Tietojärjestelmien valvonta ohjelmistorobotiikalla

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus
syksy, 2023

Taru Manninen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä

Tekijä Taru Manninen Vuosi 2023

Työn nimi Tietojärjestelmien valvonta ohjelmistorobotiikalla

Ohjaajat Mirlinda Kosova-Alija

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli tuottaa ohjelmistorobotiikkakoodi, jolla automatisoidaan tietojärjestelmien valvontaa. Tarkoitus oli selvittää erilaisia valvontatoimenpiteitä ja niiden tavoitteita sekä merkitystä, valvonnan merkitystä tietojärjestelmien hallinnassa sekä sitä, miten automatisointiprosessi palvelee tietojärjestelmien valvontaa. Opinnäytetyön toimeksiantaja on järjestelmäintegraatioihin erikoistunut jyväskyläläisyritys.

Opinnäytetyön tietopohja koostuu tietohallinnon roolista ja tietojärjestelmien valvonnasta. Tietojärjestelmien valvonta voidaan jakaa kolmeen alakategoriaan: IT-infran, saatavuuden ja suorituskyvyn valvontaan. Tietopohjassa käydään läpi myös valvontaan liittyviä työkaluja, kuten SNMP:tä sekä Robot Frameworkia, jolla ohjelmistorobotti toteutettiin. Opinnäytetyö on toiminnallinen, päiväkirjamuotoisena toteutettu kehitystyö. Tietoa on kerätty päiväkirjaan, ja näistä päiväkirjamerkinnöistä on koostettu synteesi viikkotasolla.

Kehitystyössä syntyi valvontasuunnitelma, prosessikuvaus ja ohjelmistorobottikoodi. Koodissa on vielä jatkokehitystarpeita, mutta sillä pilotoi ohjelmistorobottia, jolla tullaan valvomaan tietojärjestelmiä. Johtopäätöksenä voidaan todeta, että valvonta on tärkeä osa tietojärjestelmien hallintaa, sillä sen avulla saavutetaan proaktiivinen ote työhön. Automatisaatio taas vähentää valvontaan käytettyä aikaa. Toimeksiantaja oli erityisen tyytyväinen SNMP-työkalun soveltamiseen valvonnan automatisoinnin yhteydessä.

Avainsanat Robot Framework, valvonta, SNMP

Sivut 39 sivua ja liitteitä 3 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology **Abstract**

Author Taru Manninen Year 2023

Subject Monitoring Information Systems with Software Robotics

Supervisors Mirlinda Kosova-Alija

ABSTRACT

The goal of the thesis was to produce a software robotic code to automate the monitoring of information systems. The aim was to explore various monitoring measures their objectives and significance, the importance of monitoring in information system management, and how the automation process serves information system monitoring.

The knowledge base of the thesis consists of the role of information management and information system monitoring. Information system monitoring can be divided into three subcategories: IT infrastructure, availability, and performance monitoring. The knowledge base also covers monitoring tools such as SNMP and Robot Framework, which was used to implement the software robot. The thesis is a functional development work carried out in a diary format. Information has been collected in diary, and the entries of the diary have been synthesized on a weekly basis.

The development work resulted in a monitoring plan, process description, and software robot code. The code still has room for further development, but it serves as a pilot for a software robot that will be used to monitor information systems. As a conclusion, it can be stated that monitoring is an important part of information system management, as it enables a proactive approach to work. Automation, on the other hand, reduces the time spent on monitoring. The client was particularly satisfied with the application of the SNMP tool in the context of automated monitoring.

Keywords Robot Framework, monitoring, SNMP

Pages 39 pages and appendices 3 pages

Sanasto

SNMP	Simple Network Management Protocol, protokolla, joka mahdollistaa laitteiden valvonnan ja hallinnan
OID	Object Identifier, SNMP:n numeerinen merkkijono, jolla voidaan hakea erilaisia tietoja valvottavasta järjestelmästä
Robot Framework	Avoimen lähdekoodin automatisointikehys
JSON	JavaSript Object Notation, tietomuoto, joka perustuu avain-arvo-parien hierarkkiseen rakenteeseen
Kanban	Projektinhallintamenetelmä, jossa resursseja ja tehtäviä hallinnoidaan erilaisilla korteilla
CPU	Central Processing Unit, tietokoneen keskusyksikkö, joka ohjaa laitteen toimintaa
SWAP	Tietokoneen virtuaalimuistitila, johon voi tallentaa tietoja, jos RAM-muisti on täynnä

Sisällys

Kuvat

Liitteet

1 Johdanto

Tämän opinnäytetyön tarkoituksena on selvittää, miten tietojärjestelmien hallintatehtäviä voidaan toteuttaa ohjelmistorobotiikalla. Tämä aihe valikoitui opinnäytetyöksi siksi, että toimeksiantajalla on tarve suorittaa palvelinten valvontaa. Tällä hetkellä siihen ei ole selkeää prosessia, ja valvontarobotin luominen vastaa tähän tarpeeseen.

Toimeksiantajana toimii tietojärjestelmäintegraatioihin erikoistunut jyväskyläläisyritys. Opinnäytetyön tavoitteena on tuottaa ohjelmistorobotti, joka valvoo palvelinprosesseja tuottaen ja tallentaen niistä samalla tietoa jatkoanalysointia varten. Tämä ohjelmointiosio tehdään Robot Framework -työkalulla. Toimeksiantaja haluaa määrämuotoistaa ja säännönmukaistaa tietojärjestelmiin ja palvelimiin liittyvää valvontaa ja hallintaa.

Opinnäytetyön tuloksena syntyvän valvontarobotin myötä valvontaan käytettävä henkilötyöaika vähenee ja verkkopalveluiden luotettavuus kasvaa. Valvontaprosessin selkenemisen myötä myös poikkeustilanteet saadaan reaaliaikaisesti tietoon. Robotin tekemä valvonta myös tuottaa säännöllistä, täsmällistä ja reaaliaikaista seurattavaa raporttidataa, jota voi hyödyntää palvelun laadun varmistamisessa ja kuvaamisessa asiakkaalle.

Opinnäytetyö käsittelee yhden tarkkaan määritellyn prosessin suunnittelun, kuvailun sekä toteutuksen ohjelmistorobotiikalla. Se ei tule selvittämään laajasti sitä, mitä kaikkea ohjelmistorobotiikalla voi tai kannattaa tehdä. Opinnäytetyössä ei myöskään vertailla erilaisia vaihtoehtoja toteuttaa tietojärjestelmän hallintaa.

Opinnäytetyö raportoidaan päiväkirjana, jossa seurataan käytännön toteutusta suunnittelusta robotin käyttöönottoon asti.

Tutkimuskysymykset, joihin tässä opinnäytetyössä on tarkoitus vastata, ovat:

- Mikä on valvonnan merkitys tietojärjestelmien hallinnassa?
- Mitä erilaisia valvontatoimenpiteitä on, ja mikä on niiden merkitys ja tavoite?
- Miten automatisointiprosessi palvelee tietojärjestelmien valvontaa?

2 Kehittämisprojektin tietoperusta

Tässä luvussa käsitellään kehitystyön tietoperustaa. Tässä luvussa käydään läpi Robot Framework, tietojärjestelmän ja tietohallinnon määritelmiä ja eroja, erilaisia valvontatoimenpiteitä, prosessien kuvaamista, ketteriä menetelmiä sekä SNMP-työkalua.

2.1 Tietojärjestelmien valvonta tietohallinnossa

Tietohallinnolla on useita tehtäviä organisaatiossa. Keskeistä on kuitenkin liiketoiminnan tukeminen ja kehittäminen teknologisten ratkaisuiden avulla. Sen lisäksi tietohallinnon tehtävänä on tiedon hallinnointi, kerääminen, jakaminen sekä ylläpito. Tiedon hallinnointi keskittyy niin tiedon kuin tietoliikenteen sekä tiedon omistajuuden järjestelyyn ja sisältää valvontaa, käsittelyä, arviointia. Lisäksi tietohallinnossa tavoitellaan teknologisten palveluiden laadun ja kustannustehokkuuden optimointia. (Tietohallinto, n.d.)

Tietohallinto voidaan jakaa erilaisiin kategorioihin niiden tavoitteiden ja organisoitumisen mukaan. Nämä kategoriat ovat osaamisalue, tukitoiminnot, johtamistoiminnot ja liiketoiminnot. Osaamisalueen kategoriassa ICT-taitoiset ihmiset tunnistetaan, mutta he eivät ole välttämättä järjestäytyneet tietohallinnoksi, eikä yksilöiltä ei useinkaan odoteta kykyä uudistaa tai kehittää tietojärjestelmiä. Tukitoimintojen tehtävänä on varmistaa, että ICT-palvelut ovat luotettavia ja vastaavat liiketoiminnan tarpeita. Siltä voi odottaa jo tietojärjestelmien kehittämistä, strategista ajattelua, projektien läpivientikykyä sekä järjestäytyneisyyttä. (Huovinen, 2011.)

Muut tietohallinnon kategoriat ovat johtamistoiminnot ja liiketoiminnot. Johtamisen alle mielletään johtamisen kehittäminen informaatioteknologian avulla. Tämä tarkoittaa esimerkiksi prosessien ja kilpailukyvyn kehittämistä. Liiketoimintakategoriassa organisaation informaatioteknologia on usein jo palvelutarjontaa tai ulospäin myytävä tuote. (Huovinen, 2011.)

2.1.1 Tietojärjestelmä

Tietojärjestelmän määrittelyn katsotaan sisältävän tietojärjestelmän roolin organisaatiossa ja lisäksi tietojärjestelmän osatekijöitä, jotka ryhmitellään usein näin: laitteet, ohjelmistot, data,

ihmiset sekä prosessit. Kaikilla näillä osatekijöillä on oma osuutensa tietojärjestelmän toimivuuden kannalta. (What Are Information Systems? – Defined and explained in 2023, n.d.)

Laitteet (hardware) on usein näkyvä ja fyysinen osa tietojärjestelmää. Suurinta osaa ohjelmistoista ja tiedoista käytetään ja käsitellään laitteiden kautta. Ohjelmistot kertovat laitteille, mitä niiden kuuluu tehdä. Ohjelmistot jaetaan usein kahteen eri tyyppiin eli käyttöjärjestelmäohjelmistoihin sekä sovellusohjelmistoihin. Käyttöjärjestelmäohjelmisto on laitteiston ohjausta ja käyttöä varten ja sovellusohjelmisto tarjoaa hyödyllisiä toimintoja esimerkiksi tietojen käsittelyyn. (What Are Information Systems? – Defined and explained in 2023, n.d.)

Tietojärjestelmissä oleva data on kokoelma informaatiota ja faktoja. Dataa kerätään päätöksenteon tueksi sekä liiketoiminnan kehittämiseksi. (What Are Information Systems? – Defined and explained in 2023, n.d.) Datan avulla voidaan myös ennustaa trendejä ja skenaarioita sekä tarkastella liiketoiminnan suorituskykyä. (Stedman & McLaughlin, 2022.)

Ihmiset, jotka käyttävät tietojärjestelmiä, tekevät tietojärjestelmistä hyödyllisiä. Ihmiset kehittävät, tukevat ja käyttävät tietojärjestelmiä. Onkin tärkeää, että tietojärjestelmä on ihmisten opittavissa ja käytettävissä, sillä ilman ihmisiä tietojärjestelmä on hyödytön. (Rock, ym., 2022.)

Prosessi on tapahtumasarja, jolla pyritään pääsemään haluttuun lopputulokseen. Tietojärjestelmien tavoitteena on kehittää ja tehostaa liiketoiminnan prosesseja niin että toiminnasta syntyy kilpailuetua tai työskentely tehostuu. (Rock, ym., 2022.)

Tietojärjestelmän tavoitteena on vähentää inhimillisiä virheitä automaation kautta. Lisäksi se mahdollistaa määrällisen datan keräämisen organisaation sidosryhmiltä, kuten asiakkailta ja toimittajilta. Tietojärjestelmä myös tehostaa tietojen käsittelyyn käytettyä aikaa, joka taas tukee päätöksenteossa. (What Are Information Systems? – Defined and explained in 2023, n.d.)

Toimivat tietojärjestelmät ovat liiketoiminnalle nykypäivänä elintärkeitä. Pääasiallisesti näillä järjestelmillä tarkoitetaan järjestelmiä, jotka eivät välttämättä näy normaalille työntekijälle tai käyttäjälle. Tällaisia ovat esimerkiksi palvelimet, varmuuskopiointi- ja tallennusjärjestelmät sekä

verkkolaitteet. Näissä järjestelmissä toimii usein yrityksen ydintoimintoja liittyen esimerkiksi resurssienkäyttöön. (Why Do my Computer Systems need monitoring, n.d.)

Tärkeimpiä syitä tietojärjestelmien valvomiselle on se, että järjestelmänvalvojan rooli muuttuu reaktiivisesta proaktiiviseksi, eli toimintaa ennakoivaksi. Tämä vähentää esimerkiksi käyttökatkoksia palveluissa sekä tietoturvaloukkauksien mahdollisuuksia. Hyvä valvontajärjestelmä myös kerää ja säilyttää tietoa ja mahdollistaa täten erilaisten kehityssuuntien seuraamisen. Kehityssuuntien seuraaminen voi auttaa ennakoivassa päätöksenteossa. Valvonta lisää tietokonejärjestelmien luotettavuutta sekä stabiiliutta. (Why Do my Computer Systems need monitoring? n.d.) Valvonnalla voidaan saavuttaa myös säästöjä kustannuksissa. (Why is server and IT system monitoring necessary? 2020.)

2.1.2 Valvonnan kohteet

Valvonnalla yleensä pyritään vastaamaan neljään perustavanlaatuiseen kysymykseen: mikä on olemassa oleva resurssi, toimiiko se, miten hyvin se toimii ja miten paljon olemassa olevaa resurssia käytetään? Tässä ei ole vielä merkitsevää se, mitä valvotaan. (Bigelow, 2020.) Ei ole olemassa virallista listaa erilaisista IT-valvonnan muodoista, mutta erilaisia keskeisiä valvonnan muotoja ovat käytettävyyden valvonta, verkon suorituskyvyn valvonta, sovellusten ja sovellusten suorituskyvyn hallinta, API-valvonta, liiketoiminnan valvonta, tietoturvavalvonta sekä todellisen käytön valvonta. (What is IT Monitoring? 2023.)

IT-infrastruktuuri on yhdistelmä organisaation laitteita, ohjelmistoja ja verkkoja, jotka luovat organisaation tietoteknisen ympäristön. (Wrobel, 2023.) Tietohallinto eroaa infrastruktuurista siten, että IT-infrastruktuuri on tietohallinnon alainen kokonaisuus, joka sisältää lähinnä IT-ympäristön teknologisen perustan. (Infrapalvelut, n.d.)

IT-infrastruktuuri jaetaan usein kolmeen tyyppiin: perinteiseen (”on-premise”), pilvipohjaiseen sekä hyperkonvergenttiin infrastruktuuriin. Perinteinen IT-infrastruktuuri tarkoittaa sitä, että organisaatio itse hallinnoi laitteita, ohjelmistoja sekä verkkoja. ja se tarvitsee fyysisiä tiloja sekä energiaa ylläpitoonsa, mikä saattaa nostaa kustannukset korkeaksi. Se kuitenkin tarjoaa vahvaa

tietoturvaa ja on hyvä valinta tilanteessa, jossa organisaation on noudatettava tiukkoja standardeja. (Wrobel, 2023.)

Pilvipalvelu-infrastruktuurissa käyttäjien ei tarvitse asentaa mitään omalle tietokoneelleen, vaan virtualisoinnin myötä käyttäjät yhdistävät infrastruktuuriin internetin kautta. Pilvipalvelu myös jakaa resursseja ja tarjoaa yksinkertaisempia käyttöliittymiä niin että loppukäyttäjän ei tarvitse tietää juuri teknologisia yksityiskohtia käyttämästään palvelusta. (IT- Infrastucture, 2021.)

Hyperkonvergentti infrastruktuuri yhdistää sekä perinteistä infrastruktuuria että pilvipalvelua integroimalla erilliset komponentit, kuten tietojenkäsittelyn, tallennustilan, verkon ja virtualisoinnin, yhteiselle alustalle, mikä lisää tehokkuutta ja skaalautuvuutta. (Wrobel, 2023.)

IT-infrastruktuurin valvontaan kuuluu datan keräys ja analysointi erilaisista systeemeistä ja prosesseista. IT-Infrastruktuurin valvonnan tavoitteena varmistaa, että IT-infrastruktuurin saatavuus ja suorituskyky soveltuvat yhteen liiketoiminnan tavoitteiden ja loppukäyttäjän kokemuksen kanssa. Sen avulla luodaan kokonaiskuvaa IT-ympäristön suoriutumisesta. (Marinelli, 2023.)

Palvelinten valvonta on osa IT-intrastruktuurin valvontaa. Sen tehtävä on havainnoida ja jaotella olemassa olevia erilaiset palvelimia ja muita resursseja, kuten ajureita ja käyttöjärjestelmiä. Valvonta kerää ja käsittelee palvelimiin liittyvää informaatiota, kuten saatavuutta, suorituskykyä ja resurssien käyttöä. Mittareita seuraamalla voidaan havaita suorituskyvyn muutokset, tunnistaa häiriötilanteet sekä trendimuutokset, jotka voivat ennustaa ongelmatilanteita. (Bigelow, 2020.)

Borginin (2020) mukaan palvelinvalvonnan tärkeitä erilaisia komponentteja ovat muistin, tallennusverkon, palvelimen levykapasiteetin, tallennuksen nopeuden, fyysisten lämpötilojen, käyttöjärjestelmien, ohjelmiston ja sovelluksien sekä CPUn eli keskusyksikön (Central Processing Unit) valvonta. Bertram (2020) lisää listaan vielä käyttäjien aktiivisuuden, virheet ja niiden lokit sekä passiivisuuden.

Hälytysten huolellinen valinta ja säätö sekä keskittyminen systeemiin ja sovelluksiin ovat tietojärjestelmävalvonnan käytäntöjen ydintä. Oikeiden mittareiden valinta tavoitteisiin nähden tuo oikeita tuloksia. Hälytysten huolellinen säätö ja valinta niin, että valvonta tuottaa oikeaa tietoa

oikea-aikaisesti vähentää taas henkilöstön kuormitusta. Kaikki informaatio ei ole yhtä tärkeää ja mahdollisesti kriittiset hälytykset jäävät vähemmälle huomiolle, jos valvonta lähettää paljon informaatiota, joka ei vaadi toimenpiteitä tai reagointia. (Bigelow, 2020.) Hälytysten tarkka määrittely myös vähentää vääriä hälytyksiä. Muita hyviä käytänteitä valvonnassa on hyödyntää automaatiota, tapahtumalokeja, tarkistaa ja testata mittarit ja valvontaprosessit säännöllisesti. (Marinelli, 2023.)

2.1.3 Saatavuuden valvonta

Saatavuus on tärkeä liiketoiminnan ja asiakastyytyväisyyden edellytys. Sen valvonnalla ja hallinnoinnilla minimoidaan palveluhäiriöiden kestoa, vaikutuksia sekä ilmenemistiheyttä. Hallinnoimalla ja valvomalla saatavuutta näitä häiriöitä on myös mahdollista havaita ja tunnistaa sekä korjata sujuvammin. (Gillingham, 2023.)

Saatavuuden tai käytettävyyden hallinnan työtehtävinä on muun muassa luoda, ylläpitää ja toteuttaa saatavuussuunnitelma, joka ottaa huomioon liiketoiminnan tavoitteet, määrittelee saatavuustavoitteet ja -vaatimukset ja varmistaa, että palvelut saavuttavat nämä tavoitteet ja vaatimukset. Jos tavoitteet eivät täyty, saatavuuden hallinta tunnistaa ja ratkaisee tähän liittyviä ongelmia ja häiriötilanteita. Lisäksi saatavuuden hallinta valvoo saatavuutta, eli suorittaa mittauksia, hallinnoi ja kehittää palvelua, samalla kun se ohjaa saatavuuteen liittyvää toimintaa. (Gillingham, 2023.)

Saatavuus jaetaan ylätasolla kahteen, kuitenkin osittain päällekkäiseen toisensa lomittuvaan tasoon: Palvelun saatavuuden hallinta ja komponenttien hallinta. Kokonaiskäytettävyyteen ja käyttökatkoksien havaitsemiseen liittyy näiden edellä mainittujen tasojen käytettävyyteen (kyky suorittaa annettu tehtävä vaadittaessa), luotettavuuteen (kuinka pitkään palvelu tai osa voi toimia ilman katkoksia), huollettavuuteen (kuinka helppo ja nopea palvelu tai komponentti on palauttaa toimintakykyiseksi vikaantuessaan) ja palvelukelpoisuuteen (kyky täyttää sovitut ehdot, mukaan lukien käytettävyys, huollettavuus ja luotettavuus). (Gillingham, 2023.)

Keskeisiä käytettävyyden hallinnan toimia ovat käytettävyysvaatimusten määrittely, puutteiden sekä ilmi tulevien häiriöiden, keskeytysten ja muiden ongelmien analysointi sekä ratkaisu,

palveluiden ja komponenttien käytettävyyden seuraaminen, mittaus, raportointi ja analysointi, erilaisten vikojen vaikutusanalyysi sekä luoda malleja, joilla selvitetään täyttävätkö komponentit ja palvelut liiketoiminnan vaatimukset. (Gillingham, 2023.)

2.1.4 Suorituskyvyn valvonta

Suorituskyvyn hallinnan tarkoitus on varmistaa, että organisaation tietotekninen infrastruktuuri on linjassa organisaation tavoitteiden kanssa. Tällä tarkoitetaan muun muassa yleistä järjestelmän suorituskykyä, palvelinten ja pilvipalveluiden valvontaa, sovellusten suorituskyvyn valvontaa sekä verkon hallintaa ja valvontaa. Valvomalla suorituskykyä voidaan ennustaa tulevaisuuden kysyntää ja tunnistaa alikäytettyjä tai alisuoriutuvia resursseja. Suorituskyvyn valvonta ulottuu kaikille organisaation tasoille; valvonnan piiriin kuuluu laitteet, selaimet, sovellukset ja palvelut, palvelimet, virtualisoinnit, levyn muisti, IT-infrastruktuuri, verkko ja tietoturva. (Allen, 2023.)

Suorituskyvyn valvonta tarjoaa mahdollisuuksia optimoida IT-infraa. Tärkeää on määritellä tavoitteet, valvottavat kohteet, työkalut ja mittarit, jotta valvonta tarjoaa oikea-aikaista tietoa. Kyse on nimenomaan suorituskyvyn arvioinnista eikä arvostelusta. Arviointi voi toimia strategisena työkaluna, jolla voidaan johtaa muita liiketoiminnan toimintoja. (Bigelow, 2019.)

Suorituskyvyn prosessin valvonta seuraa seuraavia askeleita: asetetaan tavoitteet, määritellään mittauksen laajuus, kerätään dataa, analysoidaan ja arvioidaan kerättyä dataa ja lopuksi tehdään suosituksia suorituskyvyn optimoinniksi. Tärkeää on määrittää tärkeät KPI:t (key performance indicator), jotta ei mitata vääriä asioita, vaan oikeasti liiketoiminnalle tärkeitä indikaattoreita. (Bigelow, 2019.)

Suorituskyvyn mittaamisen KPI:ksi voidaan laskea CPU:n eli keskusyksikön käyttö, muistin käyttö, muistin vapautuminen, kuinka monta pyyntöä minuutissa ja kuinka monta tavua pyyntöä kohden suoritetaan, viive ja käytettävyys, keskimääräinen vastausaika, turvallisuus, käyttäjätyytyväisyys, virheiden määrä sekä liikennemäärä. (Kanjilal, 2022.)

Korkea keskusyksikön käyttö kertoo, että valvottava laite on kiireinen ja saattaa kertoa siitä, että kohde on suorituskykynsä rajoilla. Korkea muistin käyttö taas indikoi sitä, että resursseja käytetään

paljon. Tällöin kone saattaa alkaa sivuttaa ohjelman suoritusta siirtämällä osan suoritettavasta prosessista kovalevylle. Tämä heikentää suorituskykyä. Muistin vapautuminen saattaa myös käyttää keskusyksikköä ja pysäyttää kohteen hetkittäin. Tätä voi mitata erilaisilla muistin vapautumisen käsittelyajoilla ja prosenteilla. (Kanjilal, 2022.)

Palvelimen käyttäytymistä kuormitustilanteissa arvioidaan sillä, kuinka monta pyyntöä minuutissa vastaanotetaan ja kuinka monta tavua per pyyntö käsitellään. On mahdollista, että pyyntöjä saadaan enemmän kuin voidaan käsitellä tai käsiteltävän tiedon määrän on liian suuri. Viive ja käytettävyys voidaan testata ping- komennolla. Tällä selvitetään viivettä käyttäjän ja sovelluksen tai palvelimen vastauksen välillä. Keskimääräinen vasteaika liittyy myös vahvasti tähän; pieni vasteaika kertoo yleensä paremmasta suorituskyvystä. Vastausaika liittyy myös käyttäjien tyytyväisyyteen. (Kanjilal, 2022.)

Virheiden määrää voi seurata esimerkiksi virhelokeista, heitetyistä poikkeuksista sekä http-virhesuhteesta eli epäonnistuneiden verkkopyyntöjen määrästä. Kasvava virheiden määrä saattaa tarkoittaa, että jonkinlainen merkittävä isompi vika on tulossa. Liikennemäärä mittaa ja indikoi kohteen hiljaisista jaksoista ja toisaalta mahdollisesta liikenteen kasvavasta tai vähenevästä trendistä ja antaa toisaalta tietoa siitä, miten kohde voisi skaalautua paremmin. (Kanjilal, 2022.)

2.1.5 SNMP-valvontatyökalu

SNMP tulee sanoista Simple Network Management Protocol. Se on ollut verkkovalvonnan käytössä jo 1990-luvulta asti ja erilaiset verkkolaitteet ja valvonta-alustat tukevat sitä laajasti. SNMP:llä on kolme erilaista versiota, joista versio 3 (SNMPv3) sisältää eniten tunnistautumisominaisuuksia. SNMP kerää dataa kyselymekanismilla ja palauttaa tiedot sitten hallinnointialustalle. (Slattery, 2021.)

SNMP tarjoaa yhteisen kielen tietojen jakamista varten. Sen arkkitehtuuri jaetaan kolmeen tasoon: SNMP-hallintajärjestelmä, SNMP-agentti sekä hallintatietokanta. Hallintajärjestelmä toimii ikään kuin asiakkaana SNMP-agentin toimiessa palvelimena, ja hallintatietokanta on SNMP-agentin tietokanta, jota SNMP-agentti käyttää apunaan tarjotakseen vastauksen SNMP-hallintajärjestelmän esittämään kysymykseen. SNMP on niin yleinen, että SNMP-agentti on usein jo valmiiksi asennettu erilaisiin verkkolaitteisiin. Se täytyy kuitenkin konfiguroida ensin, että

agentti voi kommunikoida hallintajärjestelmän kanssa. SNMP-agentti ja hallintatietokanta ovat asennettuina laitteessa ja keräävät laitteen informaatiota koko ajan. Tiedot välitetään hallintajärjestelmään kuitenkin vasta pyydettäessä. On mahdollista asettaa myös niin kutsuttu loukku, ”trap”, jolloin ennalta määritetyn rajan ylittyessä SNMP-agentti lähettää viestin. (Scarpati, 2021.)

Hallintatietokanta, eli management information base, MBI, on kuvaus valvottavan laitteen tiedoista, kuten komponenteista ja statuksesta. Sen voi luoda lähes mihin tahansa verkkolaitteeseen. Jokainen laitteen ominaisuus, jota voi hallita tai mitata SNMP:llä, saa objektin tunnisteen, OID:in. (Scarpati, 2021.)

2.2 Robot Framework

Robot Framework on avoimen lähdekoodin automatisointiohjelma, jota käytetään testien automatisoinnissa sekä robottien prosessien luonnin automatisaatiossa eli RPA:ssa (Robotic Process Automation). Se on melko helposti laajennettavissa eri ohjelmointikielillä tehdyillä kirjastoilla ja muilla työkaluilla (Robot Framework, n.d.-a). Robot Framework perustuu Python-ohjelmointikieleen ja sen syntaksia, eli kielen sääntöjä ja rakennetta, jolla automatisoitavia tehtäviä määritellään, pidetään yleisesti helppona. (Robot Framework, n.d.-b.)

Robot Framework on saanut alkunsa 2004 Pekka Klärkin pro gradu tutkielmasta ja sen kehitys alkoi, kun Nokia Networks tarvitsi testaustyökalun. Robot Frameworkin käyttö on levinnyt julkaisustaan lähtien nopeasti, ja sitä käytetään erityisesti Suomessa mutta myös maailmanlaajuisesti. (Robot Framework: Past, Present and Future, 2021.)

2.2.1 Robot Framework-kirjastot

Robot Frameworkin toiminnallisuuksia pystyy laajentamaan erilaisten kirjastojen kautta. Nämä kirjastot auttavat Robot Frameworkia toimimaan erilaisten järjestelmien kanssa. Näitä lisäosia kutsutaan kirjastoiksi. (Basic concepts of Robot Framework, n.d.) Robot Frameworkin käyttämät avainsanat ovat aina jostain kirjastosta, esimerkiksi” open browser”. Normaalisti kirjasto tuodaan robotin käyttöön asetukset-kohdassa, mutta on myös mahdollista käyttää komentoa ”import library”. (Robot Framework, n.d.-b.)

Robot Framework kirjastoja on kahdenlaisia, standardikirjastoja ja ulkoisia kirjastoja. Standardikirjastoja ovat BuildIn, Collections, DateTime, Dialogs, Operating System, Process, Screenshot, String, Telnet ja XML. Ulkoisia kirjastoja kehitetään lisää avoimen lähdekoodin ympäristössä, mistä muutamia esimerkkejä ovat SeleniumLibrary ja SwingLibrary. Näitä ulkoisia kirjastoja ei ole sisällytetty Robot Frameworkin peruskehykseen. (Robot Framework, n.d.-b.)

2.2.2 Muuttujat

Robot Frameworkissa voi käyttää määriteltäviä muuttujia. Tästä on hyötyä esimerkiksi, jos jotkin muuttujat muuttuvat usein, jolloin ainoastaan määriteltyä muuttujaa tarvitsee muuttaa. Käytettäviä muuttujia Robot Frameworkissa ovat skalaarit muuttujat, listat, sanakirjat ja ympäristömuuttujat, joita merkitään erilaisilla etumerkeillä. (Robot Framework, n.d.-b.)

Skalaarimuuttujaa merkataan käyttämällä merkkiä $. Tällöin muuttujan nimi korvataan määritellyllä arvolla. Yleisesti skalaarimuuttujaa käytetään merkkijonoissa, mutta ei ole estettä käyttää sitä muissakin tilanteissa, kuten listoissa tai numeroissa. Listoja merkitään @-merkillä. Tämä eroaa skalaarimuuttujasta niin, että listan alkiot esitetään yksittäisinä objekteina. Jos lista on skalaarimuuttuja, lista on yksi objekti. (Robot Framework, n.d.-b.)

2.3 Kanban-taulun käyttö ketteränä menetelmänä

Ketterät menetelmät ovat projektinhallinnan työkaluja. On olemassa useita ketteriä projektinhallintamenetelmiä, kuten Scrum, Lean, Kanban ja SAFe. Erilaisissa menetelmissä on eroja, mutta niillä on myös yhteneväisiä piirteitä, kuten jatkuva kehittyminen kokeilujen kautta, joustavuus, avoin kommunikaatio sekä muutosmyönteisyys. Ketterät menetelmät pyrkivät välttämään liian tarkkaan määriteltyjä rakenteita ja suunnitelmia. On havaittu, että organisaation yhteistyö ja kommunikaatio ovat parantuneet, mikäli heillä on käytössään ketteriä menetelmiä. (Mitä ovat ketterät menetelmät? – Scrum, Lean ja muut tutuksi, 2021.)

Kanban-menetelmässä hyötynä on visuaalisuus, sillä tehtävät järjestellään visuaaliseen muotoon niin, että tehtävien järjestyksen ja pullonkaulojen havainnointi on helpompaa. Kanban sopii erityisesti tasaisiin projekteihin. (Mitä ovat ketterät menetelmät? – Scrum, Lean ja muut tutuksi,

2021.) Kanban on yhdenlainen perusmalli, jota voi kehittää itselleen sopivaksi ajan kanssa, kun itse kehittyy projektinhallinnoijana. (Hietaniemi, 2020.)

Yleinen tapa käsitellä Kanban-taulua on luoda kolme saraketta: ”tekemättä”, ”työn alla” ja ”valmis”. Tekemättä sarake toimii myös niin sanotusti työjonona. Kanban yhdistetään usein muihin ketteriin menetelmiin niin, että se on vain lisätyökalu esimerkiksi Scrumissa. (Koskinen, 2021.)

Perustoimintatapoja Kanbanissa on kolme. Tarkoitus on tuoda työ näkyväksi niin, että jokainen vaihe ja työtehtävä tuodaan yhteiselle Kanban-taululle koko tiimin nähtäville. Toisena toimintatapana on mitata läpimenoaikaa niin, että ”työn alla” olevat tehtävät siirtyvät tilaan ”valmis” kohtuullisessa ajassa. Kolmas toimintatapa on rajoittaa keskeneräisen työn määrää, sillä liian iso määrä keskeneräisiä töitä häiritsee työn kulkua ja voi luoda pullonkauloja. (Koskinen, 2021.)

Kanbanin hyödyiksi luetaan yleisesti visuaalisuus sekä läpinäkyvyys tehtävänjaossa. Se mahdollistaa mahdollisten hidasteiden löytymisen sekä uudelleenpriorisoinnin, jos projektin tavoitteet muuttuvat. Kanbanin aloittaminen on helppoa ja se on varsin kevyt hallinnoida. Se voi myös lisätä itseohjautuvuutta ja osallistuvuutta, sillä jos kaikki ovat sitoutuneita Kanban-taulun käyttöön, on kaikkien mahdollista osallistua projektin hallintaan. Samalla sen puutteeksi luetaan se, että jos kaikki eivät ole sitoutuneita Kanban-taulun käyttöön, se ei tue projektinhallintaa. Menetelmä ei myöskään tue projektia, jos keskeneräiset työt kasautuvat. (Koskinen, 2021.)

2.4 Prosessikuvaus

Prosessi määritellään yleensä toiminnaksi, jossa tehdään tiettyjä toimia tietyssä järjestyksessä, jotta saavutetaan haluttu lopputulos. Prosessijohtaminen on kokonainen oma johtamisen alansa, jossa pyritään tehostamaan organisaation toimintaa muun muassa asiakastyytyväisyyden parantamiseksi. Hyvä prosessi tuo lisäarvoa asiakkaalle sekä minimoi tuottajan ’hukkaa’ – eli resursseja ja työaikaa. Lisäksi hyvä prosessi minimoi virheitä ja poikkeamia tai vähintäänkin tuo poikkeamat tietoisuuteen, jotta niihin voidaan reagoida. Prosessijohtamiselle on tärkeää, että

prosessi on tunnistettu, dokumentoitu ja että prosessia noudatetaan. (Prosessien kehittäminen. N.d.)

Prosessin johtamiselle on tunnistamisen ja kuvaamisen lisäksi tärkeää, että sen tehokkuutta mitataan, sitä parannetaan jatkuvasti, sen rajapinnat selkiytetään ja sen omistaja on nimetty. Prosessinomistajan tehtäviin voi kuulua muun muassa prosessin ylläpito ja kehitys sekä tarvittavien ylläpito ja kehitystoimien suunnittelu ja toteutus, mittariston luominen ja seuraaminen sekä tavoitteiden asettaminen. Eli prosessilla täytyy olla määritelty vastuuhenkilö. (Lindroos, 2021.)

Prosessilla on aina alku- ja loppupiste. Näiden kahden pisteen välissä tapahtuu prosessi. Prosessin tunnistaminen alkaa siitä, että nämä kaksi pistettä määritellään. Prosessin rajaaminen on myös hyödyllistä, sillä yksi prosessi tuskin voi sisältää kaikkea mahdollista. (Prosessi – miksi ja miten kehittää? 2020.)

Prosessien kuvaamisella on lukuisia hyötyjä: sen avulla voidaan tunnistaa riskejä, mahdollisuuksia, resurssiongelmia, työn tekoon liittyviä vaaroja ja vaiheita, jotka eivät tuota lisäarvoa. Prosessin kuvaamisen myötä on mahdollista myös arvioida tehokkaammin sitä, millaisia automatisointimahdollisuuksia prosessissa on. Tämä taas voi luoda mahdollisuuksia optimoida resursseja, kuten esimerkiksi työntekijöiden ajankäyttö. Prosessinomaisen toiminnan hyötyjä on myös toiminnan läpinäkyvyys sekä erilaisten sovellusten tuottamien tuotanto-ongelmien tunnistaminen. Osittain nämä ongelmat voivat olla tiedon ja sisällön liikkumiseen liittyviä ongelmia. (Lindroos, 2021.)

Prosesseja kuvataan monilla eri tavoin, esimerkiksi prosessikartoilla. Näillä kartoilla pyritään havainnollistamaan prosessien etenemistä sekä prosessin mahdollisia sisäisiä linkityksiä. Prosessikuvauksessa on tärkeää nostaa esiin ne asiat, mitkä ovat prosessin toiminnan kannalta oleellisia. Prosessikuvauksessa käytettävien käsitteiden olisi oltava yhtenäisiä ja ristiriidattomia. (Prosessi – miksi ja miten kehittää? 2020.) Tärkeää on myös, että prosessin kuvaus on joustava sillä tavoin, että sen muokkaaminen ja päivittäminen on helppoa, ja kuvauksella ja prosessilla itsellään on selkeät tavoitteet. (Saarinen, 2021.)

2.5 Avoimen datan siirto JSON:lla

JSON, eli JavaScript Object Notation, on helppokäyttöinen tapa siirtää avointa dataa. Sen rakennetta pidetään suoraviivaisena ja helppolukuisena muun muassa jäsentelynsä ansiosta. JSON käyttää arvotyyppeinään merkkijonoja, numeroita, loogisia arvoja, luetteloita, objekteja ja nollaa. Nämä arvotyypit löytyvät yleensä lähes kaikista ohjelmointikielistä, minkä takia sitä tuetaankin natiivisti tai kirjastojen kautta. (Fruhlinger, n.d.)

JSON on niin sanotusti skeematon, eli se ei vaadi tarkkaa tietorakenneskeemaa. Tämä luo joustavuutta siihen, että voit käytännössä luoda minkälaisia tahansa tietoja. Tämä luo kuitenkin myös riskin vääristyneille tiedoille. Toinen rajoitus on se, että JSON käyttää vain yhtä numerotyyppiä, kaksitarkkaa liukulukua. JSON ei sisällä myöskään päivämäärätyyppiä, joten tästä voi syntyä muotoilueroja. JSON ei myöskään mahdollista kommentointia, ja lisäksi se on sanallinen, eli verrattuna XML -muotoon se kärsii ytimekkyyden puutteesta. (Fruhlinger, n.d.)

JSON rakenteessa on neljä sääntöä. Data erotetaan pilkuilla, aaltosulkeet pitävät sisällään objekteja ja neliösulkeiden sisällä on taulukoita. Lisäksi JSON-tiedot ovat nimi-arvo-pareja. (JSON-Introduction, n.d.)

3 Kehittämisprojektin tavoite ja tarkoitus

3.1 Tavoite ja tarkoitus

Opinnäytetyön tavoitteena on kehittää ohjelmistorobotti, joka valvoo palvelinprosesseja tuottaen ja tallentaen niistä samalla tietoa jatkoanalysointia varten. Tämä ohjelmointiosio tehdään Robot Framework -työkalulla. Toimeksiantaja haluaa määrämuotoistaa ja säännönmukaistaa tietojärjestelmiin ja palvelimiin liittyvää valvontaa ja hallintaa.

Kehitystyön tietoperusta on tietohallinnossa sekä Robot Frameworkissa. Teoreettisessa osuudessa perehdytään tietohallintoon ja sen jälkeen syvennytään tietohallinnon alaiseen valvontaan. Valvonnan teoriaosuudessa käydään läpi erilaisia valvonnan tyyppejä sekä selvitetään erilaisia valvonnan tapoja. Robot Frameworkiin liittyvässä teoriaosuudessa käydään läpi Robot Frameworkin ominaisuuksia, jotta niitä voidaan hyödyntää itse valvontarobotin rakentamisessa. Nämä molemmat ovat selkeästi yhdistettävissä itse kehitystyöhön, sillä on mahdotonta rakentaa robottia ymmärtämättä ohjelmaa, ja lisäksi perehtymällä valvontaan on mahdollista selvittää juuri tähän kehitystyöhön sopivat valvontatoimenpiteet.

Menetelmäksi valikoitui päiväkirjamuotoinen opinnäytetyö, sillä kehitystyötä on mahdollista dokumentoida näin. Se kehittää myös kirjoittajan taitoja asiantuntijakirjoittajana. Kehitystyön tekijä on jo vahvasti asiantuntijatehtävissä kiinni ja koska työ tehdään aidosti käyttöönotettavana kokonaisuutena, tämä yhdistelmä tarjoaa myös joustavuutta kehitystyön tekemiseen.

3.2 Kehittämisprojektin suunnittelu ja toteutus

Kehitystyön teoriaosuudessa käsitellään tietohallinnon ja tietojärjestelmien käsitteitä sekä syvennytään yleisimpiin erilaisiin valvontatoimenpiteisiin. Lisäksi teoriassa käydään läpi valvonnan merkitystä yleisesti, sekä erilaisten valvontatoimenpiteiden merkitystä. Toinen osa teoriasta syventyy Robot Frameworkkiin ja sen käyttöön, sekä SNMP työkaluun.

Käytännön osa koostuu päiväkirjasta sekä Robot Frameworkin koodiosiosta sekä selvityksestä, miten robotti saadaan toimimaan. Käytännön osiossa luonnostellaan prosessikaavio sekä valvontasuunnitelma. Päiväkirjan tuotoksista kirjoitetaan viikkotasolla analyysi siitä, mitä uutta

tietoa syntyi. Päiväkirjana toimii Microsofin Planner- projektinhallintaympäristö, joka antaa määrämuotoisen kehikon päiväkirjamerkinnöille ja se dokumentoi samalla projektin etenemisen. Microsoft Planner toimii myös samalla projektinhallinnan Kanban-tauluna.

Kanban-menetelmän käyttö ketterän kehittämisen menetelmänä valikoitui siksi, että projektin eri osa-alueita oli mahdollista edistää samanaikaisesti. Se oli myös helppo muokata ja jäsennellä tähän projektiin sopivaksi. Kanban sopii myös helposti ylläpidettäväksi päiväkirjamuotoisen opinnäytetyön oheen, sillä projektin vaiheita on helppo seurata Kanban-taulusta ja merkitä valmiiksi, kun päiväkirjan myötä voidaan todeta, että jokin asia on tehty. Kanbanin hyötynä on myös se, että se ohjaa myös päiväkirjan tekemistä rajoittamalla työnkulkua.

Varsinkin prosessikuvauksesta syntyy kattava dokumentaatio, joka talletetaan organisaation dokumentointikäytäntöjen mukaisesti. Itse robottikoodia versioineen hallinnoidaan organisaation GitHubissa. Päiväkirjaa hallinnoidaan Microsoft Planner -projektinhallinta työkalussa, joka toimii samalla koko projektin hallinnan työkaluna.

Päiväkirjamerkinnät koostuvat työ- ja kehitystehtävien kuvaamisesta sekä etenemisestä. Päiväkirjassa kuvataan tavoitteita, havaintoja sekä suunniteltuja tehtäviä sekä sitä, kuinka ne toteutuivat. Viikkotason analyysissa taas arvioidaan viikon aikana kertynyttä tekstiä ja oman osaamisen kehittymistä, sekä reflektoidaan tehtyjä asioita tieto-osioon. Lisäksi arvioidaan työn edistymistä, viikon onnistumisia ja kehityskohteita ja miten niissä aiotaan jatkossa toimia. (Päiväkirjamuotoinen opinnäyte Karelia-ammattikorkeakoulussa, 2021.)

Tässä opinnäytetyössä ei näillä ole erityisiä GDPR-vaatimuksia eikä anonymisointitarpeita. Aineistonhallinta tapahtuu toimeksiantajan Microsoft -pilvitallennustilassa Sharepointissa sekä Microsoft Planner -projektinhallintatyökalussa.

Opinnäytetyön tuloksena syntyvän valvontarobotin myötä valvontaan käytettävä henkilötyöaika vähenee ja verkkopalveluiden luotettavuus kasvaa. Valvontaprosessin selkenemisen myötä myös poikkeustilanteet saadaan reaaliaikaisemmin tietoon. Robotin tekemä valvonta myös tuottaa säännöllisempää, täsmällisempää ja reaaliaikaisempaa seurattavaa raporttidataa, jota voi hyödyntää palvelun laadunvarmistamisessa ja kuvaamisessa asiakkaalle.

4 Kehittämisprojektin toteutus päiväkirjamuodossa

4.1 Viikko 1

Viikon aikana aloitettiin Robot Framework -koodin tekeminen. Alkutavoitteena oli, että kaikki valvottavat kohteet ovat jossakin tiedostossa, josta ne luetaan robotin käyttöön. Tiedostoksi valikoitui tässä kohtaa Excel, koska koettiin, että tämän muotoista tietokantaa olisi helppo muokata. Kohdelistassa on neljä saraketta: Index luku, kohteen nimi, toimenpiteet ja IP-osoite. Esimerkkirivinä tästä on "2, S1, ping, 65.109.183.75”

Robot Framework vaatii Excel-kirjaston toimiakseen. Se asennettiin komentorivillä komennolla *pip install robotframework-excellib*, minkä jälkeen kirjasto piti vielä tuoda eli importata Robot Frameworkin käyttöön.

Koodin tavoitteena on, että kohdelistaa pystyy muokkaamaan myös jälkikäteen, kun valvontakohteet voivat muuttua. Tätä varten pitää luoda silmukka, ja silmukkaa varten tarvitsee eritellä muutama muuttuja, esimerkiksi rivien määrä. Kuvassa 1 näkyvässä koodissa haetaan Index-sarakkeen sisältö (col_num=1) ja lasketaan alkioiden määrä.

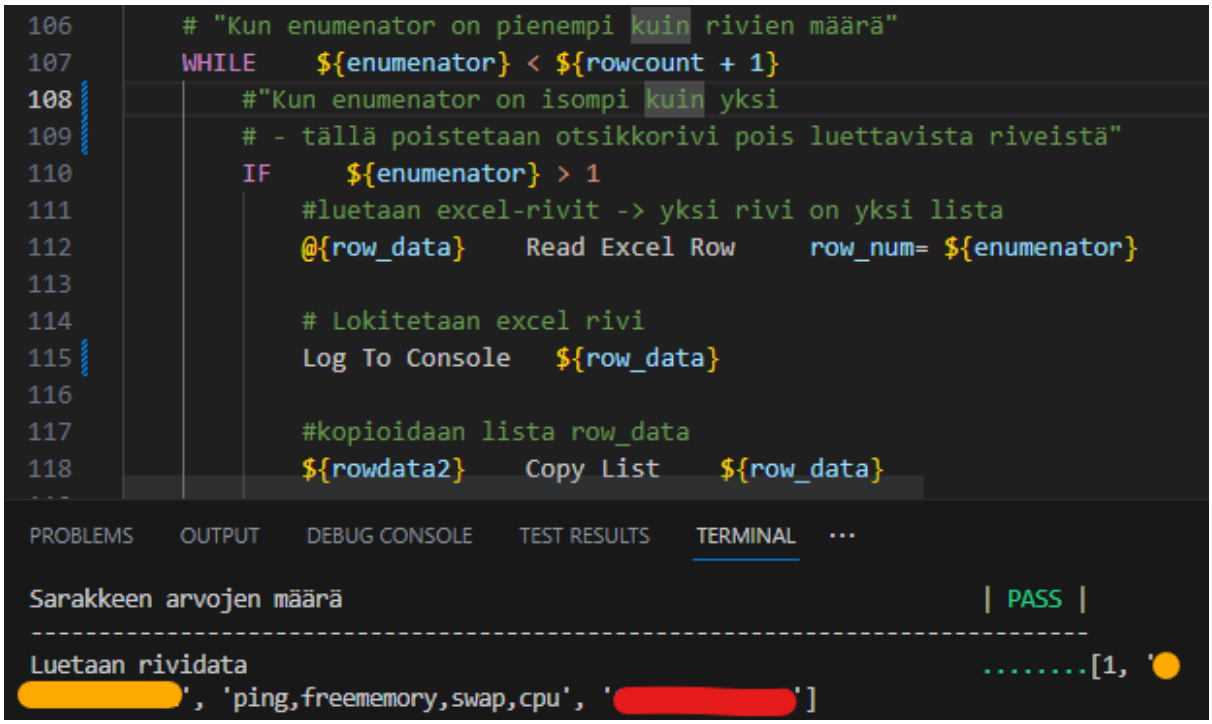
Kuva 1. Lasketaan sarakkeessa olevien alkioiden määrä

```
*** Test Cases ***
Run | Debug | Run in Interactive Console
Avataan excel-tiedosto
    Open Excel Document    ${path}/serverlista.xlsx    doc_id=serverlista
Run | Debug | Run in Interactive Console
Luetaan sarakedata
    ${columndata}    Read Excel Column    col_num=1
    Log    ${columndata}
    #Log To Console    ${columndata}

    ${lista}    Convert To List    ${columndata}
    Log    ${lista}
    Set Global Variable    ${lista}

Run | Debug | Run in Interactive Console
Sarakkeen arvojen määrä
    ${rowcount}    Get Length    ${lista}
    Set Global Variable    ${rowcount}

```

Kun meillä on rivien määrä tiedossa, voidaan lähteä rakentamaan silmukkaa eli loop-lausetta. Loin saatteistajan, muuttujan nimeltä ${enumenator}, arvolla yksi. Tällä ikään kuin numeroidaan rivit. Lisäksi käytän aiemmin luotua globaalia muuttujaa rowcount, jossa on määritelty sarakkeen rivien määrä. Yritän selvittää logiikkalauseen "Niin kauan kun enumeraattori on pienempi kuin (rivienmäärä + 1), tapahtuu asia X" Koodipätkä näyttää tältä: WHILE ${enumenator} < ${rowcount + 1} - tapahtuu jotain.

Seuraavaksi luotiin IF-lause, jolla poistetaan Excel-taulukon otsikkorivi luettavista "Jos enumeraattori on isompi kuin yksi" eli: IF ${enumenator} > 1 Sitten luetaan yksi rivi kerrallaan Read Excel Row -komennolla ja annetaan row_num -muuttujaksi enumeraattori. Log to Console -riville tulostuu: [1, '[serverin nimi]', '[halutut toimenpiteet]', '[valvottavan kohteen IP-osoite]']. Kuvassa 2 näkyy koodi sekä konsoliin tulostunut rivi, josta on keltaisella piilotettu valvottavan kohteen nimi, ja punaisella valvottavan kohteen IP-osoite.

Kuva 2. Luetaan Excel-rivin tiedot while-loopilla



IF-lauseen lopun (IF-lause loppuu END-sanaan) jälkeen vielä kirjoitetaan While-silmukan loppuun pätkä, joka kasvattaa enumeraattoria jokaisella kierroksella yhden niin, että silmukka siirtyy joka kierroksella uudelle riville: ${enumenator} Set Variable ${enumenator+1}.

Robotin tehtävä on lukea toimenpidesaraketta, jonka perusteella se tekee tarvittavat testit kullekin kohteelle. Teen siis muuttujat ${toimenpide} ja ${kohde}, jolla siis tarkoitetaan IP-osoitetta. Nämä löytyvät @{row_data} listasta indekseillä [2] ja [3]. Lisäksi tein muuttujan ${kohdeNimi}, joka oli ${row_data}[1], selkeyttääkseni tuloksien lukemista, sillä serverin nimi on jokseenkin helpompi tunnistaa kuin IP-osoite.

Logiikkalauseena on "jos toimenpide on ping, suoritetaan ping kohteelle". Aluksi lähdin miettimään, miten kahta merkkijonoa (string) vertaillaan Robot Frameworkissa. Löysin lausekkeen "Should be Equal As Strings", joka testaa, että ovatko merkkijonot samoja. Tämän testin soveltaminen IF-lauseeseen ei kuitenkaan onnistunut.

Olin ajatellut että "<", ">" ja "==" tyyppisiä vertailuita voidaan käyttää ainoastaan lukuarvoihin. Painin tämän kanssa varsin pitkään, ja lopulta käytin kuitenkin näitä luoden lausekkeen IF ${toimenpide} == ${ping}". Ilmeisesti Robot Framework ei tunnista näitä tekstipätkiä muuttujiksi ilman lainausmerkkejä, ja kun muuttujat laitettiin lainausmerkkeihin "${toimenpide}" == "${ping}", IF-lause toimi, sillä se korvasi toimenpiteen ja kohteen muuttujilla, jotka oli yllä asetettu.

Logiikkalause jatkuu "... suoritetaan ping kohteelle". Tähän käytän komentoa Run and Return RC and Output koodirivin näyttäessä tältä: ${pingTulos} Run and Return RC and Output ${toimenpide} ${kohde}, jossa komento tallentaa kohteelle tehdyn toimenpiteen muuttujaan nimeltä ${pingTulos}. Muuttuja asetetaan globaaliksi muuttujaksi, jolloin sitä voidaan hyödyntää vielä myöhemmin mahdollisesti eri testeissä. Tämän jälkeen tulee iso määrä tekstinkäsittelyä, kun halutaan jotain tiettyä tuloksesta tallentaa, kuten tapauksessa lukuarvo.
Kuvassa 3 näkyy koodattu ping-testin alkuosa, jossa myös käsitellään merkkijonoa niin, että lopulta ${pingAverage} muuttujaan tallennettu arvo on vain luku, esimerkiksi ”13”, joka on vertailukelpoinen myöhemmin.

Kuva 3. Ping-testin palauttaman merkkijonon käsittely

```
# toimenpide sarakkeessa toimenpiteet pilkulla eroteltu? ping, jotain muuta ”jos ping löytyy täältä sisältä”
# #jos toimenpide on ping, tehdään ping-testi. Logiikka ”jos ping löytyy täältä sisältä”
    IF    "${ping}" in "${toimenpide}"
        ${pingTulos}    Run and Return RC and Output    ${ping} ${kohde}
    # Jos pingtulos sisältää "Average", ping ok tai ping hidas
    # Jos viesti ei sisällä "Average" -> ping testi epäonnistui
        IF    "Average" in "${pingTulos}"
            # Muunnetaan pingTulos Stringiksi (vasta tässä kohtaa, että IF-testi toimii)
            ${pingTulos2}    Convert To String    ${pingTulos}
            #Splitataan pingTulos oletuserottimella ,
            ${pingTulos2}    Split String    ${pingtulos2}
            #Lasketaan alkioiden määrä
            ${Tulos2count}    Get Length    ${pingtulos2}
            #Etsitään pingtestin averagen tulos
            ${pingAverage}    Set Variable    ${pingTulos2}[56]
            #trimmataan pingAverage niin, että tulokseksi jää vain luku
            ${pingAverage}    Replace String    ${pingAverage}    ms')    ${EMPTY}
```

Yhteenvetona koodilta vaadittiin

- Kohdelistaa pitää pystyä muokkaamaan
- Robotti lukee toimenpidesarakkeen ja tekee sen perusteella tarvittavat testit
- Robotti tekee useampia erilaisia testejä

Tällä viikolla koodattiin paljon, ja asensin useampiakin erilaisia kirjastoja komentorivillä. Logiikkalauseet olivat jokseenkin omatekoinen keksintö, mutta koin ne helpottaviksi esimerkiksi koodia kommentoidessani ja dokumentoidessa. Tällä viikolla opin myös paljon merkkijonojen käsittelystä. Muuttujien laittaminen lainausmerkkeihin oli toimintatapa, jota en olisi millään keksinyt itse. Avun kysyminen muilta tuottaa monesti hyviä vastauksia sekä uusia näkökulmia, kun itse on jäänyt jumiin ajatuksissaan.

Mitä tekisin eri tavalla: ${kohde} muuttuja olisi pitänyt nimetä ehkä ${ipAddress}, koska se olisi kuvannut paremmin. Lähdin myös suoraan kirjoittamaan koodia, sen sijaan että olisin suunnitellut kokonaisuutta paremmin. Tästä ei vielä seurannut ongelmia, mutta riski sille on isompi tällä tavalla. Suuri osa tehtävälistasta on ainoastaan minun päässäni, sen sijaan että se olisi kirjattu Kanban-taululle. Muuttujien määrittelyissä tehtiin vähän turhaa työtä, sillä ei ollut tarve kopioida listaa ja tehdä siitä erilaisia merkkijonoja. Jatkossa tiedän, että listan muuttaminen erillisiksi merkkijonoiksi ei ole välttämätöntä.

4.2 Viikko 2

Tällä viikolla selvitin teoriaa tietohallinnon, tietojärjestelmän ja valvonnan käsitteistä. Tämä oli minulle hyödyllistä luoda omaa käsitehierarkiaa siitä, mikä on valvonnan suhde tietohallintoon. Tietohallinto on kokonaisuuden hallintaa ja valvonta vain yksi osa tietohallintoa. Tietohallinto myös määrittelee tarkemmin valvontaan liittyvät mittarit. Syvennyin saatavuuden valvontaan ja erityisesti minulle uutena tuli esiin tietohallintoon kuuluvat saatavuustavoitteet ja vaatimukset. Nämä pitäisi ottaa osaksi opinnäytetyötä ja tuottaa dokumenttia siitä, mitkä nämä saatavuustavoitteet ovat valvomilleni laitteille.

Tässä kohtaa projekti on vielä aika alussaan, ja minun on jonkun verran tehtävä tämänkaltaista perehtymistä saadakseni kokonaiskuvaa. Tämän myötä minun olisi pitänyt muokata Kanban-taulua, joka on varsin ylätasoinen lista tehtävistä asioista.

4.3 Viikko 3

Kokonaisuus ja se, mitä halutaan robotilla saada aikaiseksi, on jo paremmin nähtävissä. Pääajatuksena on, että robotti kerää listan testien tuloksista ja tallentaa ne tiedostoon, ja lisäksi lähettää huomioarvon ylittävistä tiedoista viestin vielä erikseen esimerkiksi Slackiin.

Hahmottelimme toimeksiantajan kanssa hieman saatavuussuunnitelmaa. Määriteltiin Ping-testille aika ja validaatio: 80 ms. Epäonnistunut ping-testi ja tuon ylittävä arvo pitäisi saada johonkin hälytykseksi, esimerkiksi organisaation Slackiin Webhookilla. Tarkistettiin, että se on Robot Frameworkilla mahdollista. Se, mitä muuta testataan, on vielä auki. Periaatteessa CPU:n testaamisen hyödyllisyyttä ei nähty, mutta curl-toimintoa mietittiin. Tehtiin projektille uutta rajausta ja todettiin että curl ei nyt ole palvelinten valvonnassa merkittävä, kun palvelimille tehdään jo ping-testi.

Teknisesti kuvan 4 koodi toimii ja koin onnistumisen tunteita siitä, että pystyin tekemään itse tämän koodin, se oli kuitenkin pidemmän päälle ongelmallinen logiikaltaan. Ping-validaattori alkoi viemään koodia väärään suuntaan, sillä yhtäkkiä minulla oli kuvan 4 mukaisesti ping-tulokselle kolme eri muuttujaa, riippuen siitä oliko ping onnistunut vai ei, ja jos ping oli alle tai yli 80. Tämä

oli vaikea hallinnoida ja huomasin melko myöhään, että tämä ei ollut koodin kannalta onnistunut valinta. Tätä kohtaa olisi pitänyt työstää vasta myöhemmin, kun kaikki valvottavat kohteet ovat selvillä ja mietitään mitä tuloksilla tehdään.

Kuva 4. Ping-tuloksen erilaiset muuttujat

```
            # "Jos tulos on pienempi kuin pingValidator, tulos on ok, muuten hidas "
            IF    ${trimAverage} < ${pingValidator}
            #Tehdään tuloksesta muuttuja (lista, joka konvertoidaan stringiksi, jos myöhemmin konvertoidaan string vielä JSON:niksi)
                ${resultPingOk}    Create List    ${DateTime} ${kohdeNimi} ping ok ${trimAverage} ms
                ${resultPingOk}    Convert To String    ${resultPingOk}
                Log To Console    ${resultPingOk}
            ELSE
            #Tehdään tuloksesta muuttuja (lista, joka konvertoidaan stringiksi, jos myöhemmin konvertoidaan string vielä JSON:niksi)
                ${resultPingSlow}    Create List    ${DateTime} ${kohdeNimi} ping hidas ${pingAverage}
                ${resultPingSlow}    Convert To String    ${resultPingSlow}
                Log To Console    ${resultPingSlow}
            #tämän pitäisi hälyttää jonnekin
            END

ELSE
    #Tehdään tuloksesta muuttuja (lista, joka konvertoidaan stringiksi, jos myöhemmin konvertoidaan string vielä JSON:niksi)
    ${resultPingFail}    Create List    ${DateTime} ${kohdeNimi} Ping Test Failed
    ${resultPingFail}    Convert To String    ${resultPingFail}
    Log To Console    ${resultPingFail}
    #Log To Console    ${DateTime} ${kohdeNimi} Ping Test Failed
```

Tällä viikolla lähinnä sovellettiin jo opittua, edelleen Kanban-taulun päivitys on jäänyt tekemättä ja projektin sisäinen prosessi esimerkiksi viikoittaisesta projektitaulun päivittämisestä olisi ollut hyödyllistä, sillä tehtävät olisivat tarkentuneet.

4.4 Viikko 4

Tällä viikolla aloin kokoamaan Exceliin taulua, jonka tehtävä olisi havainnollistaa eri testejä ja sitä, mitä mitataan. Alun perin ajattelin, että tämä toimii oman ajatteluni ja työni tukena havainnollistaen samalla hallinnon ja valvonnan eroja, mutta mitä pidemmälle taulukko eteni, sitä enemmän ymmärsin tämän sovellettavuuden valvontasuunnitelmaksi. Tästä kehittyi taulukko, joka johdettiin sitten valvontasuunnitelmaksi (Liite 2). Valvontasuunnitelma esitellään paremmin johtopäätöksissä.

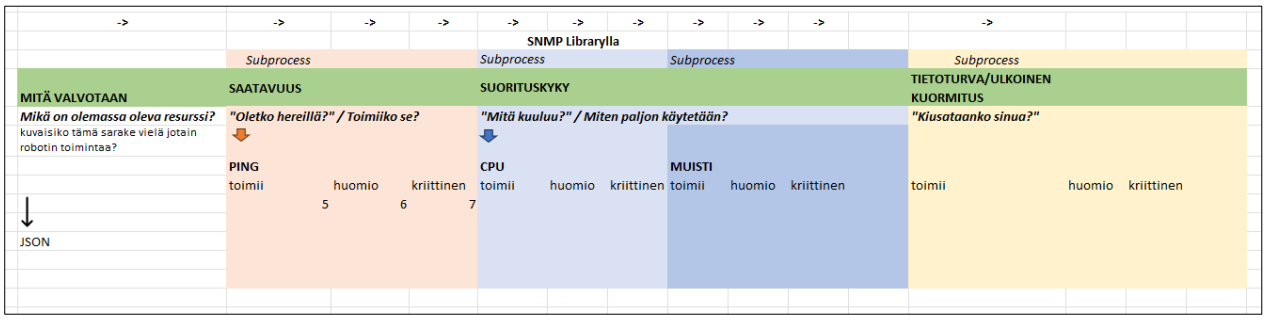
Tämä kuvaa valvontasuunnitelmaa enemmän strategisella tasolla, sillä siinä kysytään ”mitä, miksi, miten?” Kaksi viimeistä saraketta olen jättänyt tyhjiksi, sillä tämän suunnitelmakehikon avulla käydään jokainen kohde läpi erikseen ja määritellään testifrekvenssi ja hälytysrajat valvontakohteittain.

Taulukon strategiset sisällöt on koottu teoriasta ja organisaation tarpeista. Teoriassa mainitaan esimerkiksi useita erilaisia suorituskyvyn valvontakohteita, ja niitä voidaan vielä jatkokehittää tulevaisuudessa, mutta tähän opinnäytetyöhön on nyt valittu kaksi kohdetta: CPU ja muisti.

On monta erilaista tapaa jäsennellä näitä käsitteitä, esimerkiksi kaikki voisivat mennä myös IT-infrastruktuurin alle, mutta itse olen nyt päättänyt käsitellä IT-infrastruktuuria sen fyysisenä osiona, jolloin voitaisiin mitata sen lämpötilaa.

Suunnitelman pohjalta lähdin rakentamaan prosessikaaviota. Tässä kohtaa se on nyt luotu Exceliin, koska tässä on helppo myös ottaa huomioon valvontakohtaiset tavoitteet, joiden kuvaaminen vuokaaviossa on hieman vaikeampaa.

Kuva 5. Ensimmäinen prosessikuvauksen suunnitelma



Alun perin prosessikaavio näytti kuvan 5 mukaiselta. Päädyimme kuitenkin mittaamaan IT-infrastruktuuria, eli lähinnä ulkoista kuormitusta, emmekä tietoturvaa. Täydensin prosessisuunnitelmaa kuitenkin vielä reippaasti (liite 3). Tästä olisi myös kohtalaisen helppo muotoilla vuokaavio tämän perusteella. Prosessikaaviota käsitellään enemmän vielä johtopäätöksissä.

Teorian mukaan valvonnalla yritetään saada vastaus seuraaviin kysymyksiin: mikä on valvottava kohde, toimiiko se, ja miten hyvin se toimii? Ihmisläheisenä ihmisenä aluksi kysymykseni olivat muotoa ”oletko hereillä? Mitä kuuluu? Miten voit?” Nämä kysymykset olivatkin samalla hyvin teoriaan sopivia, joten olen jättänyt ne myös osaksi prosessikaaviota. Siinä on hyvä ’tarinallistamisen maku’, mikä tekee prosessikaaviosta helpomman käsitellä.

Prosessikaavion tekeminen oli mielestäni varsin kiinnostavaa ja helppoa. Oli oikeastaan lähinnä vanhan, jo olemassa olevan osaamisen soveltamista, mutta olin todella tyytyväinen tulokseen. Itse hälytysten raja-arvot sekä saatavuustavoitteet jäivät vielä vajavaisiksi. Tässä olisi varmasti myös mahdollista ottaa kantaa valvontafrekvenssiin, mitä ei vielä tässä kohtaa ole tehty.

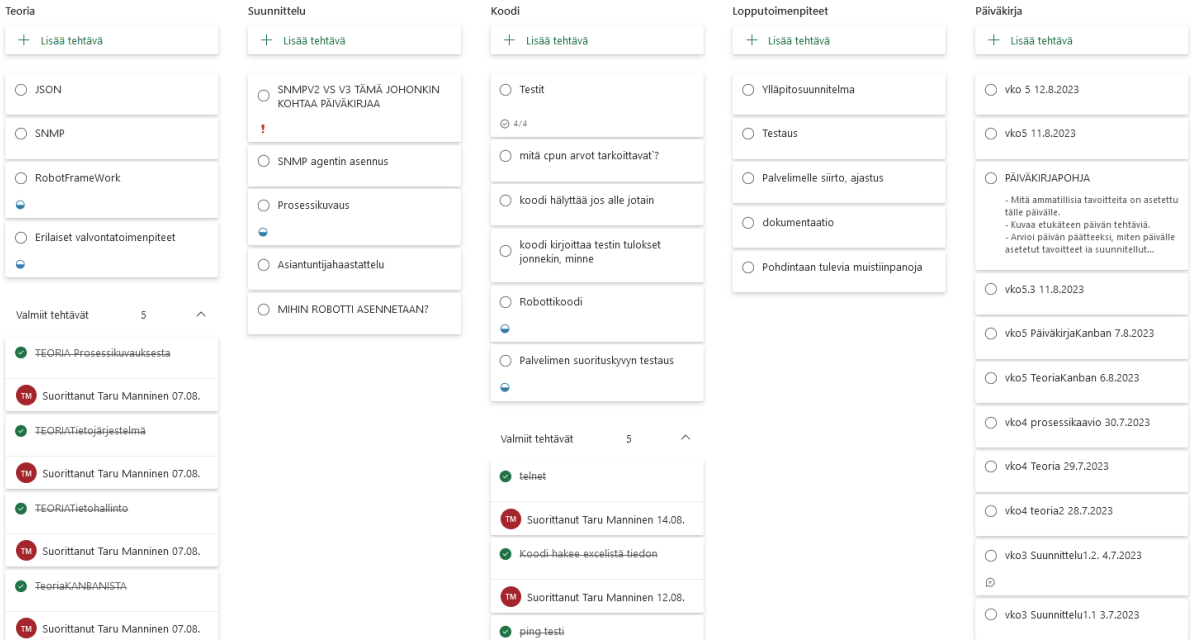
4.5 Viikko 5

Kirjoitin teoriaa Kanban-menetelmästä ja aloin arvioimaan oman projektini onnistumista tähän mennessä. Sain tavallaan synninpäästön sille, että oma Kanban-tauluni ei ole minkään yleisen kuvauksen mukainen, sillä sellaista yleistä kuvausta Kanban-taulusta ei olemassakaan.

Oman arvioni mukaan en ole kuitenkaan onnistunut visualisoimaan Kanban-taululla riittävän tarkasti tätä projektia. Kiinnitin tähän huomiota ja yritin luoda Kanban-tauluun tarkemmin projektin osia sekä merkitsemään valmiiksi kohtia, jotka olin tehnyt. Keskeneräisinä tällä hetkellä on pari teoriaosuutta ja testausprosessien teko sekä prosessikuvauksen luonti.
Kanban-menetelmän ’nyrkkisäännön’ mukaan ei kannata kasvattaa keskeneräisten tehtävien määrää liikaa, joten keskityn edistämään näitä.

Kuvassa 6 näkyy Microsoftin Planner-työkalulla luotu Kanban-taulu, jossa on pystytty merkitsemään osa tehtävistä valmiiksi ja osa keskeneräisiksi. Osa tehtävistä on täysin aloittamatta. Alkuperäinen ajatukseni on ollut tehdä ylätason jako teoriaan, suunnitteluun, koodiin, lopputoimenpiteisiin ja päiväkirjaan, ja teoriassa ainakin aikataulullisesti työn olisikin pitänyt edetä näin.

Kuva 6. Projektinhallinnan kanban-taulu



Ajatukseni oli, että Kanban-taulu huomioisi teoriaan kirjoitettavat asiat omana kokonaisuutenaan, kun suunnittelun taas näin pitävän sisällään juuri prosessikuvauksen ja valvontasuunnitelman. Koodi-taulu taas pitää sisällään Robot Frameworkissa tehtävät asiat. Lopputoimenpiteissä on hieman sekalaista tavaraa, joka liittyy osittain suunnitteluun sekä kehitystyön pohditaan. Viimeisenä sarakkeena on päiväkirja, jota olen pyrkinyt hallinnoimaan myös tässä työkalussa.

Olen pitänyt Kanban-menetelmää silti hyvänä valintana, koska projektin eri osia on ollut mahdollista edistää tasaisesti eteenpäin, vaikka tiettyjä riippuvuuksia aikajanassa teoria-suunnittelu-koodi onkin nähtävissä.

Kanbanin lisäksi tällä viikolla jatkoin Robot Framework -koodia. CPU:n testaamisen myötä tutustuin SNMP-nimiseen työkaluun, jota käytetään usein valvonnassa. Myös JSON-muotoinen saadun tiedon vakiointi onnistuu Robot Frameworkilla. Robot Frameworkille löytyy SNMP-kirjasto, jonka latasin käyttöön. SNMP-kirjaston kanssa syntyi heti ongelmia, koska pyasm1 0.5 ei ollut yhteensopiva SNMP-kirjaston kanssa. Poistin sen ja latasin pyasm1 0.4.8, minkä jälkeen ohjelma toimi. OID:en, eli Object Identifier-tunnisteen kanssa oli aluksi myös haasteita, sillä sen muotoilu ei alun perin viitannut siihen, että tunniste vaatii pisteen eteen.

- Väärä muotoilu: ${memTotalReal} 1.3.6.1.4.1.2021.4.5
- Oikea muotoilu: ${memTotalReal} .1.3.6.1.4.1.2021.4.5

SNMP:n myötä kävi ilmi, etteivät kaikki SNMP:n OID:t toimi samalla tavoin, jos laisinkaan Linux- ja Microsoft-palvelimilla. Tämä oli todella tärkeä havainto, mikä olisi pitänyt ottaa jotenkin huomioon testivaiheessa. Sanoisin jopa, että alkuperäiseen Exceliin, josta tiedot luetaan, voisi lisätä ”malli”-nimisen sarakkeen kuvan 7. esimerkin mukaisesti. Tätä tietoa voi tarvita, kun koodia muokataan.

Kuva 7. Esimerkki tietotaulusta, josta testattavat kohteet haetaan

testi id	nimi	toimenpide	osoite	malli
1	A7	ping	12.345.678.99	linux
2	2CU	ping	98.76.54.32	linux

Latasin myös DateTime Libraryn ja hain testille päivämäärän komennolla ${DateTime} Get Current Date.

4.6 Viikko 6

Tällä viikolla selvitin SNMP agentin, hallintajärjestelmän ja MIB:in eroja. Aloin ymmärtämään SNMP:n toimintaa paremmin ja soveltamaan sitä käytännössä. En ajatellut käyttää SNMP:n tarjoamia traptoimintoja, vaan ainoastaan Get ja Walk -toimintoja, joilla haen tiedot Robot Frameworkin käsiteltäviksi.

Oma roolini valvontarobotin pilotoijana vahvistui. Sovellutuksena saman robotin voidaan tulevaisuudessa odottaa lukevan tietoja myös kytkimiltä ja reitittimiltä, joten siinäkin mielessä on hyvä, että lähtötietoihin lisättiin malli-sarake (kuva 7).
Aluksi on avattava SNMP-yhteys, ja tähän liittyi tietoturvariski. Meille käyttöön valikoitui SNMP V2c -yhteys, jonka syntaksi on seuraava:

```
host, community_string=None, port=161, timeout=1.0, retries=5,
alias=None
```

Hostilla tarkoitetaan kohteen IP-osoitetta, mutta itse community_string, eli yhteisöavain, tekee tästä haavoittuvan, sillä se on oletuksena ”public”. Tätä on organisaatiossamme muokattu niin, ettei oletusyhteisöavaimella pääse palvelimelle. SNMP V3c -yhteys olisi kaikista tietoturvallisin, koska se vaatisi oikean tunnistautumisen. Tässä yhteydessä organisaatiossa koetaan kuitenkin, että SNMP V2c on riittävän turvallinen, sillä SNMP:ssä määritellään tilaksi view only, tiedot ja niiden kohteet eivät ole kriittisiä ja lisäksi myös organisaation palomuuriasetuksissa on määritelty, mistä palvelimiin voi yhdistyä. Lopuksi SNMP-yhteys on myös suljettava.

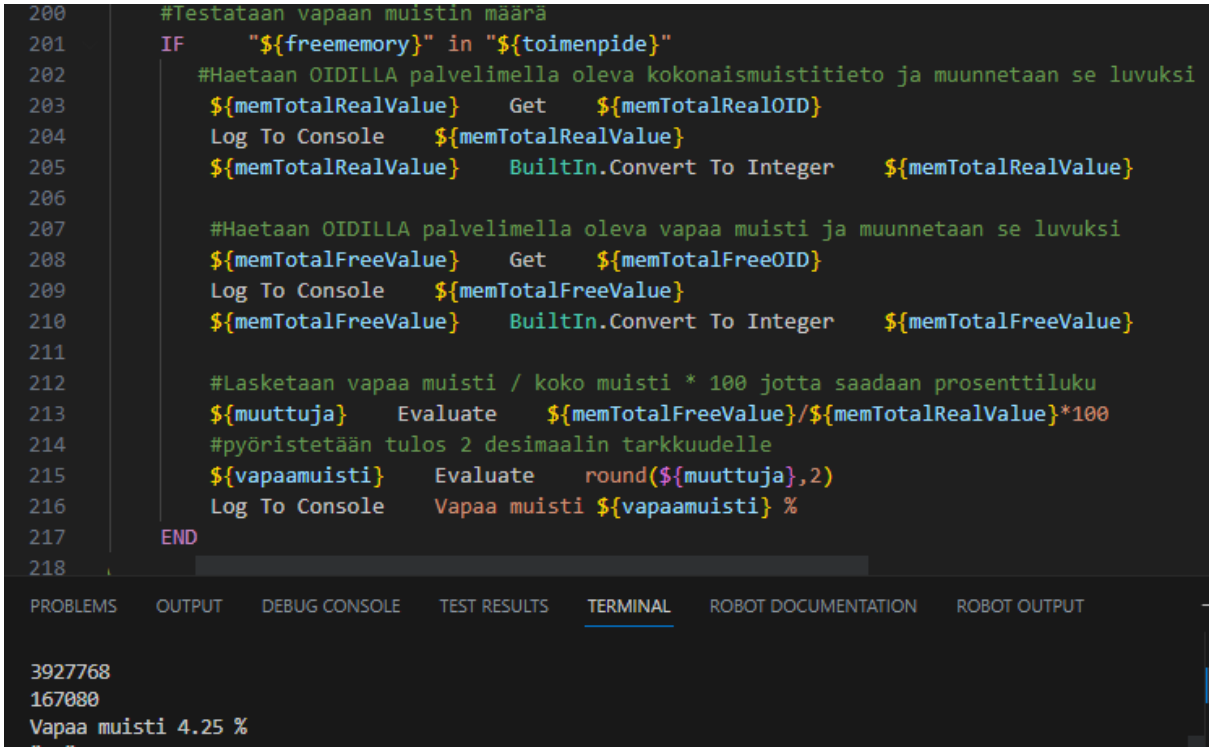
Kirjoitin useita SNMP-kirjaston avulla toteutettuja testejä muun muassa palvelimen muistiin liittyen. Koodissa logiikka on muotoa ”jos muisti mainitaan toimenpidesarakkeessa, tehdään seuraavat asiat”. Tämän myötä lähtötietoja muokattiin niin, että toimenpidesarakkeessa on useampia toimenpiteitä, kuten kuvassa 8 näkyy. Uutena tietona verrattuna kuvaan 7 on toimenpidelistassa näkyvät ”ping” lisäksi ”freememory,” ”swap” ja ”cpu”.

Kuva 8. Täydennetty esimerkki tietotaulusta, josta testattavat kohteet haetaan

testi id	nimi	toimenpide	osoite	malli
1	A7	ping,freememory,swap,cpu	12.345.678.99	linux
2	2CU	ping,freememory,swap,cpu	98.76.54.32	linux

En löytänyt sellaista OID:ta, joka kertoisi suoraan vapaan muistin määrän prosentteina, joten hain kaksi tietoa palvelimelta: Muistin kokonaismäärän ja vapaan muistin määrän. Tämän jälkeen kuvassa 9 näkyvällä tavalla laskin näistä kahdesta vapaan muistin prosentuaalisen osuuden.

Kuva 9. Vapaan muistin laskeminen kahdesta arvosta

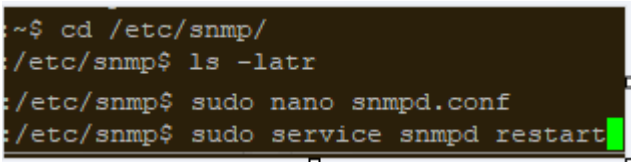


4.7 Viikko 7

Projektin etenemisen suhteen alkupään metatyön tärkeyttä ei voi väheksyä, sillä on todellakin todettava, että Kanban-taulu voisi olla palkitsevampi työkalu, jos sieltä saisi todella merkittyä tehdyiksi hyvin selkeitä asioita. Toisaalta ketteryyteen viittaa myös se, että en ole esimerkiksi merkinnyt kaikkia teoria-alueita valmiiksi, koska uskon, että näihin voisi vielä tulla lisättävää. SNMP:n CPU- testien koodaaminen jatkuu. Opettelin konfiguroimaan itse palvelimella SNMP:n OID-lukuja. Itse oikeiden OID-lukujen löytäminen on ollut todella vaikeaa, ja olin tässä jumissa koko viikon.

Opin jonkun verran Linux-komentokehotteita, kun kävin itse palvelimella muokkaamassa konfiguraatiota. Kuvassa 10 näkyy käytetyt komentokehotteet. Näiden komentokehotteiden käytön opettelu sujuvoittaa työskentelyä ja tuo itsevarmuutta työskentelyyn.

Kuva 10. snmp.conf konfiguraatiossa käytetyt komentokehotteet



4.8 Viikko 8

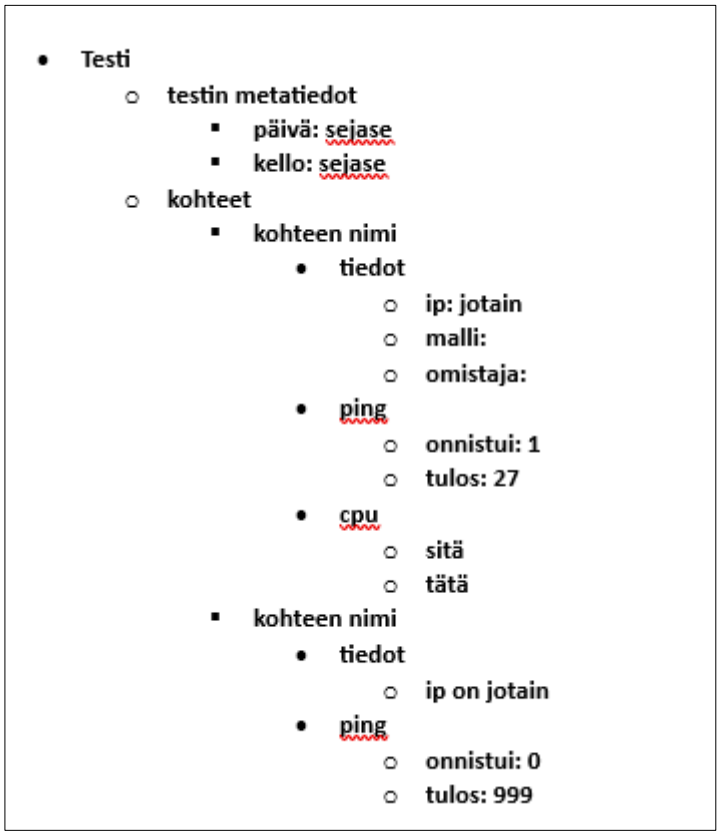
Tavoitteena oli selvittää, miksi CPU:ta ei pysty mittaamaan. Epäilin kesällä vaihdettua palomuuria syyksi. Kuitenkin erilaisia OIDE:ta yritä ja erehdy -menetelmällä kokeillen sain viimeinkin yhden OID:in toimimaan. OID numerolla .1.3.6.1.4.1.2021.11.11 toimii ja tuottaa minulle prosenttiluvun siitä, kuinka monta prosenttia viimeisen minuutin aikana CPU on ollut toimettomana.

Kävin konfiguroimassa palvelimella olevia OID:ja niin, että sinne jäi vain yksi ylätason Object Identifier: .1.3.6.1.4.1.2021.11. Tämän myötä kaikki CPU OID:it alkoivat toimimaan: kun lisää viimeisen .11 perään .9 tai .52, Robot Framework antoi tuloksia. Osittain myös selvisi, että osa OID:sta oli vanhentuneita ja tuottivat siksi arvon 0.

Tämän jälkeen tein useita CPU testejä, jotka sain toimimaan, vaikkei niiden kaikkien merkitys nyt olekaan järin suuri. Lisäksi tein testin sille, kauanko järjestelmän edellisestä käynnistämisestä on aikaa. Näiden lukeminen on oma haasteensa, johon pitää palata vielä myöhemmin.

Suunniteltiin JSON -rakennetta ja päädyttiin kuvan 11 kaltaiseen suunnitelmaan. Tämä oli varsin helppoa, sillä JSON- syntaksi on varsin yksinkertainen. Tässä suunnitelmassa ei näy vielä varsinainen tulos.

Kuva 11. JSON-rakenteen suunnitelma



Tämän myötä huomasin, että olen aikaisemmin tehnyt liikaa muuttujia kuten alun ping-testissä. Ping-testin muuttujia piti muokata niin, että testi palauttaa vain kaksi arvoa: onnistuiko ja tulos. Kuvassa 12 näkyy uusi koodi, jossa nämä kaksi muuttujaa entisten kolmen muuttujan sijaan. Tuota ${pingResultValue} muuttujaa tulisi nyt verrata 80 ms validaatoriin. Projektinhallinnan näkökulmasta koin tämän pienenä epäonnistumisena, sillä jos olisin alun perinkin tiennyt, mitä haetaan, olisin alun perinkin ehkä ymmärtänyt koodata asiat niin, ettei olisi tarvinnut takautuvasti korjata koodia.

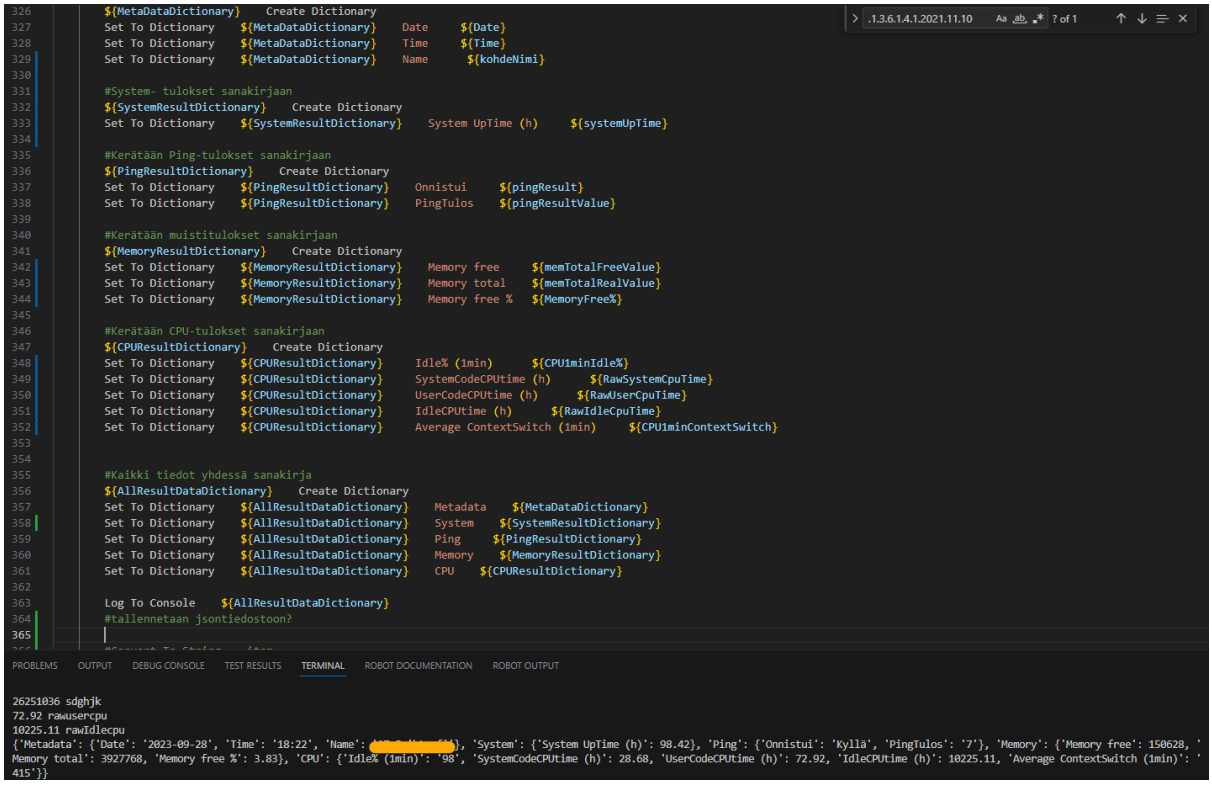
Kuva 12. Korjattu Ping-testi tuloksen muuttuja

```
140             IF    "${ping}" in "${toimenpide}"
141                 ${pingTulos}    Run and Return RC and Output    ${ping} ${kohde}
142             # Jos pingtulos sisältää "Average", ping ok tai ping hidas
143             # Jos viesti ei sisällä "Average" -> ping testi epäonnistui
144                 IF    "Average" in "${pingTulos}"
145                     # Muunnetaan pingTulos Stringiksi (vasta tässä kohtaa, että IF-testi toimii)
146                     ${pingTulos2}    Convert To String    ${pingTulos}
147                     #Splitataan pingTulos oletuserottimella ,
148                     ${pingTulos2}    Split String    ${pingtulos2}
149                     #Lasketaan alkioiden määrä
150                     ${Tulos2count}    Get Length    ${pingtulos2}
151                     #Etsitään pingtestin averagen tulos
152                     ${pingAverage}    Set Variable    ${pingTulos2}[56]
153                     #trimmataan pingAverage niin, että tulokseksi jää vain luku
154                     ${pingAverage}    Replace String    ${pingAverage}    ms')    ${EMPTY}
155
156                     #vko8
157                     #muuttuja joka kertoo onnistuiko pingtesti, saa arvon onnistui/epäonnistui, toinen muuttuja joka kertoo numeerisen arvon
158                     ${pingResult}    Set Variable    Kyllä
159                     ${pingResultValue}    Set Variable    ${pingAverage}
160
161                 ELSE
162                 #Tehdään tuloksesta muuttuja (lista, joka konvertoidaan stringiksi, jos myöhemmin konvertoidaan string vielä JSON:niksi)
163                 #muuttuja joka kertoo onnistuiko pingtesti, saa arvon onnistui/epäonnistui, toinen muuttuja joka kertoo numeerisen arvon
164                 ${pingResult}    Set Variable    Ei
165                 ${pingResultValue}    Set Variable    "--"
```

Lisäksi muokkailin aikaisemmin haettua ${DateTime} muuttujaa kahdeksi erilliseksi muuttujaksi: ${Date} ja ${Time}. Lisäksi muokkasin ${Time} variaabelin niin, että se esittää kellonajan vain hh:mm -muodossa, sillä aikaisemmin se kirjasi ylös myös sekunnin sadasosat.
Lopulta tällä viikolla päästiin myös siihen, että tein ensimmäinen kokeilun muokata tulokset JSON-muotoiseksi. Tässä kohtaa selvitin ensin, miten tätä olisi järkevä toteuttaa. Robot Frameworkilla on myös JSON-kirjasto, jossa on toiminto ”convert String to JSON”. Lisäksi on tietotyyppi Dictionary, eli sanakirja, joka tekee lähestulkoon valmiiksi JSON-muotoista tekstiä.

Kuvassa 13 näkyy kaikki sanakirjat, joita olen tehnyt. Kuvassa on piilotettu organisaation nimen sisältävä palvelimen nimi. Ensin olen koonnut kaikki tiedot omiksi sanakirjoikseen: metadata, system, ping, memory ja cpu. Sitten olen luonut sanakirjan, johon kaikki sanakirjat on koottu yhden sanakirjan alle, mikä tuottaa minulle tarpeellisen JSON-hierarkian saamiini tietoihin.

Kuva 13. Tulosten rakentaminen dictionary-muotoiseksi



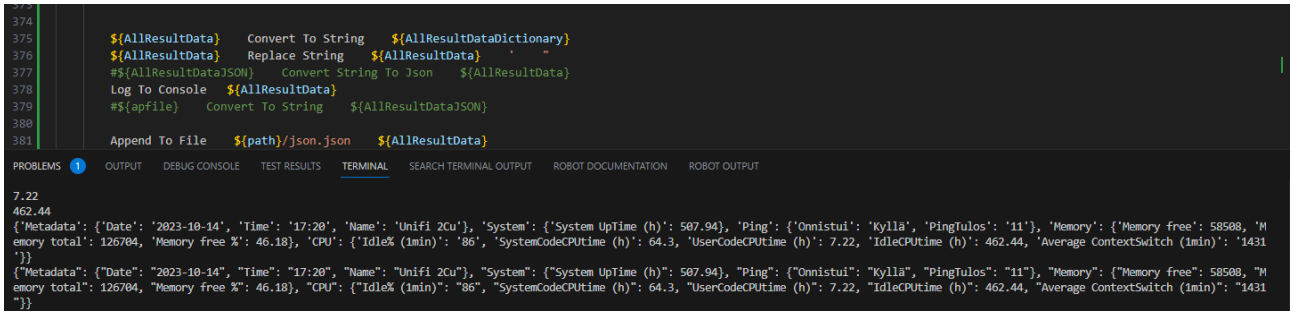
Tällä viikolla Robot Frameworkin koodi osiossa minulle tuli hyvin vähän enää mitään uutta. Sisäkkäiset sanakirjat olivat hieno löytö. Oli ilo soveltaa jo osattua tietoa ja nähdä melko paljon valmista työtä. Lisäksi kokeilin tuota koodia yhdelle organisaation reitittimelle, ja se tuotti luotettavalta vaikuttavia tuloksia, eli jossain määrin ainakin koodin muokattavuus on säilynyt.

4.9 Viikko 9

Tällä viikolla erityisesti haluttiin tallentaa tieto JSON-muotoon. Epäilin, että kuinka vaikea tehtävä on tallentaa JSON-tiedostoon Robot Frameworkista. Paljastui, että se onnistui hyvin helposti: tein vain tiedoston jonka päätteeksi laitoin .json, ja sitten Robot Frameworkissa käytin komentoa: Append To File ${path}/json.json ${AllResultData}.

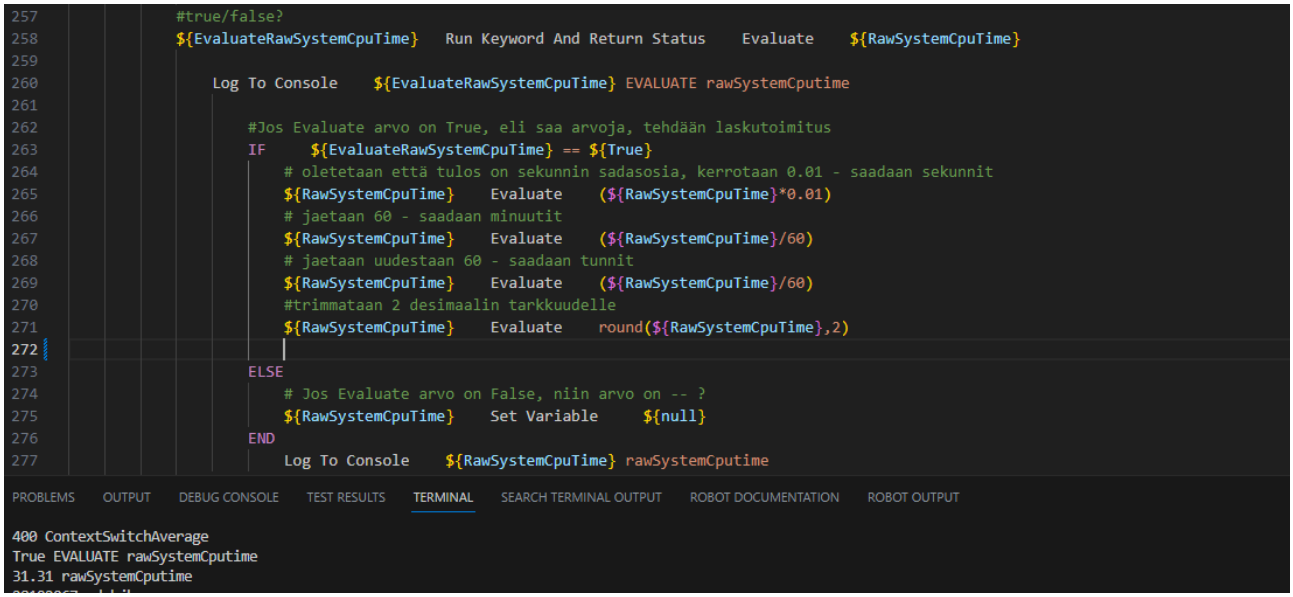
Alun perin koodi tulosti tiedot yksinkertaisten lainausmerkkien sisällä, mistä Visual Studio Code sitten herjasi sanomalla, että tiedostojen pitää olla kaksinkertaisten merkkien sisällä. Ratkaisin tämän kuvassa 14 näkyvällä tavalla, jossa koko sanakirja on muunnettu merkkijonoksi, ja sen jälkeen yksinkertaiset lainausmerkit on korvattu kaksinkertaisilla lainausmerkeillä. Kuvassa näkyy Log to Console -tulostus ennen ja jälkeen konvertoinnin.

Kuva 14. Merkkijonon muunto json- muotoiseksi dataksi



Lisäksi testatessa paljastui, että täytyy luoda jonkinnäköinen true/false testi siltä varalta, että SNMP ei palautak mitään arvoa. Jos SNMP ei palauta mitään arvoa, merkkijonon käsittely aiheuttaa koodissa virheen. Kuvassa 15 näkyy, miten tämä on ratkaistu: Käytetään Evaluate-toimintoa, joka konsolissa tässä tapauksessa palauttaa arvon true. Kun arvo on true, palauttaa koodi myös arvon (tässä tapauksessa 31.31). Jos arvo on false, tulostuu vain muuttujissa määritelty variaabeli ${null}.

Kuva 15. Evaluate-toiminnon käyttö



5 Johtopäätökset ja pohdinta

Tuloksena tästä kehittämisprojektista syntyi valvontasuunnitelma (liite 2), prosessikuvaus (liite3) sekä Robot Framework -valvontarobottikoodi. Robot Framework -koodin olennaisimpia osia on esitelty jo päiväkirjaosiossa.

SNMP-valvontatyökalu oli hyvä löytö. sillä se on todella monikäyttöinen, ja erityisen ilahduttavaa oli, että sitä voi käyttää myös Robot Frameworkissa sitä varten rakennetun kirjaston kautta. SNMP on asennettavissa myös lukuisiin muihin valvontaa vaativiin laitteisiin, kuten kytkimiin, mikä palvelee sitä, että tämä valvontarobotti on mahdollisuuksien mukaan hyödynnettävissä myös laajemmin kuin palvelinten valvonnassa.

Tässä työssä valvonta rajautui saatavuuteen, suorituskykyyn ja IT-infrastruktuuriin organisaation tavoitteiden mukaisesti. Valvontasuunnitelma esitellään liitteessä 2. Se vastaa samalla ensimmäiseen strategiseen tutkimuskysymykseen siitä, mikä on valvonnan merkitys tietojärjestelmien hallinnassa: proaktiivinen ote resurssienhallinnassa on yksi tärkeimmistä. Lopulta tämänkaltainen proaktiivinen ote myös tuottaa asiakastyytyväisyyttä ja vähentää käyttökatkoja, mikä sujuvoittaa kaikkea tietojenkäsittelyä vaativaa toimintaa.

Valvontasuunnitelmassa (liite 2) vastataan myös toiseen tutkimuskysymykseen siitä, mitä erilaisia valvontatoimenpiteitä on ja mikä on niiden merkitys ja tavoite. Tässä yhteydessä valvontasuunnitelma mainitsee pingin ja SNMP:n Object Identifierit. Se vastaa myös muihin hallinnollisiin kysymyksiin, kuten miten tuloksista raportoidaan. Tätä valvontasuunnitelmaa on helppo täydentää tulevaisuudessa. Myös valvottavat kohteet voisi lisätä tähän omaksi sarakkeekseen.

Liitteessä 3 esitellään prosessikaavio. Vasemmassa reunassa kysytään ”mitä valvotaan?”, eli mikä on olemassa oleva resurssi. Tässä sarakkeessa on palvelimet nimeltä mainittuna. Vihreä vaakalinja pyrkii kuvaamaan pääprosessia, eli koko testiprosessia, ja eriväriset pystyrivit taas aliprosesseja eli testejä. Itse testeissä taas vaakarivillä näkyy myös kyseiselle palvelimelle määritellyt tavoitteet tasolla ”toimii, huomio, kriittinen”.

Alaspäin prosessikaavio kuvaa sitä, että kun kaikki toimii tavoitteiden mukaisesti, tieto viedään raporttiin, eikä mitään erityisiä toimenpiteitä tarvita. Raportin on tarkoitus muodostaa pitkällä aikavälillä data, josta voidaan alkaa havainnoimaan erilaisia trendejä. Mikäli huomioraja ylittyy, robotin tehtävä on ilmoittaa siitä alustavan suunnitelman mukaan Slackiin. Tällöin työntekijä voi tarkistaa myös aikaisempia raportteja mutta joka tapauksessa reagoida huomiota vaativaan pyyntöön. Myös huomio ja kriittinen arvon saaneet tapahtumat tulisi kirjautua raporttiin.

Kriittinen arvon saaneet tapahtumat niin ikään kirjataan raporttiin ja ilmoitetaan jonnekin, tässä vaiheessa suunnitelmaa Slackiin. Sen lisäksi tämä vaatii reagointia ja toimenpiteitä. Nämä toimenpiteet eivät ole opinnäytetyön aiheena, joten en käsittele niitä enempää.

Prosessikaavion voi nähdä vastaavan viimeiseen tutkimuskysymykseen: Miten automatisointiprosessi palvelee tietojärjestelmien valvontaa? Siinä havainnollistetaan se, että manuaalinen valvonta vapauttaa ihmisten resursseja muihin asioihin ja ihminen osallistuu ainoastaan hälyttäviin ilmoituksiin ongelmanratkaisijan roolissa. Lisäksi tietojärjestelmien valvonnassa valvottavalle kohteelle on asetettavissa kohdekohtaiset raja-arvot, jotka robotti muistaa aina.

Robot Framework -koodaus muuttui koko ajan helpommaksi ajan myötä, ja kun tähän yhdistettiin teoria muun muassa kirjastoista ja erilaisista muuttujista, itse perusasiat alkoivat olemaan hyvin hallussa. Kun kokonaisuus muuttui selkeämmäksi ja oma toiminta muuttui kokeilevasta toiminnasta yhä enemmän etukäteen suunnittelevaksi, tiedonhaku ja ongelmanratkaisutaidot kehittyivät.

Päiväkirjamuotoinen opinnäytetyö todella kehitti kirjoittajaansa asiantuntijakirjoittajana. Monilta osin metodi tuntui tosin epäselvältä, eikä viikkotason synteesejä aluksi muodostettu metodin mukaan jokaisen viikon päätteeksi. Se jätti tekijälleen myös jopa liian vapaat kädet työn tekemiseen, minkä vuoksi projektiin tuli jälkikäteen paljon korjattavaa. Toisaalta se toimi hyvin yhdessä ketterän menetelmän kanssa, sillä oli helppo palata myös virheisiinsä, kun oli ymmärtänyt kirjoittaa oleellisia asioita ylös.

Päiväkirjamuotoisen työn heikkoutena oli oman työn ja oppimisen viikkotasoinen arviointi, joka jäi heikoksi päiväkirjassa monilta osin. Jonkinlaisesta vertaistuesta tai muusta keskustelukumppanista olisi ollut hyötyä siinä, että omat valinnat olisivat tulleet kunnolla haastetuiksi. Opinnäytetyön tilaaja kuitenkin osallistui myös keskusteluun tarvittaessa varsinkin siitä, mihin suuntaan testejä viedään, vaikka ratkaisut jäivät työn tekijälle.

Projektinhallinnan näkökulmasta Kanban-taulu oli menetelmänä hyvä valinta, koska eri osa-alueita oli mahdollista edistää samaan aikaan. Ainoat ongelmat olivat käyttäjässä: Kanban-menetelmän luonteva ja palkitseva käyttö sekä projektinjohtamisen prosessi eivät olleet ajan tasalla. Jotta ketterästä projektinvetämisestä saa kaiken irti, täytyykin varata riittävästi aikaa aluksi metatyölle, jossa Kanban-taulua suunnitellaan, ja sen jälkeen osaksi projektinjohtamisen prosessia tulisi ottaa säännöllinen taulun päivitys. Kokonaisuuden hallinta tuntui enemmänkin olevan ajatuksen tasolla, kuin konkreettisesti paperilla. Lisäksi olisi pitänyt viikoittain käyttää aikaa puhtaasti Kanban-taulun päivittämiseen.

Toisaalta myös aikaisempaan palaaminen, joustavuus ja kokeileva suhtautuminen ovat ketterien menetelmien ominaisuuksia, jotka toteutuivat projektissa, eikä Kanban toimintatapana sitonut liikaa toimijaa siinä suhteessa, kun tehtävät olivat merkitty varsin ylätasoisesti.

Kuvassa 16 on visualisoitu työn etenemistä oppimisen näkökulmasta. Siihen on koottu taulukkomuotoon viikkotasolla eniten käsittelyssä olleet teemat. Kuvan tarkoitus on havainnollistaa koko oppimisprosessia mutta samalla myös koota yhteen projektin etenemistä.

Kuva 16. Opitut teemat viikoittain

Viikko	Viikon opitut teemat
Viikko 1	Merkkijonon käsittely, Excelin lukeminen Robot Frameworkilla, muuttujien nimeäminen, merkkijonojen vertailu
Viikko 2	Tietojärjestelmän, -hallinnon ja -valvonnan käsitteet. Saatavuustavoitteet ja -vaatimukset.
Viikko 3	Ping testin saatavuussuunnitelma, muut mitattavat kohteet.
Viikko 4	Valvontasuunnitelman luonnostelu, Valvontaprosessin luonnostelu
Viikko 5	Kanban-menetelmä, projektin eteneminen. SNMP valvontatyökalu, tietokanta, RFW DateTime kirjasto
Viikko 6	Tietokannan muokkaus, SNMP:n soveltaminen Robot Frameworkissa, yhteyden muodostaminen palvelimeen, mitattavan arvon laskenta kahdesta SNMP:n hakemasta tuloksesta
Viikko 7	Palvelimen OID lukujen konfigurointi, linux-komentokehotteet, projektin etenemisen ja hallinnan arviointi
Viikko 8	CPU:n mittaus ja JSON rakenteen suunnittelu, ensimmäisen viikon ping-testin koodin ja muuttujien korjaus, tulosten rakentaminen sanakirjaksi Robot Frameworkissa.
Viikko 9	Tulosten tallentaminen JSON tiedostoksi, Evaluate toiminnon käyttö, joka estää Robot Framework-koodin kaatumisen tilanteessa, jossa SNMP ei palauta arvoa. Muu viimeistely.

Opinnäytetyön tilaaja on erityisen tyytyväinen SNMP-työkalun soveltamiseen sekä siihen, että robotti on pienillä muokkauksilla helposti käytettävissä myös laajemmin tietojärjestelmien hallintakäytäntöihin. Työn aikana on syntynyt täsmentyneempi käsitys koko organisaation valvonnan ja tietohallinnon tarpeista.

6 Yhteenveto

Tutkimuskysymyksiin vastaaminen onnistui kohtalaisesti, sillä valvonnan maailma on todella laaja. Erilaisia valvontatoimenpiteitä olisi löytynyt vielä kattavamminkin kuin mitä tämä opinnäytetyö käsitteli. Selkeää ”tämä on oikeaa valvontaa ” -tyyppistä listaa ei kuitenkaan ole olemassa, vaan valvonta määräytyy aina organisaation strategiasta ja tarpeista, jotka ovat johdettaviksi valvontasuunnitelmaksi. Tutkimuskysymysten vastaukset on esitelty johtopäätöksissä.

Tätä kehitystyötä tehdessä opin muun muassa Robot Frameworkin koodaamista, syntaksin lukemista sekä tiedonhakua ja ongelmanratkaisua. Olin varsin tyytyväinen siihen, että tein itse myös valvontasuunnitelman, koska se tarjosi minulle paremman ymmärryksen valvonnan kokonaisuudesta sekä tietohallinnosta, joka kuitenkin oleellisesti liittyy valvonnan työtehtäviin. Prosessikuvauksen tuottaminen ei ollut minulle enää niinkään uutta oppimista, vaan jo osatun asian soveltamista, mutta äärimmäisen tärkeä osa työtä siitä huolimatta. Opin myös, että suunnitteluun pitäisi käyttää enemmän aikaa ja jopa kokeilla edetä siinä jopa vähän ”takaperoisesti” valmiista työstä aloitukseen.

Tulevaisuudessa tärkeitä kysymyksiä jatkossa on se, minne tämä valvontarobotti sijoitetaan ja kuinka usein se tekee nämä valvontatestit. Luultavasti robotin ajastamisessa tullaan käyttämään cronjob-toimintoa. Lisäksi testitulosten tulkintaa varten tarvitaan dokumentaatiota siitä, mitä eri testit tarkoittavat. Tulevaisuudessa olisi myös mahdollista konfiguroida itse SNMP:lle sopivia OID:ja, riippuen valvontatarpeista.

Jatkosuunnitelmana on vielä luoda testitapaus IT-infrastruktuun ominaisuuksista, jota tässä kehitystyössä ei ehditty tekemään. Myöskään Webhook-asetuksia niin, että raja-arvojen ylittyessä tulee hälytys, ei ole tehty vaan ne jäivät tulevaan kehittämiseen. Valvontasuunnitelman pariksi olisi hyvä luoda ylläpitosuunnitelma. Tulevaisuudessa tätä robottia tullaan soveltamaan laajemmin muun muassa myös kytkinten valvontaan, mikä tarjoaa laaja-alaisen ja strategisesti merkittävän työkalun toteuttaa valvontaa. Sen myötä myös työnteko muuttuu rooliltaan proaktiiviseksi.

Lähteet

Allen, A. (Päivitetty maaliskuussa 2023). IT-performance management. Haettu 29.7.2023 osoitteesta https://www.techtarget.com/searchitoperations/definition/IT-performance-management-information-technology-performance-management

Basic concepts of Robot Framework (n.d.). Haettu 12.8.2023 osoitteesta https://robocorp.com/docs/languages-and-frameworks/robot-framework/basics

Bertram, A. (27.5.2020). Get started with threshold monitoring. Haettu 31.7.2023 osoitteesta https://www.techtarget.com/searchitoperations/tip/Get-started-with-threshold-monitoring

Bigelow, S. (5.3.2019). How to measure IT performance, step by step. Haettu 7.8.2023 osoitteesta https://www.techtarget.com/searchitoperations/tip/Tackle-IT-performance-measurement-step-by-step

Bigelow, S. (11.6.2020). The definitive guide to enterprise IT monitoring. Haettu 31.7.2023 osoitteesta https://www.techtarget.com/searchitoperations/The-definitive-guide-to-enterprise-IT-monitoring

Borgini, J. (16.4.2020). Essential components and tools of server monitoring. Haettu 31.7.2023 osoitteesta https://www.techtarget.com/searchdatacenter/tip/Essential-components-and-tools-of-server-monitoring

Fruhlinger, J. (n.d.) Mikä on JSON? Parempi muoto tietojenvaihtoon. Haettu 23.9.2023 osoitteesta https://fi.verticalshadows.com/11-what-is-json-a-better-format-for-data-exchange

Gillingham, J. (25.1.2023). An Overview of Availability Management in ITIL. Haettu 21.6.2023 osoitteesta https://www.invensislearning.com/blog/availability-management/

Hietaniemi, J. (18.3.2020). Mikä on kanban? Haettu 7.8.2023 osoitteesta https://gofore.com/mika-on-kanban/

Infrapalvelut. (n.d.). Haettu 30.7.2023 osoitteesta https://www.itewiki.fi/opas/it-infrapalvelut/

It-infrastructure. (28.9.2021). Haettu 30.7.2023 osoitteesta https://www.atatus.com/glossary/it-infrastructure/#Types-of-IT-Infrastructure

JSON – Introduction (n.d.) Haettu 23.9.2023 osoitteesta https://www.w3schools.com/js/js_json_intro.asp

Kanjilal, J. (22.7.2022). 10 application performance metrics and how to measure them. Haettu 7.8.2023 osoitteesta https://www.techtarget.com/searchapparchitecture/tip/5-application-performance-metrics-all-dev-teams-should-track

Koskinen, I. (26.3.2023). Mikä on Kanban? Katsaus menetelmään ja sen käyttöön ketterässä projektinhallinnassa. Haettu 7.8.2023 osoitteesta https://severa.fi/blogi/mika-on-kanban-katsaus-menetelmaan-ja-sen-kayttoon-ketterassa-projektinhallinnassa/

Lindroos, E. (11.10.2021). Prosessijohtamisen hyödyt ja vinkit alkuun pääsemiseksi. Haettu 7.8.2023 osoitteesta https://www.arter.fi/prosessijohtamisen-hyodyt-ja-vinkit-alkuun-paasemiseksi/

Marinelli, C. (17.7.2023). What is Infrastructure monitoring and why is it mission-critical in the new normal? Haettu 30.7.2023 osoitteesta https://www.dynatrace.com/news/blog/what-is-infrastructure-monitoring-2/

Mitä ovat ketterät menetelmät? – Scrum, Lean ja muut tutuksi. (27.1.2021). Haettu 31.7.2023 osoitteesta https://www.koulutus.fi/oppaat/projektinhallinta/ketteratmenetelmat-19939

Prosessien kehittäminen. (n.d.) Haettu 7.8.2023 osoitteesta https://www.logistiikanmaailma.fi/tuotanto/prosessien-kehittaminen/

Prosessi – miksi ja miten kehittää? (1.7.2020). Haettu 7.8.2023 osoitteesta https://mcs.fi/prosessi-miksi-ja-miten-kehittaa/

Päiväkirjamuotoinen opinnäyte Karelia-ammattikorkeakoulussa. (15.4.2021). Haettu 1.6.2023 1.6.2023 osoitteesta https://libguides.karelia.fi/ld.php?content_id=34719751

Tietohallinto. (n.d.-a). Haettu 6.6.2023 osoitteesta (https://www.itewiki.fi/opas/tietohallinto/

Robot Framework (n.d.-a). Haettu 6.6.2023 osoitteesta https://robotframework.org/

Robot Framework (n.d.-b.). Käyttöohjeet. Haettu 6.6.2023 osoitteesta https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html

Robot Framework: Past, Present and Future. (1.6.2021). Haettu 6.6.2023 osoitteesta https://www.eficode.com/blog/en/blog/robot-framework

Rock, S., Smith, B., Fowler, J., Bourgeois, D. (2022). Information Systems for Business and Beyond. Fanshawe College Pressbooks. https://ecampusontario.pressbooks.pub/informationsystemscdn/chapter/1-6-people-information-systems/

Saarinen, T. (12.4.2021). Prosessien kuvaamisen ensiaskeleet – Mitä tehdä ennen kuin aloitat kuvaamisen? Haettu 7.8.2023 osoitteesta https://www.arter.fi/prosessien-kuvaamisen-ensiaskeleet-mita-tehda-ennen-kuin-aloitat-kuvaamisen/

Slattery, T. (2020). Telemetry vs. SNMP: Is one better for network management? Haettu 14.8.2023 osoitteesta https://www.techtarget.com/searchnetworking/answer/Telemetry-vs-SNMP-Is-one-better-for-network-management

Why is server and IT system monitoring necessary? (25.6.2020). Haettu 11.6.2023 osoitteesta https://www.stackscale.com/blog/server-system-monitoring/

What Are Information Systems? – Defined and explained in 2023. (n.d.). Haettu 11.6.2023 osoitteesta https://plextrac.com/what-is-an-information-system-defined-and-outlined/

Scarpati, J. (2021). Simple Network Management Protocol (SNMP). Haettu 14.8.2023 osoitteesta https://www.techtarget.com/searchnetworking/definition/SNMP

Stedman, C. McLaughlin, E. (2022). Data collection. Haettu 29.7.2023 osoitteesta https://www.techtarget.com/searchcio/definition/data-collection

Huovinen, J. (31.3.2011). Tunnista tietohallinnon neljä toimintakategoriaa. *Business Technology Standard.* Haettu 22.7.2023 osoitteesta https://btmalli.fi/article/tunnista-tietohallinnon-nelja-toimintakategoriaa/

What is IT Monitoring? (31.1.2023). Haettu 11.6.2023 osoitteesta https://www.splunk.com/en_us/data-insider/what-is-it-monitoring.html

Why Do my Computer Systems need monitoring. (n.d.). Haettu 10.6.2023 osoitteesta https://www.swicktech.com/SWICKtech/Resources/Blog/Why-do-my-computer-systems-need-monitoring.htm

Wrobel, M. (5.7.2023). The Basics of the IT Infrastructure: Definition, Component, And Types. Haettu 30.7.2023 osoitteesta https://blog.invgate.com/it-infrastructure

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojekti:

Kehitysprojektin aikana pidetään päiväkirjaa (aineisto), johon kerätään teknistä tietoa projektista. Tämä tieto analysoidaan ja siitä koostetaan synteesejä opinnäytetyötä varten. Päiväkirja luodaan työnantajan Office365 -ympäristössä, tarkemmin Microsoft Plannerissa. Opinnäytetyön tulokset tulevat jäämään työnantajan käyttöön, joten ne tallennetaan myös työpaikan Sharepoint-pilviympäristöön. Työstä syntyvä koodi jaetaan myös organisaation GitHubissa.

Opinnäytetyössä ei käsitellä henkilötietoja.

Liite 2: Valvontasuunnitelma

Valvontasuunnitelma							
Valvontatyyppi	**Mitä sisältää**	**Miksi tärkeää**	**Mitä valvotaan**	**Miten valvotaan**	**Raportointi**	**Kohdekohtainen seurantatarve**	**Kohdekohtaiset tavoitteet/vaatimukset**
Saatavuuden valvonta	palvelun saatavuus ja komponenttien hallinta.	ennakointi, palveluhäiriöiden keston, ilmenemistiheyden ja vaikutusten minimointi. Asiakastyytyväisyys.	onko laite verkossa, onko verkkoyhteyden laatu hyvä?	ping. Tämä on yhteensopiva erittäin monien valvontakohteiden kanssa.	JSON-raportti, webhook esim. Slackiin raja-arvojen ylittyessä		
Suorituskyvyn valvonta	palvelun suorituskyky	ennakointi, ali- ja ylisuoriutuvien resurssien tunnistaminen, kuormitus	cpu,muisti (ram ja swap)	SNMP OID:t, tässä on mahdollisesti käyttöjärjestelmäkohtaisia eroja. Mahdollistaa räätälöidyt valvontatoimet	JSON-raportti, webhook esim. Slackiin raja-arvojen ylittyessä		
IT-infra	fyysisen ympäristön kuormitusriski	ennakointi: onko laite kunnossa, hyvin sijoitettu?	lämpötila	SNMP OID:t, tässä on mahdollisesti käyttöjärjestelmäkohtaisia eroja. Mahdollistaa räätälöidyt valvontatoimet	JSON-raportti, webhook esim. Slackiin raja-arvojen ylittyessä		

Liite 3: Prosessikuvaus