kieliä kuoren potentiaaliseen korvaamiseen. Näitä käsitellen tarkoituksena oli etsiä vastausta tutkimuskysymykseen: Miksi yrityksissä käytetään yhä kuoria tai kuoriohjelmointia?

Tutkielmassa on neljä osaa: teoria, tutkimuksen toteutus, kyselyn aineiston tulokset ja lopuksi tulosten pohdintaosio. Teoriaosuudessa oli kaksi osiota. Ensimmäisessä teorian osiossa käytiin läpi kuoreen liittyvää kirjallisuutta kuoren ominaisuuksista, siihen kohdistuvasta hyvistä ja huonoista puolista, aikaisemmasta kuoren tulevaisuuden arvioinnista, sekä mahdollisista korvaavista vaihtoehtoisista innovaatioista. Toisessa teorian osiossa katseltiin teknologian adaptaatioon liittyvää kirjallisuutta ja käsitteitä, sekä perehdyttiin eri TAM-malleihin.

Tutkimuksen toisessa osiossa käytiin läpi, miten tutkimus toteutettiin. Tutkimuksen aineistonkeruu suoritettiin kyselyllä, jossa hyödynnettiin enimmäkseen avoimia kysymyksiä. Kysely jaettiin suomalaisilla tietotekniikan foorumeilla. Kysymykset analysoitiin perinteisellä sisällönanalyysillä.

Tutkimuksen kolmannessa osiossa käytiin aluksi läpi kyselyn toteuttamisen taustatiedot, jonka jälkeen käytiin kysely läpi kysymys kysymykseltä. Jokaista kysymystä tarkasteltiin perinteisen sisällönanalyysin tyyliin koodaamalla vastaukset ja tuomalla esille vastauksissa esiintyvät maininnat ja motiivit, ja niiden esiintyvyydet.

Tutkielman viimeisessä osiossa yhdistettiin kaksi edellistä osiota, teoria ja aineisto, ja verrattiin niitä keskenään. Aineistossa esiintyviä havaintoja tarkasteltiin aiemman teorian näkökulmasta verraten yhteensopivuuksia tai ristiriitoja niiden välillä. Kaikkein yleisimmiksi kuoren yleisen käytön syiksi nousivat esille *helppous*, *nopeus* ja *siirrettävyys*.

Teorian pohjalta arvioitiin kuoren havaittuun helppouteen vaikuttavia tekijöitä TAM-mallin muuttujien pohjalta. Vaikuttavat muuttujat olivat saavutettavuus, leikkisyys, havaittu nautinnollisuus, yhteisöllinen vaikutus ja johtotuki / kokemus. Yksi iso vaikuttava tekijä kuoressa

oli siirrettävyys. Kuoren POSIX-mukautuvuus ja Unix-natiivisuus tekevät kuorista laiteriippumattoman.

Tutkimuksen suorittamisessa kyselytutkimuksena on omat hyötynsä, mutta myös puutteensa. Kyselyssä kerätään aineistoa henkilönäkökulmasta, minkä takia vastaukset ovat yleensä yksilöiden havaintoja, joten takia vastauksissa esiintyvien väitteiden oikeellisuutta on hankala mitata. Tuloksissa esiintyvien havaintojen mittaamisen, kuten esimerkiksi yleisesti esiintyneiden *helppouden*, *nopeuden* ja *siirrettävyyden* voisi suorittaa erilaisilla tutkimusmenetelmillä. Varsinkin nopeus on objektiivisesti mitattavissa oleva ominaisuus, jonka määrällinen tutkiminen voisi tuoda esiin hyödyllisiä tuloksista kuorista ja jopa havainnollistaa nopeutta.

Tutkimuksessa selvisi erilaisia teknologioita ja kieliä, joita voidaan käyttää kuoren sijaan. Vastaajat eivät olleet pystyneet ilmaisemaan, minkä takia vaihtoehtoisiin teknologioihin siirtyminen ei ole ollut heille mahdollista. Olisi kiinnostavaa jatkossa tehdä isompaa mittausta siitä, millaisia kustannuksia vaihtoehtoisiin teknologioihin siirtymisistä voisi tulla. Sen lisäksi vielä tärkeämpää olisi mitata objektiivisesti, ovatko mitkään vaihtoehtoisista teknologioista tai kielistä ylipäätänsäkään kuoria kustannustehokkaampia työtehtävissään. Loppujen lopuksi pienemmätkin aikaa vievät toimenpiteet voivat aiheuttaa yritysmittakaavassa suuria kustannuksia.

Lähteet

Alam, MD Shamsh, MD Faizan ja Shams Tabres & DR TK Shaik. 2017. “Automization of Services Using Shell Script for Server Deployment”.

Babuji, Yadu, Anna Woodard, Zhuozhao Li, Daniel S Katz, Ben Clifford, Ian Foster, Michael Wilde ja Kyle Chard. 2019. “Scalable parallel programming in Python with Parsl”. Teoksessa *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning),* 1–8.

Babuji, Yadu N, Kyle Chard, Ian T Foster, Daniel S Katz, Mike Wilde, Anna Woodard ja Justin M Wozniak. 2018. “Parsl: Scalable Parallel Scripting in Python.” Teoksessa *IWSG.*

Chen, Yaofei, Rose Dios, Ali Mili, Lan Wu ja Kefei Wang. 2005. “An empirical study of programming language trends”. *IEEE software* 22 (3): 72–79.

Davis, Fred D, Richard P Bagozzi ja Paul R Warshaw. 1989. “User acceptance of computer technology: A comparison of two theoretical models”. *Management science* 35 (8): 982–1003.

Davis, Ian J, Mike Wexler, Cheng Zhang, Richard C Holt ja Theresa Weber. 2015. “Bash2py: A bash to Python translator”. Teoksessa *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER),* 508–511. IEEE.

Dougherty, Dale, ja Arnold Robbins. 1997. *sed & awk: UNIX Power Tools.* O’Reilly Media, Inc.

Gift, Noah, ja Jeremy M Jones. 2008. *Python for Unix and Linux system administration.* O’Reilly Media, Inc.

Greenberg, Michael. 2017. “Understanding the POSIX shell as a programming language”. *Off the Beaten Track.*

———. 2018. “Word expansion supports POSIX shell interactivity”. Teoksessa *Conference Companion of the 2nd International Conference on Art, Science, and Engineering of Programming,* 153–160.

Greenberg, Michael, Konstantinos Kallas ja Nikos Vasilakis. 2021. “Unix Shell Programming: The Next 50 Years”.

Hsieh, Hsiu-Fang, ja Sarah E Shannon. 2005. “Three approaches to qualitative content analysis”. *Qualitative health research* 15 (9): 1277–1288.

Igbaria, Magid, Nancy Zinatelli, Paul Cragg ja Angele LM Cavaye. 1997. “Personal computing acceptance factors in small firms: a structural equation model”. *MIS quarterly,* 279–305.

Karahanna, Elena, ja Moez Limayem. 2000. “E-mail and v-mail usage: Generalizing across technologies”. *Journal of organizational computing and electronic commerce* 10 (1): 49–66.

Karahanna, Elena, ja Detmar W Straub. 1999. “The psychological origins of perceived usefulness and ease-of-use”. *Information & management* 35 (4): 237–250.

Lee, Younghwa, Kenneth A Kozar ja Kai RT Larsen. 2003. “The technology acceptance model: Past, present, and future”. *Communications of the Association for information systems* 12 (1): 50.

Mary, C. 2015. “Shellshock attack on linux systems–bash”. *International Research Journal of Engineering and Technology* 2 (8): 1322–1325.

McKinney, Wes. 2012. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* "O’Reilly Media, Inc.".

Meyerovich, Leo A, ja Ariel S Rabkin. 2013. “Empirical analysis of programming language adoption”. Teoksessa *Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications,* 1–18.

Payette, Bruce. 2007. *Windows PowerShell in action.* John Wiley & Sons.

Prechelt, Lutz. 2003. “Are scripting languages any good? A validation of Perl, Python, Rexx, and Tcl against C, C++, and Java.” *Adv. Comput.* 57:205–270.

Raghavan, Deepti, Sadjad Fouladi, Philip Levis ja Matei Zaharia. 2020. “POSH: A Data-Aware Shell”. Teoksessa *2020 USENIX Annual Technical Conference (USENIX ATC 20),* 617–631.

Rahman, Akond, Md Rayhanur Rahman, Chris Parnin ja Laurie Williams. 2021. “Security smells in ansible and chef scripts: A replication study”. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30 (1): 1–31.

Ray, Baishakhi, Daryl Posnett, Vladimir Filkov ja Premkumar Devanbu. 2014. “A large scale study of programming languages and code quality in Github”. Teoksessa *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering,* 155–165.

Rogers, Everett M. 2010. *Diffusion of innovations.* Simon / Schuster.

Stefik, Andreas, ja Susanna Siebert. 2013. “An empirical investigation into programming language syntax”. *ACM Transactions on Computing Education (TOCE)* 13 (4): 1–40.

“The A-Z of Programming Languages: Bourne shell, or sh”. n.d. Viitattu 9. elokuuta 2020. https://web.archive.org/web/20170729170220/https://www.computerworld.com.au/article/279011/a-z_programming_languages_bourne_shell_sh.

“The Internet’s fifth man”. n.d. Viitattu 5. toukokuuta 2020. https://www.economist.com/technology-quarterly/2013/11/28/the-internets-fifth-man.

“The Origin of the Shell”. n.d. Viitattu 16. helmikuuta 2021. https://multicians.org/shell.html.

Tiloca, Marco, ja Gianluca Dini. 2016. “GREP: A group rekeying protocol based on member join history”. Teoksessa *2016 IEEE Symposium on Computers and Communication (ISCC),* 326–333. IEEE.

Walli, Stephen R. 1995. “The POSIX family of standards”. *StandardView* 3 (1): 11–17.

Vasilakis, Nikos, Konstantinos Kallas, Konstantinos Mamouras, Achilles Benetopoulos ja Lazar Cvetković. 2021. “PaSh: light-touch data-parallel shell processing”. Teoksessa *Proceedings of the Sixteenth European Conference on Computer Systems,* 49–66.

Webster, Jane, ja Joseph J Martocchio. 1992. “Microcomputer playfulness: Development of a measure with workplace implications”. *MIS quarterly,* 201–226.

Weide, Bruce W, Wayne D Heym ja Joseph E Hollingsworth. 1995. “Reverse engineering of legacy code exposed”. Teoksessa *Proceedings of the 17th international conference on Software engineering,* 327–331.

Venkatesh, Viswanath, ja Hillol Bala. 2008. “Technology acceptance model 3 and a research agenda on interventions”. *Decision sciences* 39 (2): 273–315.

Venkatesh, Viswanath, ja Fred D Davis. 2000. “A theoretical extension of the technology acceptance model: Four longitudinal field studies”. *Management science* 46 (2): 186–204.

Liitteet

A Kyselylomake

Pro Gradu -kysely kuorista

Kiitos osallistumisesta kyselyyn! Tämä kysely kuuluu osaksi pro gradu -tutkielmaa. Kyselyssä kysellään vastaajalta kysymyksiä kuoren (engl. shell) käyttöön. Vastauksia säilytetään gradun valmistumiseen asti. Henkilötietoja ei kerätä kyselyyn vastaamisen yhteydessä. Kyselyyn vastaus kestää arviolta noin 20 minuuttia.

1. Toimitko

- ◯ Yksityisellä sektorilla
- ◯ Julkisella sektorilla
- ◯ En millään

2.
Oletko

- ☐ Avoimen lähdekoodin ohjelmistojen kehittäjä
- ☐ Ohjelmistoarkkitehti
- ☐ Ohjelmistokehittäjä
- ☐ Ohjelmistotestaaja
- ☐ Opettaja
- ☐ Suunnittelija
- ☐ Tietotekniikan harrastaja
- ☐ Ylläpitäjä
- ☐ Muu

3. Käytätkö mitään kuorta (engl. shell) työssäsi?

- ◯ Kyllä
- ◯ Ei

4.
Mitä kuoria käytät työssäsi?

- [ ] Bash
- [ ] C Shell
- [ ] Korn Shell
- [ ] PowerShell
- [ ] fish
- [ ] zsh
- [ ] oilshell
- [ ] nushell
- [ ] elvish
- [ ] Muu/Muut

5. Kuinka kokeneeksi kuorten käyttäjäksi koet itsesi?

	1	2	3	4	5	
Erittäin kokematon	◯	◯	◯	◯	◯	Erittäin kokenut

Seuraavan sivun kysymykset ovat avoimia kysymyksiä kuoriohjelmoinnista, eli englanniksi 'shell-scripting'. Kuoriohjelmoinnilla tarkoitetaan kuoresta ajettavien erillisten skriptitiedostojen koodamista.

Esim:

```
#!/bin/bash
echo "Hello World"
```

6. Mitkä / Minkälaiset tehtävät kuoriohjelmoit (engl. shell-scripting)?

7. Miksi käytät edellisessä kohdassa mainittuihin tehtäviin erityisesti kuoriohjelmointia?

8. Mitkä toiminnot / ominaisuudet koette hyödyllisinä kuoriohjelmoinnissa?

9. Mitkä tehtävät koette olevan hankalia tai epäkäytännöllisiä kuoriohjelmoinnissa?

10. Haluaisitko käyttää kuoriohjelmointia tehtäviin, joihin et tällä hetkellä käytä sitä?

11. Käyttäisitkö kuoriskriptien sijasta jotain muuta kieltä tai teknologiaa?

Seuraavan sivun kysymykset ovat avoimia kysymyksiä interaktiivisen kuoren käytöstä. Interaktiivisella kuorella tarkoitetaan prosessia, jossa itse kuoreen syötetään komentoja yksi kerrallaan.

Esim. 'cd hakemisto', jonka suorituksen jälkeen jatketaan laittamalla 'ls'.

12. Mihin / minkälaisiin tehtäviin käytät interaktiivista kuorta?

13. Miksi käytät edellisessä kohdassa mainittuihin tehtäviin erityisesti kuorta?

14. Mitkä toiminnot / ominaisuudet koet interaktiivisessa kuoressa hyödyllisinä?

15.
Mitkä tehtävät koet olevan hankalia tai epäkäytännöllisiä interaktiivisessa kuoressa?

16. Haluaisitko käyttää interaktiivista kuorta tehtäviin, joihin et tällä hetkellä käytä kuorta?

17. Käyttäisitkö interaktiivisen kuoren sijasta edellisen kohdan tehtäviin jotain muuta kuin interaktiivista kuorta?