Iiro Iivanainen

Opasvideoiden generointi testiautomaation avulla

Tietotekniikan pro gradu -tutkielma

1. maaliskuuta 2023

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Iiro Iivanainen

Yhteystiedot: iiro.h.iivanainen@student.jyu.fi

Ohjaajat: Tommi Mikkonen ja Sauli Ruuska

Työn nimi: Opasvideoiden generointi testiautomaation avulla

Title in English: Tutorial video generation using test automation

Työ: Pro gradu -tutkielma

Opintosuunta: Ohjelmistotekniikka

Sivumäärä: 45+5

Tiivistelmä:

Laadukkaan opastuksen tarjoaminen laajalle käyttäjäkirjolle on keskeinen haaste IT-tuotteen käyttöönotossa. Konsultoinnin ja koulutuksen ohella käyttäjä voi hyödyntää ohjelmadokumentaatiota, johon lukeutuu myös erilaiset multimediaoppaat. Sekä aloittelijat että kokeneemmat käyttäjät voivat hyödyntää ohjelman toimintaa havainnoivia opasvideoita niiden konkreettisen esitysmuodon ansiosta. Ohjelman kehityksen myötä sen toiminnallisuus ja ulkonäkö muuttuvat versiosta toiseen. Pieni muutos käyttöliittymässä voi pahimmillaan vaatia tuotteeseen liittyvien videoiden täyden uudelleennauhoituksen. Tämä tutkimus esittelee suunnittelutieteen menetelmällä kehitetyn työkalun, joka luo automatisoidusti käyttövalmiita opasvideoita Robot Framework -testeistä. Nauhoitusta varten testisarjoja on täydennetty työkalun hyödyntämillä avainsanoilla, joiden avulla opasvideoon voidaan lisätä erikoistehosteita kuten tekstitystä ja käyttöliittymäkomponenttien korostusta. Generoituja opasvideoita arvioitiin Mayerin multimediaoppaiden periaatteita sekä opasvideoiden hyviä käytänteitä vasten, joista ne täyttivät suurimman osan. Työkalua voidaan jatkokehittää niin että sen generoimat opasvideot täyttävät kaikki arviointikriteerit. Generoimalla opasvideoita voidaan säästää resursseja, aikaa ja kehittäjien kuormaa.

Avainsanat: Loppukäyttäjän dokumentaatio, Näyttökaapattu opasvideo, Robot Framework, Test automation, Selenium

Abstract:

Providing high-quality guidance to a wide range of users is one of the key challenges in the introduction an IT product. In addition to consulting and training, the user can utilize the program documentation, which also includes various multimedia guides. Tutorial videos are used by both beginners and power users alike thanks to the concrete demonstrations that they provide. When a software is developed its functionality and appearance can change from version to version. At worst, a minor change in the user interface may require a complete re-recording of the videos related to the product. This study presents a tool developed using design science method, which automatically generates ready-to-use screencast video tutorials from Robot Framework tests. This is accomplished by augmenting to the test suites with keywords that can add video effects such as subtitles and UI element highlighting to the final tutorial video. The generated tutorial videos were evaluated against Mayer's principles for multimedia tutorials as well as general best practices for tutorial videos. Most of the principles and best practices were met. The tool can be further developed so that the tutorial videos it generates meet all evaluation criteria. By generating tutorial videos it is possible to save resources, time and the burden on developers.

Keywords: End user documentation, Screencast tutorial video, Robot Framework, Test automation, Selenium

Kuvat

Taulukot

Sisällys

1 Johdanto

Laadukkaan opastuksen tarjoaminen laajalle käyttäjäkirjolle on keskeinen haaste IT-tuotteen käyttöönotossa (Doll ja Torkzadeh 1993). Käyttöohjeen roolina on esitellä syy tuotteen olemassaololle, sen toimintaperiaate ja miten sitä käytetään (Doll ja Torkzadeh 1987). Videomuotoisten oppaiden on havaittu parantavan oppimistuloksia ja lisäävän mielenkiintoa opittavaan aiheeseen (Kharishma 2020). Opasvideoita käyttävät sekä aloittelijat että kokeneemmat käyttäjät niiden tarjoaman konkreettisen esitysmuodon ansiosta (Li ym. 2020).

Ohjelman kehityksen aikana sen toiminnallisuus ja ulkoasu muuttuvat. Tämä tuo mukanaan tarpeen päivittää myös tuotteisiin liittyvää dokumentaatiota, testejä ja muuta oheismateriaalia. Muutos käyttöliittymässä saattaa pahimmillaan vaatia tuotteeseen liittyvien videoiden täydellisen uudelleennauhoituksen. Teknisen dokumentaation automaattinen generointi eri tietolähteistä kuten lähdekoodista on informaatioteknologian alalla yleistä, kun taas loppukäyttäjän dokumentaatiota ylläpidetään usein käsin (Descher, Feilhauer ja Amann 2014).

Hyväksymistestauksessa ohjelmaa suoritetaan käyttäjän näkemällä tavalla (Miller ja Collins 2001). Testitapauksissa ilmeneviä käyttötapauksia voidaan hyödyntää myös loppukäyttäjän dokumentaation laatimisessa. Tässä tutkimuksessa esitellään suunnittelutieteen menetelmällä kehitetty työkalu, joka automaattisesti generoi Robot Framework -testeistä valmiita opasvideoita. Testitapauksia on täydennetty työkalun käyttämillä avainsanoilla, joiden avulla opasvideot nauhoitetaan ja jälkikäsitellään halutunlaisiksi. Tällainen avainsana voi lisätä tekstitystä tai visuaalisesti korostaa tiettyä käyttöliittymäkomponenttia opasvideolla.

Luvussa 2 tutustutaan erilaisiin tapoihin ohjeistaa loppukäyttäjää tuotteen käyttöönotossa keskittyen erityisesti multimediaoppaisiin. Lisäksi käydään läpi dokumentaation automaattista generointia ja hyvien opasvideoiden arviointikriteereitä. Luvussa 3 esitellään tutkimuskysymykset ja tavoitteet sekä kuvataan suunnittelutieteen periaatteet ja niiden soveltamista tutkimuksen eri vaiheissa. Luku 4 käy läpi kehitetyn työkalun vaatimukset, toteutuksen, rakenteen ja toiminnan. Luvussa 5 arvioidaan työkalua ja sillä generoituja opasvideoita, vastataan tutkimuskysymyksiin sekä pohditaan mahdollisia suuntia jatkokehitykselle ja jatkotutkimukselle. Luku 6 sisältää lyhyen yhteenvedon tutkimuksesta ja sen tuloksista.

2 Loppukäyttäjän ohjeistaminen

Tässä luvussa tutustutaan eri tapoihin ohjeistaa käyttäjää sekä niiden etuihin. Oppaiden automaattisesta generoinnista esitellään kaksi esimerkkiä. Lopuksi käydään läpi multimediaoppaille yhteiset Mayerin periaatteet sekä hyviä käytänteitä opasvideoiden laatimiseen.

2.1 Loppukäyttäjän dokumentaatio

Loppukäyttäjä tarkoittaa käytössä olevan tuotteen käyttäjää. Se voi viitata myös mahdollisiin järjestelmän ylläpitäjiin ja moderaattoreihin. Tietotaito, motivaatio sekä tuotteen käytön säännöllisyys voi vaihdella merkittävästi käyttäjien välillä. Dollin ja Torkzadehin (1993) mukaan laadukkaan tuen tarjoaminen laajalle käyttäjäkirjolle on keskeinen haaste tuotteen käyttöönotossa. Loppukäyttäjätuki voidaan jakaa kolmeen kategoriaan: asiantuntijakonsultointi, kouluttaminen ja ohjelmadokumentaatio. Säännölliset käyttäjät hyötyvät tyypillisesti ajan myötä vähemmän konsultoinnista ja koulutuksesta. Käyttäjälle suunnattua dokumentaatiota hyödynnetään joko koulutuksen aikana tai silloin kun ongelmakohtia tulee vastaan. Suurin osa oppimisesta tapahtuu ohjelman käytön aikana.

Loppukäyttäjän dokumentaation eli käyttöohjeen tavoitteena on kertoa käyttäjälle mikä ohjelman tarkoitus on, miten se toimii ja miten sitä käytetään (Doll ja Torkzadeh 1987). Dollin ja Torkzadehin (1993) mukaan laadukas käyttöohje tarjoaa kustannustehokkaasti kohdistettua neuvoa erilaisille käyttäjille. Konsultointiin ja koulutukseen verrattuna niiden kustannukset ovat pitkälti kertaluonteisia. Käyttöohjeet voivat kasvattaa sovelluksen arvoa käyttäjälle sekä lisätä tyytyväisyyttä käytettyä tuotetta kohtaan (Doll ja Torkzadeh 1987; Gemoets ja Mahmood 1990). Käyttäjä tyypillisesti tiedostaa dokumentaation laadun vasta silloin, kun se on puutteellisesti laadittu (Toms 1982).

Sommerville (2001) jakaa erilaiset käyttöohjeet viiteen kategoriaan. Toiminnallinen kuvaus antaa yleiskuvan tuotteen tarjoamista palveluista. Asentamisdokumentaatio kertoo ylläpitäjälle vaatimukset ja askeleet järjestelmän pystyttämistä varten. Sovelluksissa se on tyypillisesti integroitu osaksi asennusohjelmaa. Johdanto-opas tarjoaa selkokielisen ja havainnoivan johdatuksen sovelluksen yleiseen käyttöön ja pääominaisuuksiin. Referenssiopas puolestaan

antaa perusteellisen ja muodollisen kuvauksen järjestelmästä, eri virhetilanteista ja niistä toipumisesta. Järjestelmänvalvojan opas kuvaa järjestelmän eri liitännät sisään ja ulos. Se on tyypillinen esimerkiksi komento- ja ohjausjärjestelmissä.

2.2 Opasvideot ja multimediaoppaat

Digitaalinen video on hallitseva muoto verkossa tapahtuvasta sisällön kulutuksesta ja kokemusten jakamisesta (Li ym. 2020). Vuonna 2019 videosisältö kattoi 77% kaikesta internetliikenteestä (Woolfitt 2015). Teknologia on mahdollistanut videoiden korkean laadun, nopean saatavuuden, kevyet laitevaatimukset sekä helpon tuotannon. Video on pitkään nähty vaihtoehtoisena oppimismediana, joka voi tukea verkko-opetusta (Kharishma 2020). Erityisesti vuonna 2019 alkanut koronaviruspandemia on korostanut etäopetuksen merkitystä ja siten myös verkko-opetusmenetelmiä. Esimerkiksi kirjastot ovat mainostaneet ja esitelleet niiden tarjoamia palveluita opasvideoiden avulla (Martin ja Martin 2015).

Opasvideot mahdollistavat itsenäisen lähestymisen oppimiseen ja kannustavat oma-aloitteiseen tiedonetsintään (Martin ja Martin 2015). Opasvideoiden on havaittu parantavan oppimistuloksia sekä opiskelijoiden mielenkiintoa aiheeseen (Woolfitt 2015). Videoiden kuluttajat vaihtelevat yksilöinä teknologisten taitojen ja mukavuustason osalta (Mestre 2010); joukossa on sekä tehokäyttäjiä että täysiä aloittelijoita. Martin ja Martin (2015) painottavat, että sisällöntuottajien tulee tuntea videoiden tekoon liittyvät teknologiset sekä pedagokiset haasteet. Puutteelliset opasvideot voivat tehottomuutensa ohella vahingoittaa niitä esittävän tuotteen sekä organisaation mainetta. Kaiken elektronisen oppimisen keskipisteenä tulisi olla itse oppiminen eikä siinä käytetty teknologia (Colvin Clark 2011).

Palvelujen opasvideot ja opetuksessa käytetyt luentovideot eroavat tavoitteiltaan (Guo, Kim ja Rubin 2014). Luentovideot ovat tyypillisesti pitkiä kokonaisuuksia, joissa painotetaan ensimmäistä katselukertaa kokonaisuudessaan. Opasvideoiden katsojat eivät välttämättä katso kerralla koko videota ja niissä painottuu navigoinnin helppous hyödyllisen tiedon löytämistä varten. Digitaaliset multimediaoppaat voidaan jakaa 5 luokkaan niissä hyödynnetyn tekniikan mukaan (Martin ja Martin 2015). Näitä ovat: näyttökaappaus, slidecast-esitys, live-action-video, animaatio sekä interaktiiviset oppaat. Eri työkaluja on mahdollista soveltaa

keskenään. Tässä tutkielmassa keskitytään näyttökaapattuihin videoihin, koska ne sopivat erityisen hyvin tietokoneohjelmien käytön opettamiseen.

Näyttökaapatussa videossa nauhoitetaan digitaalisesti tietokoneen tai älylaitteen käyttöä ja se voi sisältää prosessin selostusta tekstin tai puheen muodossa. Näyttökaapattuja videoita tyypillisesti jälkikäsitellään leikkaamalla pois ylimääräisiä osat, säätämällä ajankulkua (esimerkiksi pysäyttämällä video selostuksen ajaksi) tai lisäämällä tekstiä ja grafiikkaa näytölle. Näyttökaappauksessa käytetyt työkalut voivat myös korostaa syöttölaitteiden käyttöä esimerkiksi näyttämällä syötetyt näppäinyhdistelmät tai korostamalla kursorin paikkaa värin ja kontrastin avulla. Näyttökaapattu video sopii hyvin sekä aloittelijoille että tehokäyttäjille. Aloittelijat tyypillisesti kuluttavat enemmän videoita, siinä missä kokeneemmat käyttäjät täydentävät tietämystään niiden avulla. Näyttökaapattujen videoiden etuna muihin multimediaoppaisiin verrattuna on niiden tuotannon helppous ja kustannustehokkuus. Käyttäjien mielestä näyttökaapatun demonstraation konkreettisuus voi kompensoida sen tökeröä toteutusta ja alhaisia tuotantoarvoja. (Martin ja Martin 2015)

Slidecast-esitys on nimensä mukaisesti diaesityksen ja podcastin yhdistelmä (Martin ja Martin 2015). Esitelmä koostuu peräkkäisistä paloista informaatiota. Se voi sisältää kuvaa, tekstiä, videota, linkkejä ja animaatiota. Esitysten etuna on, että niitä voi katsoa omaan tahtiin tai videomaisesti automaattisella etenemisellä. Live-action-videot tekevät ihmisläheisyytensä ansiosta opastuksesta helpommin lähestyttävää. Live-action-videot loistavat fyysisissä tehtävissä ja tarinankerronnassa. Videot voivat mukailla henkilökohtaista interaktiota. Hyvissä live-action-videoissa esittäjän persoona korostuu. Animaatio voi tehdä videoista vähemmän uhkaavia ja leikkisämpiä ja se voi myös tarjota tuotteelle sekä organisaatiolle mieleenpainuvat kasvot. Tuttavallista ja epävirallista puhetyyliä käyttävää esittäjää myös kuunnellaan tarkemmin (Mayer 2008).

2.3 Interaktiiviset oppaat ja aktiivinen oppiminen

Ohjelman omaksuminen interaktiivisesti on tehokkaampaa kuin animoidun demonstraation seuraaminen (Martin ja Martin 2015). Interaktiivisuus ei viittaa median kaappaamiseen tai jälkikäsittelyyn, vaan mahdollisuuteen muuttaa sisältöä käyttöliittymän kautta. Videon ta-

pauksessa se voi tarkoittaa yksinkertaisesti mahdollisuutta keskeyttää tai kelata videota. Videon navigointia voidaan helpottaa YouTube-palvelun tyylisillä aikaleimoilla tai liittämällä ruudulla näkyvää oheismateriaalia kuten hyperlinkkejä videon eri kohtiin. Asiat jotka kannustavat kognitiivista toimintaa aktiivisen oppimisen muodossa voivat tehostaa oppimisprosessia videon katselun aikana (Brame 2016). Esimerkiksi opetusmaailmassa aktiivisen oppimisen hyödyt on saavutettu esittämällä katsojalle kysymyksiä ennen videon katselua. Interaktiivisten elementtien sisällyttäminen videoon voi tukea eri oppimistyylejä (Martin ja Martin 2015). Eri henkilöt voivat oppia parhaiten seuraamalla, kuuntelemalla tai käytännön toiminnan kautta (Mestre 2010).

Ohjelman käyttöohjeet on mahdollista integroida osaksi itse ohjelmaa niin että ohjelma ohjaa käyttäjää erilaisten tavoitteiden saavuttamisessa. Opasjärjestelmät tekevät käyttäjälle selväksi eri toimintojen tarkoituksen sekä niiden väliset suhteet. Esimerkiksi PyCharm-ohjelmointiympäristö ehdottaa ensimmäisen käynnistyskerran aikana läpikäyntiä, jossa käyttöliittymän perusosat ja niiden toiminnot selitetään käyttäjälle. Integroitujen opasjärjestelmien haasteena ovat niiden vaatimat lisäresurssit ja sovelluskehittäjien lisääntynyt kuorma (Iwata, Shirogane ja Fukazawa 2006).

2.4 Oppaiden automaattinen generointi

Kehittäjille suunnattujen dokumenttien generointiin tarkoitetut työkalut ja tekniikat ovat yleisiä informaatioteknologian alalla, mutta vastaavia työkaluja ei juurikaan ole loppukäyttäjän dokumentaatiota varten (Descher, Feilhauer ja Amann 2014). Tämä johtuu lähdekoodin ja sen semantiikan välisestä aukosta. Kehittäjien dokumentaatio käsittelee tyypillisesti yksittäisten komponenttien spesifikaatioita siinä missä käyttäjän dokumentaatio on kiinnostunut ohjelmasta, sen käyttöliittymästä ja toiminnoista. On mahdollista, että yritysten haluttomuus tuottaa loppukäyttäjädokumentaatiota täysin automatisoidusti johtuu myös siitä, että käyttäjille suunnattu ohjemateriaali edustaa sekä tuotetta että yritystä. Generointitekniikoita on kuitenkin hyödynnetty aiemmin interaktiivisissa- ja multimediaoppaissa (Iwata, Shirogane ja Fukazawa 2006; Chi ym. 2012, esimerkkeinä).

Iwata, Shirogane ja Fukazawa (2006) esittivät automaattisesti generoidun ohjelmaan in-

tegroidun opasjärjestelmän, joka demonstroi käyttäjälle ohjelman toimintaa askel askeleelta. Opasjärjestelmä avaa kohdesovelluksen rinnalle uuden ikkunan joka listaa ja käy läpi tietyn käyttötapauksen suorittamiseen tarvitut askeleet. Opas visualisoi askelten aikana hiiren liikettä kohdesovelluksessa ja syöttää kenttiin sopivia esimerkkiarvoja. Opasjärjestelmän generoinnissa hyödynnetään tietolähteinä testitapauksia sekä XML-muotoisia käyttötapaus- ja sekvenssikaavioita. Sekvenssikaavioihin lisätään ylimääräistä tietoa käyttöliittymän elementeistä sekä kuvauksia niiden operaatioista. Toimintojen nimet poimitaan käyttötapauskaavioista, ja esimerkkisyötteet testitapauksista. Integroitu opasjärjestelmä on lisätty osaksi lähdekoodia hyödyntäen aspektilähtöisen ohjelmoinnin tekniikoita. (Iwata, Shirogane ja Fukazawa 2006)

Chi ym. (2012) esittivät puoliautomaattisen työkalun, jonka avulla ihmisen nauhoittamista käyttötapauksista pystytään generoimaan multimediaoppaita, joissa yhdistetään askeleita kuvaavaa tekstiä ja segmentoituja videoita. Kohdesovelluksen käyttöä esittävällä videolla korostetaan hiiren liikettä, raahaamista ja klikkauksia. Videon ohella ohjelma nauhoittaa macOS:n Event Taps -ohjelman avulla järjestelmän syötettä sekä tarkkailee suoritettuja sovelluskomentoja ja käyttöliittymäkomponentteja. Lokin perusteella videot jaetaan askelten mukaisiksi osiksi ja näille askeleille annetaan kuvaukset. Komentolokiin kirjataan aikaleimat, jotta informaatio voidaan synkronoida videon kanssa. Työkalun generoima lopullinen opas esitetään verkkosivun muodossa.

2.5 Opasvideoiden arviointikriteerit

Opasvideoiden tehokkuutta voidaan arvioida kriteerein jotka perustuvat multimediaoppaille yhteisiin Mayerin (2008) periaatteisiin. Lisäksi opasvideoiden laatimisessa voidaan hyödyntää kirjallisuudesta löytyneitä hyväksi havaittuja käytänteitä.

2.5.1 Kognitiivinen kuormitus ja Mayerin periaatteet

Opasvideoita suunnitellessa tulee katsojan sitoutumisen lisäksi huomioida opetusmediaan sekä opetettavan aiheen käsittelyyn liittyvä kognitiivinen kuormitus (Brame 2016). Sweller (1988) selittää, että pieni osa henkilön havainnoista valitaan lyhytaikaisesta aistinvaraises-

ta muistista rajoitettuun työmuistiin, ennen kuin voidaan omaksua uutta tietoa pitkäaikaismuistiin. Kognitiivista ylikuormitusta on, kun yksilöön kohdistuu tietyssä ajassa liian paljon informaatiota, jota tämä ei kykene käsittelemään yhtä aikaan (Colvin Clark 2011).

Kognitiivisen kuormitusteorian mukaan oppimisessa on kolme kuormitustekijää. Luontainen kuorma (*intrinsic load*) liittyy suoraan opittavan asian monimutkaisuuteen ja laajuuteen. Olennainen kuorma (*germane load*) liittyy asian sisäistämiseen ja sovittamiseen aikaisempaan tietoon ja sisäisiin malleihin eli skeemoihin. Lisäkuormitusta (*extraneous load*) aiheuttavat huonosti suunniteltu opetus ja ympäristön häiriötekijät. Erityisesti lisäkuormitusta pyritään minimoimaan oppaita laadittaessa. (Brame 2016)

Kognitiivisen kuormitusteorian mukaan Työmuistilla on kaksi kanavaa, joiden kautta tietoa kerätään ja käsitellään. Näitä ovat audioverbaalinen ja visuaalinen kanava. Molempien kanavien kapasiteetti on rajallinen, mutta niitä samanaikaisesti hyödyntämällä voidaan maksimoida työmuistin kapasiteetti. Liiallinen informaatio voi tukkia kanavan, joten oppaita laatiessa eri kanavien kuormitus tulee ottaa huomioon ja hyödyntää niitä toisiaan täydentävästi. (Mayer 2002)

Mayer (2008) määrittelee kymmenen periaatetta, jotka pätevät yleisesti multimediaoppaisiin. Niillä pyritään vähentämään kognitiivista kuormitusta, lisäämään käyttäjän sitoutumista aiheeseen sekä kannustamaan aktiivista oppimista. Tiedonkäsittelyn vaativuutta vähentävät periaatteet ovat:

- **Yhtenäisyys**, sisältö joka ei liity opetettavaan aiheeseen tai ei tue oppimista tulisi jättää pois (Mayer 2008). Oppiaiheen ulkopuoliseen sisältöön voi kuulua musiikki, yksityiskohtaiset taustat ja animaatiot (Brame 2016).
- **Signalointi**, nosta esiin tärkeimmät ideat ja käsitteet. Tiettyä aluetta tai tekstiä voidaan korostaa värien tai kontrastin avulla (Mayer 2008). Guo, Kim ja Rubin (2014) suosittelee hyödyntämään liikkeessä visuaalista jatkuvuutta.
- **Toisteen välttäminen**, kuvan tulisi komplementoida eikä toistaa samaa sisältöä. Selostuksen aikana ruudulle ei tulisi lisätä erillistä tekstiä, koska se kuormittaa audioverbaalista kanavaa. Animaatiota on parempi esittää selostuksen aikana, sillä se kuormittaa vain visuaalista kanavaa. (Mayer 2008)

- **Alueellinen vierekkäisyys**, silmien liike tulee minimoida sijoittamalla tekstit ja kuvat lähelle asianmukaista paikkaa (Mayer 2008).
- **Ajallinen läheisyys**, sisältöä vastaava selostus ja grafiikat tulee esittää samanaikaisesti, jotta katsoja voi helpommin yhdistää ne mielessään (Mayer 2008).

Oppimateriaalin haastavuuden hallintaan liittyvät periaatteet ovat:

- **Osiin jako**, asiakokonaisuus tulisi jatkuvan esityksen sijaan jakaa helpommin sulatettaviin peräkkäisiin paloihin (Mayer 2008). Lyhyet videot vähentävät mahdollista mielen harhailua (Brame 2016). Lisäksi ne kannustavat katsojaa etenemään itselleen sopivassa tahdissa (Johnson ja Mayer 2009).
- **Aiheeseen valmentaminen**, ihmiset oppivat paremmin kun he tuntevat aiheeseen liittyvien osien nimet, sijainnit ja ominaisuudet valmiiksi (Mayer 2008).
- **Modaliteetti** valitse audioverbaalisesta ja visuaalisesta kanavasta sopivampi informaation esitystä varten. Esitä selostus tekstin sijaan mielummin puhuttuna, jos ruudulla on samaan aikaan toimintaa. (Mayer 2008)

Opitun asian omaksumista edistävät periaatteet ovat:

- **Multimedia**, sekä audioverbaalista että visuaalista kanavaa tulee hyödyntää opettamisessa. Ihmiset oppivat paremmin kuvien ja sanojen yhdistelmistä. (Mayer 2008)
- **Personointi**, suosi epävirallista puhetyyliä. Keskustelukumppaniksi omaksutun henkilön puhetta kuunnellaan tarkemmin. Auditiivinen signalointi mukailee ihmisvuorovaikutusta ja lisää katsojan kiinnostusta sekä edistää sitoutumista sisältöön ja opitun omaksumista (Yu ym. 2009). Guo, Kim ja Rubin (2014) suosittelevat ripeää ja innostunutta puhetyyliä.

2.5.2 Hyvät käytänteet

Sisällöltään opasvideoiden tulisi olla ajantasaisia ja niistä vastaavan henkilön tulisi seurata muutoksia kohdesovelluksen eri versioiden välillä. Pienet muutokset käyttöliittymässä eivät välttämättä aina vaadi videoiden täyttä uudelleennauhoittamista. Yleisiä käsitteitä ja menettelytapoja esittävät opasvideot pysyvät paremmin ajan tasalla kuin työkalun yksityiskohtiin

keskittyvät. (Martin ja Martin 2015)

Videon osiin jako voi motivoida oppijoita etenemään heille sopivalla tahdilla (Johnson ja Mayer 2009). Guo, Kim ja Rubin (2014) havaitsivat, että opetustarkoituksessa alle 6 minuutin videoissa mediaanikatseluaika oli lähes 100%. 9–12 minuutin pituisten videoiden sisällöstä katsottiin vastaavasti noin puolet ja 12-40 minuutin videoista vain viidesosa. Martin ja Martin (2015) huomioivat, että opiskelijat kokivat yli 3 minuutin pituisen aiheeseen johdattelevan videon liian pitkänä. He suosittelevat sisällön esitysjärjestykseksi käänteisen pyramidin mallia. Siinä tärkein tieto annetaan alussa, minkä jälkeen annetaan täydentävää ja laajempaan kontekstiin yhdistävää tietoa.

Saavutettavuus tulisi ottaa huomioon jo videon esituotannossa (Martin ja Martin 2015). Huomiota vaativat asiat ovat mm. katsojan kielitaito, heikkonäköisyys, värisokeus ja kuulovamma. Mediasoittimessa tulisi löytyä vaihtoehto asettaa tekstitykset päälle ja pois. Esimerkiksi YouTube-soittimen kuvatekstit ja koneälyn käännökset parantavat videon saavutettavuutta. Oudin 2011 mukaan jokainen käyttäjältä vaadittu operaatio tulisi selittää tekstin tai puheen avulla. Lisäksi oppaiden ei tulisi vaatia motorisesti hienovaraisten syöttölaitteiden käyttöä, kuten hiiren raahaamista.

Martin ja Martin (2015) muistuttavat, että videon esitysmuodon tulee olla yhteensopiva yleisimmin käytettyjen laitteiden kanssa. Videota voidaan katsoa pöytäkoneella, kannettavalla tietokoneella, tabletilla tai puhelimella ja siinä näkyvän tekstin tulisi olla luettavissa. *16 : 9* Kuvasuhde ja *1920 x 1080* tai *1280 x720* näyttötarkkuudet ovat yleisiä sekä tietokoneilla että älylaitteilla. Myös videon isäntäpalvelimen ja mediasoittimen valinta tulisi olla tarkoituksenmukaista. Videon katsomisen ei tulisi vaatia käyttäjää lataamaan ylimääräisiä sovelluksia laitteellensa.

3 Suunnittelutieteellinen tutkimus

Tässä luvussa perustellaan tutkimuskysymysten merkityksellisyyttä ja valittua tutkimusmenetelmää. Lisäksi kuvataan suunnittelutieteen periaatteet ja niiden soveltaminen tutkimuksen puitteissa.

3.1 Tutkimuskysymykset ja tavoitteet

Tutkimuksen aiheena on selvittää voiko kehitteillä olevan ohjelman opasvideoiden ajan tasalla pitämistä helpottaa generoimalla ne käyttöliittymän testitapauksista. Tutkimuksen pääongelmana on selvittää kuinka käytännöllistä on automatisoida opasvideoiden tuottaminen. Tutkimusongelma jaetaan seuraaviin tutkimuskysymyksiin:

1. Miten opasvideoiden generointi vaikuttaa testitapausten laatimiseen?
2. Miten hyvin järjestelmällä generoidut videot täyttävät opasvideon kriteerit?
3. Mitä videoiden generointi saavuttaa verrattuna käsin tekemiseen?

Ensimmäisen kysymyksen tarkoitus on arvioida työkalun käytön vaikutuksia testitapausten laatimiseen. Tarkentavat kysymykset ovat: tuleeko nauhoituksessa olevien testitapausten olla erillisiä muista testitapauksista, kuinka työlästä on laatia testitapauksia opasvideoita varten ja kuinka luettavia testitapaukset ovat. Toinen kysymys arvioi teoriapohjaa hyödyntäen opasvideoiden laadukkuutta. Tutkimuksen aikana luotiin hyviä periaatteita noudattaen myös käsintehty opasvideo, johon generoituja videoita verrataan. Kaksi ensimmäistä kysymystä vastaavat osittain kolmanteen kysymykseen ja antavat mahdollisia kehitysideoita tutkimuksessa luodulle työkalulle.

3.2 Suunnittelutiede

Marchin ja Smithin (1995) mukaan informaatioteknologian ilmiöihin kohdistuva tieteellinen mielenkiinto jakautuu kuvailevan (*descriptive*) ja ohjailevan (*prescriptive*) tutkimuksen paradigmoihin. Kuvaileva tutkimus vastaa luonnontieteellistä lähestymistapaa, jossa pyritään löytämään ja selittämään ilmiöitä lakien, mallien ja teorioiden kautta. Suunnittelutiede

puolestaan vastaa ohjailevaa tutkimusta. Siinä pyritään teknologiaorientoituneesti luomaan innovatiivisia artefakteja, jotka vastaavat tärkeisiin ja liiketoiminnan kannalta relevantteihin tarpeisiin (Hevner ym. 2004). Menetelmät komplementoivat toistensa kehitystä (March ja Smith 1995). Luonnontieteellinen lähestymistapa tarjoaa teorioita ja malleja kysymyksiin *miten* ja *miksi* tietty artefakti toimii. Suunnittelutiede käyttää hyväkseen luonnontieteen tuloksia artefaktien suunnittelussa ja vastaavasti tarjoaa konkreettisia testejä luonnontieteen tarjoamien teorioiden pitävyydestä.

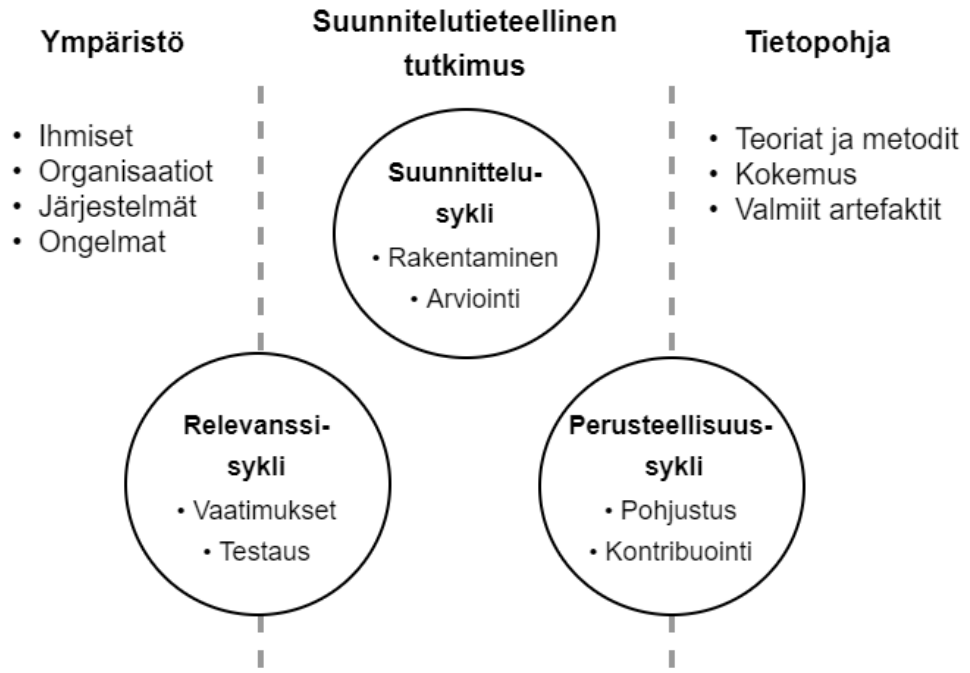
Artefaktit tyypillisesti liittyvät teknologiseen, organisatoriseen tai ihmislähtöiseen toimintaan (Hevner ym. 2004). Ne voivat olla rakennelmia, malleja, menetelmiä, teorioita tai instantaatioita. Rakennelma tarjoaa kielen, jolla ilmiötä voidaan kuvailla. Mallit ovat rakennelmista kasattuja kokonaisuuksia. Menetelmät ovat tapoja toteuttaa tavoitteita. Instantoituneet artefaktit ovat fyysisiä ilmentymiä kuten tuotteita ja prototyyppejä, joiden tarkoituksena on suorittaa tiettyjä tehtäviä. Spesifit artefaktit, kuten ohjelmien tai prosessien implementoinnit, antavat rajoitettua ja usein epäkypsää informaatiota verrattuna malleihin ja metodeihin (Gregor ja Hevner 2013). Hyvin kehittyneet teoriat ovat verrattain kattavimpia ja informatiivisimpia artefakteja.

Suunnittelutieteellisen tutkimuksen tuloksena saavutetaan kumuloitunutta tietoa ja ymmärrystä ongelma-alueesta (Gregor ja Hevner 2013). Yleensä tämä tarkoittaa uusien ratkaisujen kehittämistä tunnetuille ongelmille tai tunnettujen ratkaisujen hyödyntämistä uusissa ongelmissa. Kontribuutiot voivat löytää uusien ratkaisujen lisäksi uusia ongelmia eli tutkimusmahdollisuuksia. Edistyksen tiedetään tapahtuneen silloin, kun suunniteltu artefakti korvaa aikaisemman tarkoitukseen suunnitellun teknologian (March ja Smith 1995).

3.2.1 Suunnittelutieteen vaiheet

Hevnerin (2007) mukaan suunnittelutieteen vaiheista tulisi erottua kuvan 1 mukainen kolmen syklin rakenne. Näitä ovat relevanssisykli (*relevance cycle*), perusteellisuussykli (*rigor cycle*) ja suunnittelusykli (*design cycle*). Perusteellisuussykli erityisesti erottaa suunnittelutieteellisen tutkimuksen tavallisesta tuotekehityksestä.

Relevanssisykli toimii siltana ongelmakontekstin ja suunnitteluprosessin välillä. Sen aika-



Kuva 1. Suunnittelutieteen monisyklinen rakenne.

na kootaan vaatimuksia ja testataan artefaktin tarkoituksenmukaisuutta sille luonnollisessa ympäristössä. Ympäristö voi koostua inhimillisistä, organisatorisista tai teknisistä tekijöistä. Relevanssisykli tarjoaa lähtökohdan suunnittelutyölle sekä lopulliset kriteerit artefaktin hyväksymiselle. Artefaktia tulisi testata kohdeympäristössä vasta usean suunnittelusyklin jälkeen. (Hevner 2007)

Perusteellisuussykli yhdistää tieteelliset käytänteet ja suunnittelutyön (Hevner ym. 2004). Siinä etsitään ja hyödynnetään valmiina olevia teorioita, malleja ja menetelmiä sekä vastaavasti kontribuoidaan tutkimuksen aikana opittua tietotaitoa. Jos aikaisempaa tietopohjaa ei käytetä ei voida taata, että artefakti todella tarjoaa jotain innovatiivista ongelmien ratkaisussa. Hevner (2007) suosittelee, että tutkija etsii useita vaihtoehtoisia lähteitä, joiden tarjoamiin teorioihin suunnitteluratkaisuja voidaan pohjustaa. Hevner myös huomauttaa, että voi olla epärealistista ja jopa haitallista pyrkiä perustelemaan jokaista ratkaisua teoriaan nojautuen. Aiemman tiedon puute on suunnitteluprosessille tyypillistä.

Suunnittelusykli on tutkimuksen keskeisin prosessi, jossa artefaktia vuoron perään iteratiivi-

sesti rakennetaan ja arvioidaan (Hevner 2007). Rakentamisen ja arvioinnin tulisi olla Hevnerin mukaan määrältään tasapainoista. Artefaktia pyritään parantamaan luomalla vaihtoehtoisia toteutuksia, joita verrataan relevanssisyklin aikana koottuihin vaatimuksiin. Arviointi perustuu perusteellisuussyklin aikana koottujen teorioiden, mallien ja menetelmien hyödyntämiseen. Suunnitteluprosessi itsessään on pitkälti luova prosessi, jonka harjoittamiseen liittyvät teoriat ovat usein puutteellisia (March ja Smith 1995). Siinä tulee nojautua intuitioon, kokemukseen sekä virheiden kautta oppimiseen.

3.2.2 Artefaktien arviointi

Artefaktin arviointi tieteellisin standardein on suunnittelutieteessä välttämätöntä (Hevner ym. 2004). Artefaktien perusteellinen arviointi voi olla suunnittelutieteen isoimpia haasteita ympäröivien tekijöiden ja resurssien puutteen takia. Suuri osa ohjelmistoalan tuotteiden testaamisesta on todellisuudessa vain käytön demonstrointia (Zelkowitz ja Wallace 1998). Tällaiseen kokeiluun liittyy harvoin datan keräämistä tai mallien soveltamista, puhumattakaan teorioista. Siksi se ei riitä arvioimaan artefaktien tarkoituksenmukaisuutta ja tehokkuutta.

Arvioinnissa käytettyjä kriteereitä ovat validiteetti, käytettävyys, tehokkuus ja laatu (Gregor ja Hevner 2013). Arvioitavia laadullisia ominaisuuksia voivat olla esimerkiksi. toiminnallisuus, valmius, yhdenmukaisuus, tarkkuus, suorituskyky, luotettavuus, käytettävyys sekä organisaationaalinen istuvuus (Hevner ym. 2004). Hevner luokitteli arviointimenetelmät viiteen eri kategoriaan:

- **Havainnollistava**, missä artefaktia käytetään joko tapaustutkimuksen tai kenttätutkimuksen aikana.
- **Analyyttinen**, missä artefaktin arkkitehtuuria, optimointia sekä staattisia ja dynaamisia ominaisuuksia tutkitaan.
- **Kokeellinen**, missä artefaktia tutkitaan valvotussa ympäristössä tai simuloidaan keinotekoisen datan avulla.
- **Testaava**, missä artefaktia testataan joko mustalaatikko- tai valkolaatikkotestauksen avulla.
- **Kuvaileva**, missä artefaktin hyödyllisyyttä argumentoidaan aiempaan tietoperustaan

viitaten tai sen toimintaa kuvataan skenaarioiden avulla.

March ja Smith (1995) huomauttavat, että artefaktien kuvaileminen ja arviointi helpottaa niiden kategoriointia minkä ansiosta tutkimusponnisteluja ei tarvitse hukata aiheisiin, joita on jo aiemmin tutkittu riittävästi. Tässä tutkimuksessa artefaktia arvioidaan kuvailevalla menetelmällä. Kuvailevaa arviointia suositellaan erityisesti innovatiivisille artefakteille tai kun muu arviointimenetelmä ei ole tutkimuksen puitteissa mielekäs (Hevner ym. 2004). Mutta muihin arviointimenetelmiin verrattuna sen todistusarvo on kuitenkin suhteellisen rajallinen.

3.3 Suunnittelutieteen soveltaminen tässä tutkimuksessa

Hevner ym. (2004) määritteli seuraavat seitsemän ohjenuoraa suunnittelutieteen harjoittamiselle:

1. **Artefaktin suunnittelu**, suunnittelutieteen tulee tuottaa rakennelma, malli, menetelmä tai instantaatio.
2. **Ongelman relevanssi**, tavoitteena on kehittää teknologiaperustainen ratkaisu taloudellisesti tärkeälle ja ratkaisemattomalle ongelmalle.
3. **Artefaktin arviointi**, hyöty, laatu ja tarkoituksenmukaisuus tulee demonstroida täsmällisesti suoritetun arviointimenetelmän avulla.
4. **Tutkimuksen kontribuutiot**, tutkimuksen tulee tarjota selkeät ja verifioitavat kontribuutiot artefaktin, suunnitteluperusteiden tai suunnittelumenetelmien muodossa.
5. **Tutkimuksen perusteellisuus**, artefaktin rakentamisessa ja arvioinnissa tulee noudattaa tieteellisestä täsmällisyyttä.
6. **Suunnittelu etsintäprosessina**, artefaktin tulee saavuttaa tavoitteensa mahdollisimman hyvin kunnioittaen ongelma-alueen lakeja.
7. **Tutkimusviestintä**, tutkimus tulee esittää selkeästi sekä teknologia- että johtamissuuntautuneille henkilöille.

Tutkimuksessa Suunniteltu artefakti on käyttökelpoinen työkalu eli instantaatio. Ohjelmistorobotiikkaa tai testiautomaatiota ei ole tiedettävästi aiemmin hyödynnetty opasvideoiden generoinnissa, mikä tekee sen toteutuksesta ratkaisemattoman ongelman. Artefaktin tavoitteena on säästää organisaation resursseja, mikä tekee ongelmasta taloudellisesti merkittävän

ratkaisun. Artefaktilla luotuja opasvideoita verrataan aliluvussa 2.5 esiteltyjen arviointikriteereiden perusteella ja vaikutuksia testitapausten laadintaan pohditaan. Tutkimuksessa kuvataan täsmällisesti, miten kehitetty artefakti toimii ja miten se rakennettiin jäljiteltävyyden takaamiseksi.

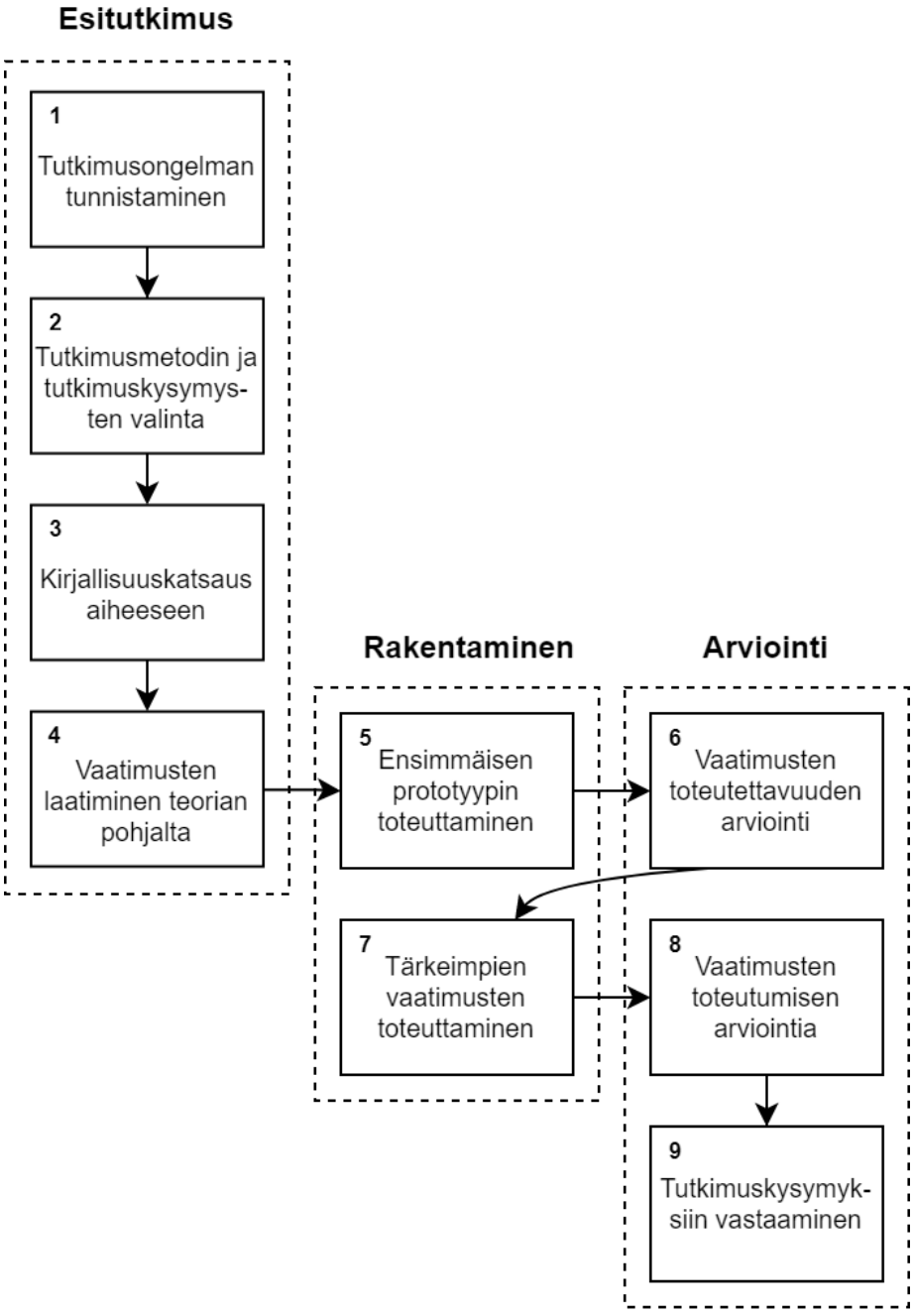
3.4 Tutkimuksen vaiheet

Tutkimuksen eri vaiheet ovat yleisesti kuvattu kuvassa 2. Esitutkimusvaihe kattoi suurimman osan tutkimuksesta ajallisesti. Se piti sisällään tutkimusongelman tunnistamisen, tutkimuskysymysten valinnan, kirjallisuuskatsauksen aiheeseen sekä arviointikriteerien ja vaatimusten laatimisen kirjallisuuskatsauksessa kootun teorian pohjalta. Artefaktin kehityksen aikana luotiin yksinkertainen prototyyppi, jonka avulla varmistettiin halutunlaisen artefaktin toteutettavuus ja siitä kehitettiin tärkeimmät vaatimukset täyttävä työkalu, jolla vastattiin tutkimuskysymyksiin. Työkalun rakentamisen ohessa tuotettiin käsintehty opasvideo, joka vastasi liiketoimintaympäristön tarpeita. Käsintehdyn opasvideon ominaisuuksia verrattiin lopullisiin generoituihin opasvideoihin puutteiden ja jatkokehitysideoiden tunnistamista varten.

3.4.1 Esitutkimus

Työn tavoitteena oli tutkia voiko kehitteillä olevan verkkosovelluksen opasvideoita pitää helpommin ajantasaisina kehityksen aikana. Ideana on yhdistää loppukäyttäjän dokumentaatio ja testaaminen hyödyntäen molemmissa esiintyviä käyttötapauksia. Tavoitteena oli tuottaa ja dokumentoida artefakti sekä saada selkeä käsitys siitä miten sen voisi jalostaa osaksi työkaluketjua uusien sovellusversioiden käyttöönotossa. Aiheen ideoinnin yhteydessä ehdotettiin sen selvittämistä voisiko dokumentaatio- ja testikattavuutta mitata yhdessä, mutta tämä päätettiin jättää tutkimusongelman ulkopuolelle.

Kirjallisuuteen tutustuttaessa ei löydetty artefaktia vastaavaa työkalua, joka generoisi täysin automatisoidusti opasvideoita testitapauksia hyödyntäen. Suunnittelutiede valittiin tutkimusmenetelmäksi, koska aihe tarjosi mahdollisuuden kehittää ja arvioida uudenlaista artefaktia. Kirjallisuuskatsauksen aikana tutustuttiin suunnittelutieteen periaatteisiin ja loppu-



Kuva 2. Tutkimuksen vaiheet järjestyksessä.

käyttäjän dokumentaation generointitekniikoihin. Suunnittelutieteeseen perehdyttäessä tunnistettiin tarve teoriapohjalle, jolla opasvideoiden laatua voitaisiin arvioida. Mayerin periaatteet ovat kirjallisuudessa eniten toistuvat arviointikriteerit multimediaoppaille. Opasvideoiden arviointikriteerien sekä kohdeympäristön tarpeiden pohjalta laadittiin vaatimukset generoiduille opasvideoille ja itse kehitettävälle työkalulle, jotka on määritelty aliluvussa 4.1. Vaatimusmäärittelyn jälkeen työkalua alettiin rakentamaan.

3.4.2 Työkalun rakentaminen

Työkalun toteutettavuutta arvioitiin tarkastelemalla kirjastoja ja testikehyksiä, jotka mahdollistaisivat opasvideoiden luomisen. Oleellisinta oli löytää työkalut käyttöliittymätestien suorittamiseen, kirjausten tekemiseen testien aikana, näyttökaappaukseen ja videotiedostojen ohjelmalliseen jälkikäsittelyyn. Sopivat työkalut löytyivät Robot Frameworkistä ("Robot Framework User Guide" 2022) ja Python-ohjelmointikielen helppokäyttöisistä kirjastoista. Myös Selenium Webdriverin ("WebDriver - Selenium" 2022) käyttöä sellaisenaan harkittiin, mutta Robot Frameworkiin päädyttiin testien luettavuuden takia.

Työkalun ensimmäisen prototyypin tarkoitus oli varmistaa, että valitut työkalut sopivat halutunlaisen lopputuloksen saavuttamiseen. Sovellusta varten laadittiin kansiorakenne, joka vastaa tavanomaisen Robot Framework -projektin kansiorakennetta (katso liite E). Testeille, resurssitiedostoille sekä tuloksille annettiin omat kansionsa. Ennen JSON-formaatin valitsemista kirjauksia varten harkittiin CSV-formaattia, mutta tämä olisi sopeutunut huonosti erityyppisiin kirjauksiin, joissa voi olla eri määrä otsikkoarvoja. JSON-formaatti sopii huonommin uuden informaation reaaliaikaiseen kirjoittamiseen, sillä JSON-tiedosto tulee lukea ennen kuin siihen voidaan liittää dataa. Tämän takia laadittiin *NarrationRecorder* Python-luokka, joka pitää tilassaan testitapauksen tietorakennetta ja liittää sen JSON-tiedostoon vasta testitapauksen lopussa. Ensimmäinen prototyyppi kykeni suorittamaan sille annetut testisarjat, tekemään aikaleimattuja kirjauksia testien suorituksen aikana sekä nauhoittamaan testitapausten videotiedostot Robot Frameworkin valmiin ScreenCapLibraryn avulla. Mikäli virheitä tapahtui pääohjelman tai Robot Frameworkin suorituksen aikana, virheilmoitus tulostettiin konsoliin.

Ensimmäisestä prototyypistä sekä Robot Frameworkin ja MoviePyn dokumentaatioon tutustumisen jälkeen varmistuttiin, että halutunlainen työkalu voidaan toteuttaa valituilla teknologioilla. MoviePyn avulla videotiedostot kyettiin yhdistämään yhdeksi videoksi ja tekstitykset lisäämään siihen vaivattomasti. Alun perin eri testitapausten mukaisia osioita oli tarkoitus esittää pelkkien tekstitiedostossa olevien aikaleimojen muodossa. Videosta kuitenkin tulisi olla itsenäisesti selvää, mitä ruudulla esitetään. Tämän vuoksi nauhoitettujen testitapausten alkuun lisättiin otsikot, jotka aikaleiman tapaan johdettiin testitapausten nimistä. Viimeinen ominaisuus mikä opasvideoihin lisättiin, oli HTML-elementin korostus. Korostusta varten harkittiin punaisen ympyrän tai nuolien käyttöä, mutta lopulta päädyttiin suorakulmioihin. Elementin rajaavaa laatikkoa (*bounding box*) on yksinkertaista pyytää verkkoselaimelta ja se esittää usein elementin rajat täsmällisesti.

4 Video-oppaiden generointi käyttöliittymätesteistä

Tässä luvussa käydään läpi työkalulle ja sillä generoiduilla videoilla laaditut vaatimukset. Sen jälkeen esitellään toteutusta varten valitut tärkeimmät työkalut ja lopuksi käydään läpi ohjelman rakennetta ja toimintaa yksinkertaisen testisarjan avulla.

4.1 Työkalun vaatimusmäärittely

Generoiduille videoille laaditut vaatimukset perustuvat haluttuihin perustoiminnallisuuksiin sekä aliluvussa 2.5 määriteltyihin Mayerin periaatteisiin ja hyviin käytänteisiin. Osa hyvien periaatteiden toteutumisesta tulee olemaan työkalun käyttäjän vastuulla kun testisarjoja täydennetään työkalun hyödyntämillä avainsanoilla. Opasvideoille asetetut vaatimukset on listattu taulukossa 1. Vaatimuksia 1.7, 1.8 ja 1.9 ei toteutettu lopullisessa työkalussa. Syötetyn tekstin ja hiiren painallusten korostamista voidaan korvata HTML-elementin korostuksella. Koneääniselostus puolestaan nähtiin laskevan opasvideoiden koettua laadukkuutta.

Vaatimus	Kuvaus	Prioriteetti
1.1	Testitapausten nauhoittaminen.	Pakollinen
1.2	Testitapauksista johdetut videot voidaan yhdistää yhdeksi videoksi.	Tärkeä
1.3	Ruudulla näkyvä selostavat tekstitykset.	Tärkeä
1.4	HTML-elementin korostaminen.	Tärkeä
1.5	Oppaan eri osiot otsikoidaan testitapauksen mukaan.	Tärkeä
1.6	Testiaskeleet suoritetaan helposti seurattavassa tahdissa.	Tärkeä
1.7	Syötetyn tekstin korostaminen.	Valinnainen
1.8	Hiiren painallusten korostaminen väreillä.	Valinnainen
1.9	Koneääni selostus.	Ei toteuteta

Taulukko 1. Opasvideoiden vaatimukset.

Työkalun toiminnallisuuteen ja käytettävyyteen liittyvät vaatimukset on listattu taulukossa

Vaatimus	Kuvaus	Prioriteetti
2.1	Työkalu tekee kaiken vaadittavan yhdellä komennolla, missä testisarjat ovat argumentteja.	Tärkeä
2.2	Kirjaukset tehdään JSON-tiedostoon, joka mahdollistaa uudenlaisten kirjaustyyppien lisäämisen.	Tärkeä
2.3	Työkalu ilmoittaa, jos jokin testitapauksen tai pääohjelman ajon aikana meni vikaan.	Tärkeä
2.4	Tuloskansioon lisätään tekstitiedosto, joka sisältää aikaleimat eri osioihin.	Tärkeä
2.5	Tuloskansioon lisätään videotiedosto ilman jälkikäsittelyä, joka sisältää nauhoitetut testitapaukset yhdessä.	Valinnainen
2.6	Tuloskansioon lisätään tekstitystiedosto joka noudattaa jotain yleistä tekstitysformaattia.	Valinnainen

Taulukko 2. Työkalun vaatimukset.

2. Vaatimusta 2.6 lukuunottamatta kaikki työkaluun liittyvät vaatimukset toteutettiin.

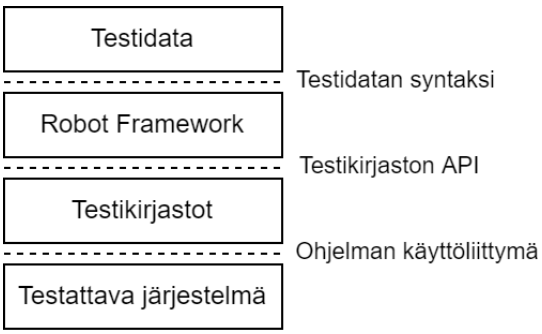
4.2 Toteutusta varten valitut työkalut

Kehitetty ohjelma toteutettiin pääosin Python-kielellä sekä Robot Frameworkin testinotaatiolla. Python oli luonnollinen valinta, sillä Robot Frameworkin käyttö vaati valmiiksi Python-ympäristön. Lisäksi Pythonille löytyi vaatimukset täyttävä videon ohjelmalliseen jälkikäsittelyyn tarkoitettu MoviePy-kirjasto. Ohjelman tuottamiseen käytettyjen ohjelmistotyökalujen tarkat versionumerot on lueteltu liitteessä A.

4.2.1 Robot Framework ja testikirjastot

Robot Framework on avoimen lähdekoodin ohjelmistokehys, jota voidaan hyödyntää hyväksymistestauksen automaatiossa sekä ohjelmistorobotiikassa eli digitaalisen liiketoiminnan automatisoinnissa. Sitä voidaan käyttää esimerkiksi prosessien, verkkosovellusten tai graafisten Java-ohjelmien suorittamisessa. Robot Framework on alustariippumaton mutta ei it-

sessään mahdollista testaamista, vaan vaatii back-endinä toimivan testauskirjaston avukseen. Kuva 3 havainnollistaa Robot Frameworkin arkkitehtuurin. Testikirjastojen valinta riippuu testattavasta järjestelmästä. (“Robot Framework User Guide” 2022)



Kuva 3. Robot Frameworkin arkkitehtuuri.

Robot Frameworkin testidatalla on oma avainsanoihin perustuva syntaksi (“Robot Framework User Guide” 2022). Taulukkomuodossa esitetyt avainsanat ovat verrattavissa aliohjelmakutsuihin, joilla voi olla mahdollisia parametreja ja paluuarvoja. Uusia avainsanoja voi laatia kokoamalla vanhoja avainsanoja yhteen ja lisäämällä kutsuja ulkoisiin kirjastoihin. Avainsanan suorittaminen vastaa yhtä testiaskelta. Mikäli sen aikana aiheutuu poikkeustilanne, se nähdään testin epäonnistumisena. Syntaksi tukee myös muuttujien käyttöä ja loogisia rakenteita, kuten silmukoita ja if-lauseita. Esimerkkejä avainsanaperustaisesta syntaktista on esitetty liitteissä B ja D. Klärck (2019) suosittelee, että testissä käytettyjen avainsanojen tulisi olla ihmisluettavia ja alemman tason teknisten yksityiskohtien tulisi olla abstrahoitu niiden taakse. Abstraktiotaso voi kuitenkin tarpeen mukaan vaihdella eri testisarjojen ja testitapausten välillä.

Robot Frameworkin toiminnallisuutta voidaan laajentaa valmiilla tai käyttäjän toteuttamilla kirjastoilla (“Robot Framework User Guide” 2022). Ensisijainen ohjelmointikieli käyttäjien toteuttamille kirjastoille on Python, jolla Robot Framework on myös toteutettu. Valmiit kirjastot voidaan asentaa Pythonin Pip-paketinhallintajärjestelmän avulla. ScreencapLibrary on valmis Robot Frameworkin kirjasto kuva- ja näyttökaappaukseen. Testiajoja suorittaessa ScreencapLibrarya käytetään testien nauhoittamiseen (“ScreenCapLibrary” 2022). Testeissä voi suorittaa valmistelevia testiaskeleita ennen näyttökaappauksen aloittamista.

Kirjastona toimiva Python-moduuli voi olla staattinen joukko aliohjelmia tai luokka, jolla on

oma tilansa (“Robot Framework User Guide” 2022). Luokkana implementoitu kirjasto voidaan alustaa olioksi esimerkiksi robot-testitiedoston sisältämän testisarjan alussa (*Test suite setup*) ja sen metodeja kutsua testitapausten aikana. Python-aliohjelmakutsut voidaan suorittaa matalan tason avainsanoissa, jotka kootaan osaksi uudelleenkäytettävää resurssitiedostoa. Resurssitiedosto vastaa syntaksiltaan testidatatiedostoa, josta puuttuu testitapaukset. Tutkimusta varten luotiin NarrationRecorder-niminen Python-luokka, jonka avulla testitapauksia suorittaessa tehdään kirjauksia JSON-tiedostoon.

4.2.2 Selenium

Selenium WebDriver on avoimen lähdekoodin verkkoselaimen automatisointityökalu, joka tarjoaa geneerisen rajapinnan eri selainten käskyttämiselle (“WebDriver - Selenium” 2022). Selaimen käyttäminen edellyttää, että selainta vastaava ajuritiedosto on asennettuna; esimerkiksi ChromeDriver Google Chromen tapauksessa. Selenium ei yksinään riitä ohjelmistotestaukseen vaan vaatii testauskehyksen. SeleniumLibrary on Robot Frameworkin syntaksille toteutettu testauskirjasto (“SeleniumLibrary” 2022). Kirjaston avulla voidaan mm. navigoida verkkosivua, kutsua JavaScript-metodeja ja testata verkkosivun sisältöön liittyvien väitelauseiden pitävyyttä.

4.2.3 MoviePy

Moviepy on avoimen lähdekoodin Python-kirjasto videotiedostojen ohjelmalliseen käsittelyyn ja prosessointiin (“MoviePy documentation” 2022). Graafisten videonmuokkausohjelmien tapaan videota käsitellään leikkeiden (*clip*), erikoistehosteiden ja muunnoksien avulla. Leikkeet voivat olla videoita, ääniä, tekstejä, kuvia tai animaatioita. Kirjaston tarjoamiin perusoperaatioihin kuuluu leikkeiden leikkaaminen, ketjutus, sommittelu ja nopeuden käsittely. MoviePy vaatii ImageMagick-ohjelmistopaketin asennuksen tekstin käsittelemiseksi minkä avulla videoon voidaan lisätä esimerkiksi otsikointia ja tekstitystä. MoviePyta voidaan käyttää yhdessä myös muiden Python-kirjastojen kanssa animointiin ja edistyneeseen kuvankäsittelyyn. MoviePyn ohella grafiikan piirtämisessä hyödynnetään Gizeh-vektorigrafiikkakirjastoa, joka perustuu Cairo-nimisen C-kirjaston Python-sidontaan (“Gizeh Github” 2018).

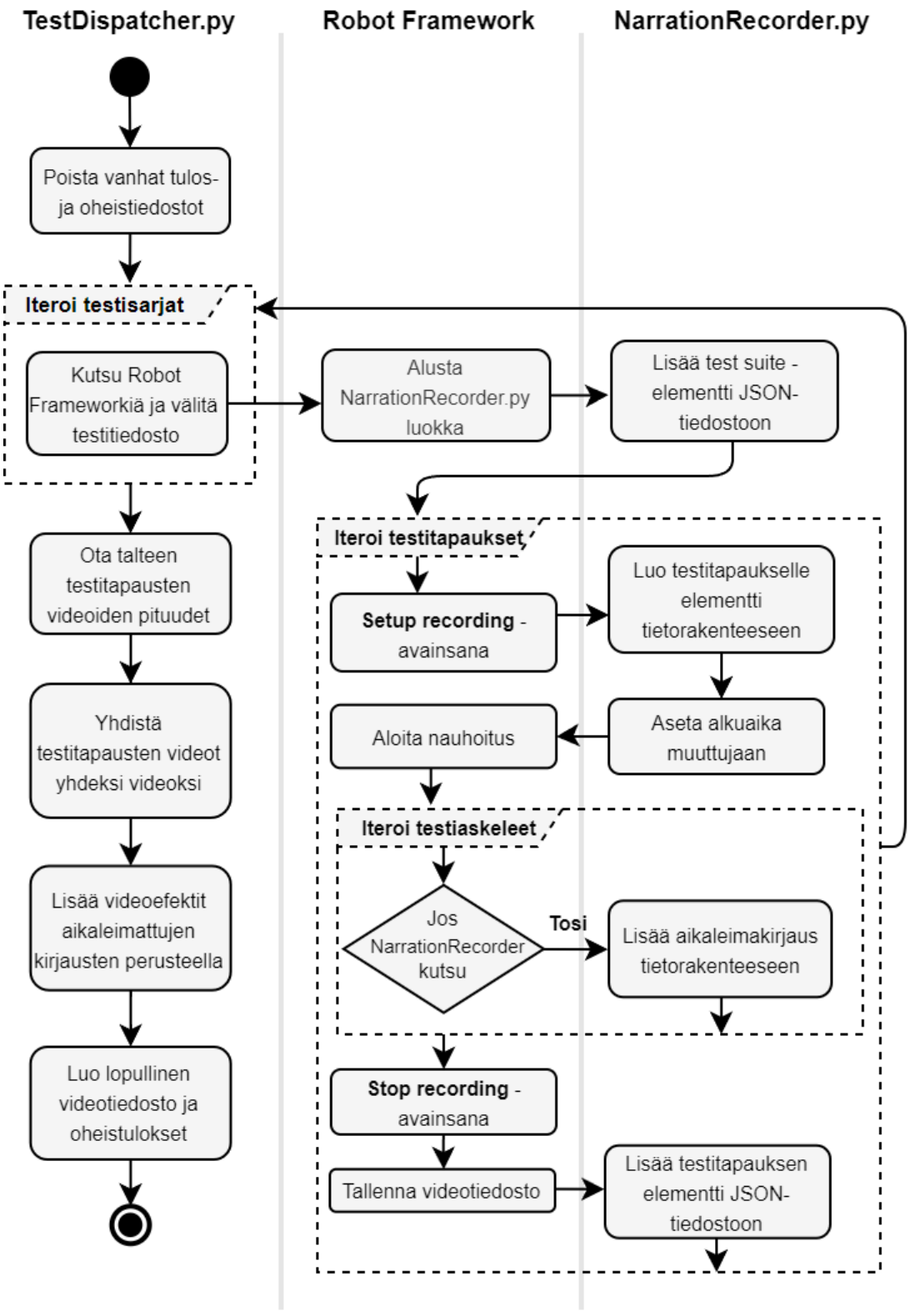
4.3 Ohjelman rakenne ja toiminta

Ohjelman perusrakenne noudattaa proseduraalista ohjelmointia, jossa yksi pääohjelma kutsuu aliohjelmia mukaan lukien Robot Frameworkiä. Kuva 4 esittää eri komponenttien toiminnan vuokaaviona. Ennen testien suorittamista aiemmin generoidut tulos- ja oheistiedostot poistetaan kansioista. *TestDispatcher.py*-pääohjelmalle annetaan argumentteina yksi tai useampi robot-testisarja, jotka se välittää Robot Frameworkille suoritettaviksi. Opasvideo muodostetaan testisarjojen testitapauksista annetussa järjestyksessä. Robot Frameworkin suorituksen aikana testisarjojen testitapaukset nauhoitetaan videotiedostoiksi. *NarrationRecorder.py*-moduulin vastuulla on kirjoittaa JSON-tiedostoon testin aikana tehdyt kirjaukset aikaleimoineen. Kun testitapaukset on suoritettu ilman virheitä pääohjelma kasaa MoviePy-kirjaston avulla videotiedostot yhteen ja lisää kirjauksien mukaiset erikoistehosteet osaksi videota.

NarrationRecorder.py on Robot-kirjastona käytetty Python-luokka joka hoitaa aikaleimojen ja kirjauksien lisäämisen *NarrationRecord.json*-tiedostoon (liite C). Testisarjan alussa luokka alustetaan ja se lisää JSON-tiedoston juuritaulukkoon testisarjaa vastaavan JSON-olion (*suite*), jolla on testisarjan nimi sekä taulukko testitapauksille. Testitapauksen tietorakenteella on vastaavasti myös nimi ja taulukko aikaleimattuja kirjauksia varten (*records*). Testitapauksen tietorakenne lisätään JSON-tiedostoon vasta testitapauksen lopussa. Testitapauksen alussa otetaan nauhoituksen aloitusaika talteen *DateTime*-oliona, jotta kirjatut aikaerot voidaan johtaa siitä sekunteina ja lisätä JSON-kirjauksiin.

Nauhoituksen aloituksessa, lopetuksessa ja kirjausten tekemisessä käytetään avainsanoja, jotka on määritelty *screencast.resource*-tiedostossa (liite B). Avainsanoista tärkeimmät ovat *Setup recording* ja *Stop recording*. *Setup recording* laittaa Chrome-selaimen kokonäytöntilaan, asettaa Seleniumin toiminnoilla halutun nopeuden (joka voidaan antaa argumenttina) ja pyytää ScreenCapLibraryä aloittamaan näytön nauhoittamisen. *Stop recording* -avainsana lopettaa nauhoittamisen ja pyytää *NarrationRecorder.py*-luokkaa lisäämään kirjaukset JSON-tiedostoon. Avainsanojen ei välttämättä tarvitse olla testisuorituksen alussa ja lopussa. Testien laatija voi aloittaa ja lopettaa nauhoittamisen opasvideolle mielekkäässä vaiheessa.

Kirjauksia tekeviä avainsanoja voidaan sijoittaa testisarjassa nauhoituksen aikaisille riveil-



Kuva 4. Vuokaavio ohjelman toiminnasta.

le eli *Setup recording* ja *Stop recording* -avainsanojen välille. *Add narration* -avainsana ottaa argumentikseen tekstityksessä käytettävän merkkijonon ja pyytää tekstitystä varten odottamaan 3 sekuntia ennen seuraavaa toimintoa *Sleep*-komennolla. *Hilghlight element* -avainsana ottaa argumentikseen HTML-elementin tunnisteen ja käyttää *getBoundingClientRect*-JavaScript-metodia selainnäkymän pikselikoordinaattien kaappaamiseen.

Testisarjojen suorituksen päätyttyä muodostetaan lopulliset tulostiedostot *Results*-kansioon. Testitapausten videotiedostot järjestetään niiden luontiajan mukaan jonoksi ja yhdistetään MoviePyn avulla yhdeksi *RawVideo.mp4* videotiedostoksi. Videotiedostojen tarkat kestoajat otetaan talteen ja lisätään vastaavien testitapausten kirjauksien aikoihin. Näin saadaan lopulliset aikaleimat, jotka vastaavat yhdistetyn videon eri osioita. Testitapausten mukaan nimetyt aikaleimat sijoitetaan tuloskansion *Timestamps.txt*-tiedostoon. Mitään tuloksia tai oheistiedostoja ei tuhota ennen kuin ohjelma suoritetaan uudestaan.

Lopullinen jälkikäsitelty opasvideo muodostetaan MoviePyn avulla. Eri osioiden alkuihin lisätään aikaleimoja vastaavat otsikkotekstit. HTML-elementtejä korostetaan ohuilla punaisilla suorakulmioilla, jotka sijoitetaan elementtiä vastaavaan rajaavan laatikon koordinaatteihin. Suorakulmiot piirretään Gizeh-kirjaston avulla läpinäkyvään kuvaan, jonka resoluutio vastaa videon tarkkuutta ja tallennetaan *Overlays*-kansioon. Kuvat lisätään sitten leikkeinä aikaleiman mukaiseen kohtaan videon päälle. Tekstityksiä lisätessä fontin koko määräytyy näytön resoluution mukaan. Videota ei tarvitse jälkikäsittelyssä pysäyttää tekstityksen ajaksi. Toteutuksessa Robot Framework pysähtyi jo nauhoituksen aikana kolmeksi sekunniksi, kun tekstitystä kirjaava avainsana tuli vastaan.

4.4 Esimerkki ohjelman toiminnasta

Demonstroidaan ohjelman käyttöä yksinkertaisen testisarjan avulla. Testisarjan kahdessa testitapauksessa tehdään hakukonehaut Googlen ja Bingin avulla. Liite D sisältää koko testisarjan sisällön. Python-pääohjelmalle voi antaa argumentteina halutun määrän robot-testisarjoja. Kutsumuoto *Tests*-kansiossa olevalle *Hello1.robot*-testisarjalle on muotoa:

```
py .\TestDispatcher.py .\Tests\Hello1.robot
```

Pääohjelmassa kutsutaan Pythonin *robot*-kirjaston avulla Robot Frameworkiä niin, että Robot Frameworkin omia tuloksia (loki-, raportti- ja tulostustiedostoa) ei luoda. Tuloskansio *RecordedClips* kuitenkin määritellään Robot Frameworkin ScreenCapLibraryä varten. Robot Frameworkiä kutsutaan pääohjelmassa testisarja kerrallaan muodossa:

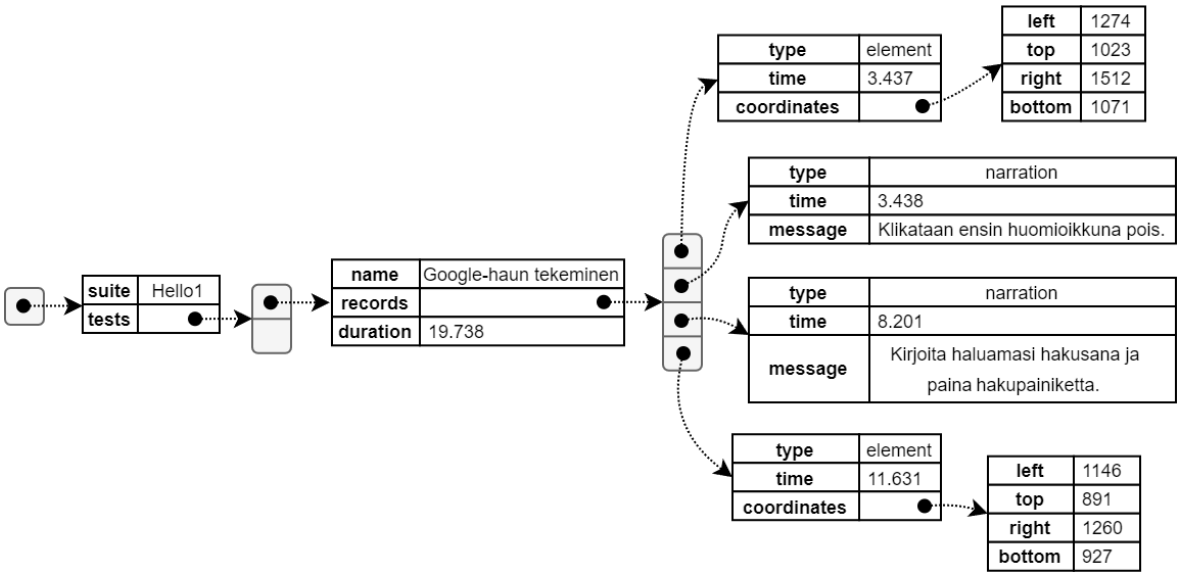
```
robot.run(sys.argv[i], log='NONE', report='NONE', output='NONE',
          outputdir='RecordedClips')
```

Testisarjan alussa tuodaan tarvitut kirjastot sekä resurssitiedosto, joka sisältää opasvideon laatimista varten tarvitut avainsanat. *NarrationRecorder*-kirjaston tuominen alustaa kirjauksia tekevän Python-olion ja lisää JSON-tiedostoon testisarjaa vastaavan elementin. Kirjastojen tuominen testisarjan alussa on muotoa:

```
*** Settings ***
Resource    ../Resources/screencast.resource
Library    NarrationRecorder    ${SUITE NAME}    WITH NAME    Narrator
Library  SeleniumLibrary    run_on_failure=Nothing
Library  ScreenCapLibrary
```

Kokonaiset ohjelmistolistaukset laaditulle resurssitiedostolle ja kirjastolle ovat liitteissä B ja C. *Setup recording* -avainsanan jälkeen avainsanoilla *Add narration* ja *Highlight element* voidaan lisätä aikaleimattuja kirjauksia. Ensimmäisessä testitapauksessa korostetaan ensin painiketta, jolla Google-sivun huomioikkuna piilotetaan ja seuraavaksi sivun hakupainiketta. Esimerkissä Seleniumin *Click Button* -käskyt käyttävät HTML-elementin määrityksessä samoja lokaattoreita kuin *Highlight element* -käskyt, mutta joskus voi olla mielekästä korostaa videolla esimerkiksi painikkeen vanhempielementtiä. Ensimmäinen testitapaus on muotoa:

```
*** Test Cases ***
Google-haun tekeminen
    Open Browser    http://www.google.com    ${BROWSER}
    Setup recording
    Highlight element    id=L2AGLb
    Add narration    Klikataan ensin huomioikkuna pois.
    Click Button    id=L2AGLb
    Input Text    name=q    Hello world!
    Add narration    Kirjoita haluamasi hakusana ja paina hakupainiketta.
    Highlight element    name=btnK    2
    Click Button    name=btnK
    Stop recording
```

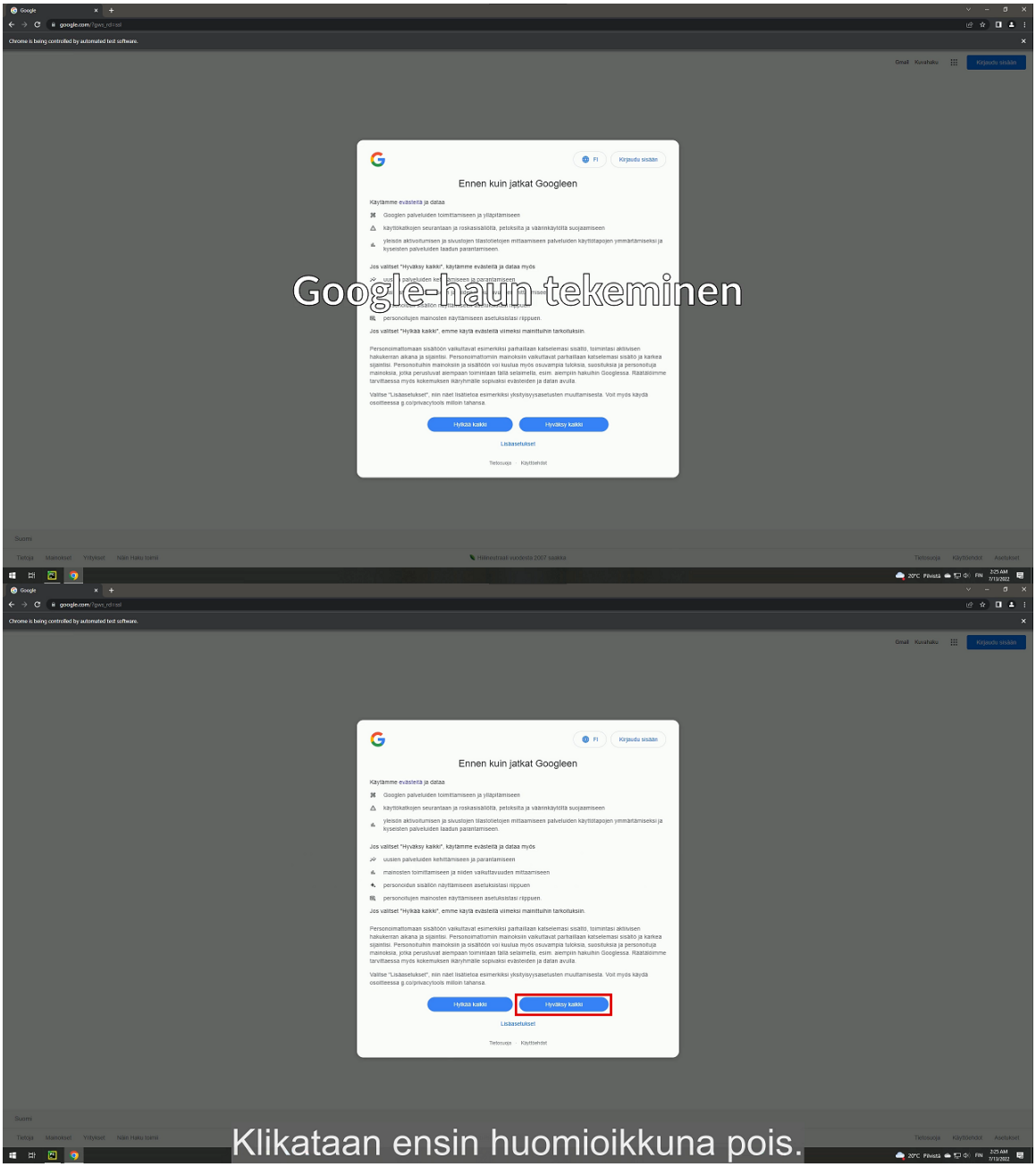


Kuva 5. JSON-tiedoston kirjaukset ensimmäisestä testitapauksesta.

Testitapauksen kirjaukset lisätään JSON-tiedostoon vastaavaan testisarjaan *Stop recording* -avainsanan kohdalla. Yllä olevasta testitapauksesta muodostettu JSON-rakenne on esitetty kuvassa 5. JSON-tiedosto alkaa nimetyistä testisarjoista koostuvasta taulukosta. Testisarja sisältää taulukon, joka koostuu testitapauksista. Testitapauksella on nimi, kesto sekä oma taulukkonsa kirjauksia. Kirjaukset sisältävät aina tyypin, ajan ja videon käsittelyyn tarvittavan informaation. Tekstityksessä käytetty *narration*-tyypin olio sisältää merkkijonon ja *element*-tyyppi sisältää viitteen pikselikoordinaatit määrittelevään olioon. Työkalun jatkokehityksessä on mahdollista lisätä uuden tyyppisiä kirjauksia JSON-rakenteeseen.

Kuva 6 sisältää kaksi kuvakaappausta lopullisesta opasvideosta. Ylempi kuva on videon alusta ja siinä näkyy testitapauksen nimestä muodostettu otsikko *Google haun tekeminen*. Alemmassa kuvassa näkyy samaan aikaan tekstitys sekä painiketta korostava punainen suorakulmio. Liite E esittää sovelluksen kansiorakenteen suorituksen jälkeen. Tulostiedostoja ovat lopullinen opasvideo *HardCodedSubtitles.mp4*, video ilman MoviePy-efektejä *RawVideo.mp4* ja aikaleimat eri osioihin *Timestamps.txt*. Testitapausten aikaleimat ovat tiedostossa muotoa:

```
00:00: Google-haun tekeminen
00:19: Bing-haun tekeminen
```



Kuva 6. Generoidun opasvideon kuvakaappaukset.

5 Arviointi ja pohdinta

Tässä luvussa arvioidaan työkalun avulla generoituja opasvideoita aliluvun 2.5 arviointikriteerien mukaisesti ja listataan tutkimuksen aikana koottuja kehitysideoita työkalulle. Lopuksi vastataan tutkimuskysymyksiin ja ehdotetaan mahdollisia suuntia jatkotutkimukselle.

5.1 Generoitujen opasvideoiden arviointi

Arvioidaan aliluvussa 2.5.1 esiteltyjen Mayerin periaatteiden täyttymistä generoitujen opasvideoiden osalta. Aloitetaan tiedonkäsittelyn vaativuutta vähentävistä periaatteista, joita ovat yhtenäisyys, signalointi, toisteen välttäminen, alueellinen vierekkäisyys ja ajallinen läheisyys. Yhtenäisyysperiaatteen mukaan sisällön ei tulisi sisältää työmuistia kuormittavaa animaatiota tai musiikkia. Opasvideoihin ei lisätä musiikkia tai ääniefektejä ja käyttöliittymäkomponenttien korostus on tyyliltään yksinkertaista, joten periaate täyttyy. Signalointiperiaate täyttyy visuaalisesti käyttöliittymäkomponenttien korostuksen ansiosta. Tärkeiden ideoiden ja käsitteiden esiin nosto on testien laatijan vastuulla. Toisteen välttäminen on myös ensisijaisesti testien laatijan vastuulla, mutta työkalu ei ylipäätään salli esimerkiksi usean tekstityksen näyttämistä näytöllä kerralla. Alueellinen vierekkäisyys ei täyty tekstin kohdalla, koska tekstitykset sijaitsevat aina samassa paikassa. Ajallisen läheisyyden tulisi aina täyttyä, kunhan tekstitykset ovat asiaankuuluvien testiaskeleiden vieressä.

Oppimateriaalin haastavuuden hallintaan liittyvät periaatteet ovat osiin jakaminen, aiheeseen valmennus ja modaliteetti. Osiin jakaminen onnistuu luontaisesti työkalun avulla, kunhan hyödynnetyt testitapaukset sisältävät nimensä mukaiset käyttötapaukset. Aiheeseen valmentaminen voi täyttyä, mikäli testitapauksen alkuun lisätään tarvittaessa prosessia ja käsitteitä selittävää tekstitystä. Tällöin kohdesovelluksen demonstrointi kuitenkin vain pysähtyy videolla eikä toistaiseksi ole mahdollista näyttää esimerkiksi valmennusta edesauttavaa animaatioita tai muunlaista esitystä. Modaliteettiperiaatteen mukaan audioverbaalisesta ja visuaalisesta esitysmuodosta tulisi valita sopivin niin ettei yhtä kanavaa ylikuormiteta. Opasvideoihin ei voi lisätä puheselostusta, mutta kohdesovelluksen suorittaminen pysähtyy aina tekstityksen ajaksi, joten visuaalinen kanava ei ylikuormitu.

Opitun asian sisäistämistä edistävät periaatteet ovat multimedian hyödyntäminen ja personointi. Multimediaperiaateen mukaan sekä audioverbaalista että visuaalista kanavaa tulee hyödyntää opettamisessa. Multimediaperiaate toteutuu osittain, koska videot hyödyntävät kuvamateriaalia sekä sanallista kuvausta. Audioverbaalista kanavaa ei kuitenkaan täysin hyödynnetä puheselostuksen puutteen takia. Personointiperiaatteen mukaan tulisi suosia epävirallista puhetyyliä. Ilman puheselostusta periaate jää pitkälti toteuttamatta, mutta tekstityksessä voidaan käyttää rentoa keskustelutyyliä.

Arvioidaan opasvideoiden hyvien käytänteiden toteutumista. Työkalun avulla opasvideot voidaan pitää sisällöltään helposti ajantasaisina. Jos muutos ohjelmaan on visuaalinen eikä koske testitapaukseen liittyvää prosessia, opasvideot voidaan vaivattomasti nauhoittaa uudelleen. Työkalun käyttäjän vastuulla on valita testisarjat, jotka muodostavat sopivan pituisen (alle 6 minuuttia pitkän) opasvideon. Videon resoluutio määräytyy käytetyn laitteen mukaan, jolla testitapaukset suoritetaan. Käyttäjän tulee siis asettaa näyttö halutulle tarkkuudelle ennen testien suorittamista. Saavutettavuuteen osalta videoon kovakoodatut tekstitykset eivät ole ihanteellisia. Erillisten tekstitysraitojen etuna olisi, että soittimesta riippuen ne voidaan vaihtaa eri kielille, niiden avulla voidaan löytää haluttu kohta videolta ja niitä voidaan hyödyntää tekstistä puheeksi (*text-to-speech*) -puhesynteesissä.

5.2 Jatkokehitysideoita

Tutkimuksen aikana koottiin kehitysideoita sille, miten kehitettyä työkalua voitaisiin parantaa käytettävyyden ja generoitujen opasvideoiden osalta. Kehitysideoita kerääntyi sekä kehityksen että arvioinnin aikana. Lisäksi generoituja videoita verrattiin käsintehtyyn näyttökaapattuun opasvideoon, joka demonstroi laajemman verkkosovelluksen käyttöä.

5.2.1 Opasvideoiden parantaminen

Arvioinnista kävi ilmi, että rajoittavin tekijä generoiduissa opasvideoissa on puheselostuksen puute. Mayerin multimediaperiaatteen mukaan audioverbaalisen kanavan hyödyntäminen tukee paremmin oppimista. Mayerin modaliteettiperiaatteen mukaan puheselostus sopii pelkkää tekstitystä paremmin kun näytöllä on samanaikaisesti toimintaa. Puheen voisi to-

teuttaa avainsanojen avulla kahdella eri tavalla, joista helpompi olisi koneäänen lisääminen suoraan videoon puhesyntetisaattorin avulla. Mayerin personointiperiaatetta ajatellen monotonista koneääntä ei välttämättä kuunnella yhtä sitoutuneesti kuin aidon ihmisen nauhoitettua puhetta. Toinen vaihtoehto olisi hyödyntää valmiiksi nauhoitettuja äänitiedostoja joihin viitataan avainsanoissa. Tällöin puheääni noudattaisi personointiperiaatetta paremmin ja opasvideot antaisivat laadukkaamman vaikutelman käyttäjälle. Äänitiedostot tulisi kuitenkin aina nauhoittaa uudelleen, kun selostukseen tehdään lisäyksiä. Äänitiedostoihin viittaavat avainsanat voisivat samalla lisätä tekstityksen mikä tekisi avainsanoista luettavampia ja antaisi tekstityksen kestolle tarkan arvon.

Generoiduista opasvideosta löytyi ominaisuuksia, jotka poikkeavat tavasta, jolla ihminen tyypillisesti käyttää verkkosovellusta. Suurin heikkous Seleniumin käytössä on se, ettei hiiren kursori ole näkyvillä ollenkaan demonstraation aikana. Tämä rikkoo visuaalista jatkuvuutta ja siten vaikeuttaa ohjelman käytön seuraamista. Ainut tapa nykyisessä toteutuksessa hyvittää ongelmaa on käyttää runsaasti HTML-elementtejä korostavaa avainsanaa, kun liikutaan käyttöliittymän osien välillä. Elementin korostuksesta toiseen siirtyessä voitaisiin hyödyntää liikkuvaa animaatiota, jotta visuaalinen jatkuvuus säilyisi paremmin. Lisäksi tekstikenttiä käsitellessä koko tekstisyöte lisätään kenttään välittömästi. Hitaampi tekstinsyöttö voisi helpottaa ohjelman käytön seuraamista. Ihmisen kirjoitusta voitaisiin jäljitellä toteuttamalla oma avainsana tekstin syötölle, jossa odotetaan hyvin pieni hetki jokaisen merkin syöttämisen välillä.

Käsintehdyllä opasvideolla esiintyi välillä tarve sijoittaa tekstitykset ylälaitaan, jotta ne eivät peitä tärkeitä käyttöliittymän osia. Tekstitysavainsanalle voitaisiin lisätä vaihtoehtoinen parametri, joka määrittelee tekstitykselle oletusarvosta poikkeavan sijainnin, mikäli tekstitykset ovat kovakoodattuja tai tekstitysformaatti sallii sen. Käsintehdyllä opasvideolla käyttöliittymäkomponenttien korostuksessa käytettiin nuolia, jotka sisälsivät tekstikuvauksen elementin roolista. Tekstilliset nuolet vastaavat tekstityksiä paremmin Mayerin alueellisen vierekkäisyyden periaatetta. Nuolen luova avainsana olisi käytännössä tekstitys- ja korostusavainsanojen yhdistelmä, joka ottaisi sekä merkkijonon että HTML-elementin lokaattorin argumenteikseen. Nuolen orientaatio voitaisiin määritellä algoritmisesti pikselikoordinaattien mukaan. Esimerkiksi näytön oikealla puolella olevaan elementtiin osoittava nuoli on sii-

tä katsoen vasemmalla. Nykyisestä HTML-elementtiä korostavasta avainsanasta voitaisiin myös tehdä älykkäämpi niin että korostus tapahtuu eri muodoilla ja väreillä riippuen esimerkiksi siitä, onko elementti syöttökenttä tai painike. Tämä kuitenkin voi tulla haasteelliseksi ylläpitää, mikäli käytettyjen elementtien nimet ja rakenteet poikkeavat tavanomaisista Web-komponenteista.

Tietyissä kohdissa käsintehtyä opasvideota oli tarvetta leikata ja nopeuttaa videota, kun ruudulla odotettiin pitkävetoista prosessia. Tämän voisi toteuttaa avainsanaparilla, joka pyytää MoviePyta muokkaamaan videon nopeutta tietystä kohdasta toiseen kohtaan joko nopeuttaen tai kokonaan leikaten avainsanojen välisen osion. Käsintehdyllä opasvideolla oli kuva kuvassa (*picture-in-picture*) -osio, jolla rinnakkain näytettiin miltä tuotteeseen liittyvän mobiilisovelluksen tulisi näyttää siinä vaiheessa prosessia. Tämä voitaisiin lisätä avainsanana, joka sijoittaa kuva- tai videotiedoston päävideon rinnalle haluttuun kohtaan. Sovelluksen käytöstä saatetaan haluta demonstroida näppäinpainalluksia ja -yhdistelmiä. Niitä varten voitaisiin luoda toiminnallisuuden käärivä avainsana, joka toteuttaa näppäinpainallukset sekä lisää ne videon kulmaan tekstimuodossa näkyville.

Opasvideot keskittyvät testitapausten edustamien käyttötapausten suorittamiseen, eivätkä tarjoa aiheeseen valmentamista joka Mayerin mukaan voi edesauttaa oppimista. Yksinkertaisin ratkaisu olisi pitää aiheeseen johdattelevat videot erillään sovellusta demonstroivista näyttökaapatuista videoista. Opasvideon alkuun tai loppuun voitaisiin määritellä valmiiksi laadittu aiheeseen johdatteleva video, joka liitetään osaksi lopullista opasvideota. Video voisi avartaa kohteena olevan ohjelman keskeisiä käsitteitä ja toimintaperiaatetta, jotka eivät välttämättä käy ilmi pelkästä demonstroinnista. Käsitteet ja toimintaperiaatteet todennäköisesti muuttuvat sovellusta harvemmin, joten videoita voitaisiin uudelleen käyttää vaivatta. Samalla toiminnolla voitaisiin lisätä myös esimerkiksi yritystä edustavat johdanto- ja lopetusvideot.

Opasvideoiden lisäksi nykyinen työkalu voisi generoida aliluvussa 2.4 mainitunlaisia multimediaoppaita, jotka kuvaavat käyttötapauksia askel askeleelta. Robot Framework mahdollistaa videoiden nauhoituksen ohella myös kuvakaappauksen ja työkalu käyttää jo valmiiksi kuvankäsittelykirjastoa. Multimediaoppaat voisivat esimerkiksi verkkosivun muodossa yhdistää tekstikappaleita, kuvakaappauksia, videoita sekä aikaleimoja ja niihin liittyviä pik-

kukuvia. Uudet avainsanat eivät siis liittyisi pelkästään videon sisältöön vaan koko oppaan ulkoasuun. Tämän pitäisi myös tulla esiin kirjaukset sisältävästä JSON-rakenteesta. Oppaan sisällysluettelo voitaisiin muodostaa testitapausten nimistä samaan tapaan kuin opasvideon otsikoidut osiot tällä hetkellä.

5.2.2 Työkalun käytettävyyden parantaminen

Avainsanojen toteutuksessa on tehty tiettyjä oletuksia, jotka rajoittavat työkalun käyttöä. Testitapausta kohden voidaan nauhoittaa vain yksi videotiedosto, ja otsikointi tehdään aina testitapauksen nimen mukaan. Tämä on ongelmallista erityisesti jos käyttötapaus jota videolla halutaan demonstroida koostuu useasta testitapauksesta. Nykyiset avainsanat ovat myös tiiviissä yhteistyössä Seleniumin kanssa, mikä rajoittaa alustan työpöydän verkkosovelluksiin. Avainsanat voitaisiin jakaa erillisiin resurssitiedostoihin joissa nauhoitukseen ja kirjaukseen liittyvät avainsanat ovat erillään ja niistä koostettaisiin alustakohtaiset avainsanat. Testien luettavuuden kannalta voisi olla kannattavaa kääriä toiminnallisuus (esim. tekstin syöttö) ja kirjaus (esim. tekstikentän korostus) yhteen avainsanaan.

Toistaiseksi kehitetty työkalu toimii vain yhdellä kutsumuodolla. Toiminnallisuuden osalta testien suorittamisen ja tulostiedostojen muodostamisen voisi suorittaa vaihtoehtoisesti erillisillä käskyillä. Tämän etuna olisi, että käyttäjä voi tarkastella ja muokata videoita sekä JSON-tiedostoa ennen kuin opasvideo muodostetaan. JSON-tiedoston tekstitykset voitaisiin esimerkiksi kääntää toiselle kielelle, ettei opasvideoita tarvitse nauhoittaa uudestaan jokaista kieltä varten. Toisaalta uudelleennauhoitus voi olla tarpeen, mikäli kohdesovellusta halutaan esittää videolla eri kielillä.

Työkaluun voitaisiin lisätä asetustiedoston avulla muokattavuutta. Siihen kirjoittamalla voitaisiin säätä opasvideon visuaalisia ominaisuuksia, kuten tekstityksen ja otsikoiden ulkoasua, elementtien korostuksen tyyliä sekä muita yksityiskohtia joita halutaan muokata nauhoituksen ja videon jälkikäsittelyn aikana. Tiedostossa voitaisiin määrittää myös työkalun kutsussa esiintyvät testisarjat. Työkalulta voitaisiin pyytää asetustiedostossa halutut tulostiedostot ja niiden formaatit. Tällöin voidaan säästää suoritusajassa generoimalla vain halutut tulokset.

Tekstityksen kirjauksen yhteydessä Robot Framework odottaa tekstitystä varten hetken en-

nen seuraavaan testiaskeleeseen siirtymistä. Tämä hidastaa huomattavasti nauhoitettavien testien suoritusta. Pysähtyminen voitaisiin toteuttaa jälkituotannossa MoviePyn avulla muokkaamalla videon nopeutta halutuissa kohdissa. Tällöin tulisi myös muokata kirjattuja aikaleimoja käsittelystä aiheutuneen aikaeron verran. Työkalu voisi myös tarjota vaihtoehtoisen avainsanan tekstitykselle joka ei pysäytä ohjelman demonstrointia. Tällöin tulisi joko toteutuksessa huomioida tekstitysten mahdollinen päällekkäisyys tai jättää ongelma testien laatijan vastuulle.

5.3 Tutkimuskysymysten arviointi ja jatkotutkimus

Tässä aliluvussa arvioidaan tutkimuksen tavoitteiden toteutumista vastaamalla tutkimuskysymyksiin. Ensimmäisen tutkimuskysymyksen tavoite oli selvittää, miten opasvideoiden generointi vaikuttaa testitapausten laatimiseen. Ylimääräiset avainsanat voivat hankaloittaa testin luettavuutta. Testien laatijan tulee lisätä avainsanat nauhoitettavan osion alkuun, loppuun ja niihin kohtiin joihin opasvideossa halutaan lisätä tekstitystä tai HTML-elementin korostusta. Testitapausten nimet tulee olla selkokielisiä ja käyttötapauksia kuvaavia, sillä niitä käytetään videon otsikoinneissa. Testitapausten nauhoituksen aikana testiaskeleiden suoritusnopeus asetetaan hitaammaksi katsojaa varten. Työkalu suorittaa kaikki annetut testitapaukset, mutta nauhoittaa vain ne, jotka sisältävät vaaditut avainsanat.

Nauhoitukseen tarkoitettuja testisarjoja voidaan periaatteessa käyttää edelleen testaamiseen. Olisi kuitenkin vahvasti suositeltua koota opasvideoiden generointiin tarkoitetut testitapaukset erillisesti ylläpidettyihin testisarjoihin. Kansiorakenne ja työkalun toimintaperiaate pysyisi selkeämpänä, jos opasvideoiden generoinnin aikana ei samalla muodosteta testituloksia. Käsintehdystä opasvideosta kävi ilmi, että järjestys missä toiminnallisuuksia testataan tavanomaisessa testisarjassa ei välttämättä vastaa mielekästä demonstrointia opasvideolla. Testaamisessa pyritään mahdollisimman kattavasti varmistamaan, että kaikki ohjelman osat toimivat odotusten mukaisesti. Opasvideon selostuksen ansiosta sovelluksesta voidaan antaa kattava kuva lyhyessä ajassa ilman että jokaista ohjelman tilaa ja käyttöliittymän osaa tulee erikseen demonstroida.

Toinen tutkimuskysymys pyrki arvioimaan kuinka hyvin generoidut videot täyttävät opasvi-

deoiden kriteerit. Tähän vastattiin kattavammin aliluvussa 5.1. Osa opasvideoiden arviointikriteereiden täyttymisestä tulee olemaan aina testien laatijan vastuulla. Opasvideoissa suurimmat rajoitukset ovat puheselostuksen puute sekä alueellisesti vierekkäisen tekstin puute. Aliluvussa 5.2.1 esitettiin ehdotuksia miten nämä puutteet voidaan korjata kehittämällä uusia avainsanoja työkalulle. Tällä hetkellä opasvideot täyttävät suurimman osan arviointikriteereistä. Mikäli jatkokehitysideat toteutetaan, generoidut opasvideot voivat täyttää jokaisen arviointikriteerin. Työkalun arkkitehtuuri ei ole tälle este.

Kolmas tutkimuskysymys kysyi yhteenvetona: mitä opasvideoiden generoinnilla voidaan saavuttaa. Motiivina työkalun kehitykselle oli säästä yrityksen resursseja, aikaa ja työntekijöiden kuormaa. Opasvideoita nauhoitettaessa käsin ohjelman kattava demonstrointi sopivassa tahdissa on taito itsessään ja johtaa helposti pieniin virheisiin. Uuden ohjelmistoversion julkaisussa uudet opasvideot voidaan generoida työkalun avulla vaivatta automatisoidusti, kunhan testitapaukset pidetään ajan tasalla. Opasvideoiden pituutta voi myös olla helpompi arvioida generoinnin avulla ja pitää ne käyttäjän kannalta mielekkään kestoisina. Uuden ohjelmistoversion käyttöönotossa Ideaali olisi liittää kehitetty työkalu osaksi automatisoitua työkaluketjua, joka esimerkiksi sijoittaa opasvideon ja aikaleimat oikeaan kohtaan verkkosovelluksessa.

Työkalun tarkoituksenmukaisuutta perusteltiin kuvailevalla arviointimenetelmällä hyödyntäen aiempaa tietopohjaa. Tämän tutkimuksen puitteissa todellisia vaikutuksia liiketoimintaympäristöön on hankala arvioida ilman jatkotutkimusta. Kehitetyn työkalun taloudellista kannattavuutta voitaisiin arvioida havainnollistavasti tapaustutkimuksen muodossa. Lisäksi työkalulle koottiin lukuisia jatkokehitysideoita, jotka voitaisiin toteuttaa. Työkalun voisi esimerkiksi muokata generoimaan opasvideoiden lisäksi yleisiä multimediaoppaita. Jatkokehityksen jälkeen työkalua ja sillä generoituja tulostiedostoja voitaisiin uudestaan arvioida. Tutkimuksen ulkopuolelle jäi sen selvittäminen voisiko dokumentaatiokattavuutta mitata yhdessä testikattavuuden kanssa, mikä voisi myös tarjota aihetta uudelle tutkimukselle.

6 Yhteenveto

IT-tuotteiden opasvideot voivat tarjota sekä aloittelijoille että kokeneemmille käyttäjille kustannustehokasta ja konkreettista opastusta. Tämän tutkimuksen tavoitteena oli selvittää voiko testiautomaatiota hyödyntää loppukäyttäjälle suunnattujen näyttökaapattujen opasvideoiden generoinnissa. Tiedettävästi opasvideoita ei ole ennen yritetty generoida testiautomaatiota hyödyntäen. Motiivina automatisoidulle generoinnille on säästää resursseja, aikaa ja kehittäjien kuormaa, kun uuden sovellusversion käyttöönoton yhteydessä nousee tarve uudelleen nauhoittaa siihen liittyvät opasvideot.

Tässä tutkimuksessa esiteltiin suunnittelutieteen periaatteilla kehitetty työkalu, joka hyödyntää erikseen laadittuja Robot Framework -testejä verkkosovellusten opasvideoiden generoinnissa. Testitapauksia täydennettiin työkalun omilla avainsanoilla joiden avulla määriteltiin, miten opasvideot nauhoitetaan ja jälkikäsitellään halutunlaisiksi. Generoituja videoita arvioitiin kuvailevasti hyödyntäen aiempaa tietopohjaa. Tärkeimmät kriteerit arvioinnissa olivat Mayerin multimediaoppaiden periaatteiden täyttyminen. Lisäksi tutkimuksessa haluttiin selvittää, miten opasvideoiden generointi vaikuttaa testitapausten laatimiseen ja mitä videoiden generoinnilla voidaan saavuttaa

Generoidut opasvideot täyttivät suurimman osan arviointikriteereistä ja toteutettavissa olevien jatkokehitysideoiden täytyttyä ne voivat vastata hyvin käsintehtyjä opasvideoita. On suositeltavaa, että generoinnissa käytetyt testisarjat suunnitellaan ensisijaisesti opasvideota varten. Testitapauksen nimen ja laajuuden tulisi vastata yhtä osiota videolla, eikä normaalien testien tavoin ole välttämätöntä käydä läpi jokaista käyttöliittymän osaa ja ohjelman tilaa. Tutkimuksessa hyödynnetty kuvaileva arviointimenetelmä on todistusarvoltaan rajallinen, ja lisätutkimusta tarvitaan sen varmistamiseksi, että työkalu todella vastaa liiketoimintaympäristön tarpeita.

Lähteet

Brame, Cynthia J. 2016. “Effective educational videos: Principles and guidelines for maximizing student learning from video content”. *CBE—Life Sciences Education* 15 (4): es6.

Chi, Pei-Yu, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li ja Björn Hartmann. 2012. “MixT: automatic generation of step-by-step mixed media tutorials”. Teoksessa *Proceedings of the 25th annual ACM symposium on User interface software and technology,* 93–102.

Colvin Clark, Mayer. 2011. Teoksessa *e-Learning and the Science of Instruction.* John Wiley Sons, Ltd.

Descher, Marco, Thomas Feilhauer ja Lucia Amann. 2014. “Automated user documentation generation based on the Eclipse application model”. *arXiv preprint arXiv:1411.3818.*

Doll, William J, ja Gholamreza Torkzadeh. 1987. “The quality of user documentation”. *Information & management* 12 (2): 73–78.

———. 1993. “The place and value of documentation in end-user computing”. *Information & Management* 24 (3): 147–158.

Gemoets, Leopoldo A, ja Mo Adam Mahmood. 1990. “Effect of the quality of user documentation on user satisfaction with information systems”. *Information & management* 18 (1): 47–54.

“Gizeh Github”. 2018, marraskuu. https://github.com/Zulko/gizeh.

Gregor, Shirley, ja Alan R Hevner. 2013. “Positioning and presenting design science research for maximum impact”. *MIS quarterly,* 337–355.

Guo, Philip J, Juho Kim ja Rob Rubin. 2014. “How video production affects student engagement: An empirical study of MOOC videos”. Teoksessa *Proceedings of the first ACM conference on Learning@ scale conference,* 41–50.

Hevner, Alan R. 2007. “A three cycle view of design science research”. *Scandinavian journal of information systems* 19 (2): 4.

Hevner, Alan R, Salvatore T March, Jinsoo Park ja Sudha Ram. 2004. “Design science in information systems research”. *MIS quarterly,* 75–105.

Iwata, Hajime, Junko Shirogane ja Yoshiaki Fukazawa. 2006. “Automatic generation of tutorial systems from development specification”. Teoksessa *International Conference on Fundamental Approaches to Software Engineering,* 79–92. Springer.

Johnson, Cheryl I, ja Richard E Mayer. 2009. “A testing effect with multimedia learning.” *Journal of Educational Psychology* 101 (3): 621.

Kharishma, Vidya. 2020. “Design a screencast video for software learning in higher education (case study: Tutorial video for digital illustration course)”. *JISA (Jurnal Informatika dan Sains)* 3 (1): 15–20.

Klärck, Pekka. 2019. “How to write good test cases using Robot Framework”. https://github.com/robotframework/HowToWriteGoodTestCases/blob/master/HowToWriteGoodTestCases.rst.

Li, Kunpeng, Chen Fang, Zhaowen Wang, Seokhwan Kim, Hailin Jin ja Yun Fu. 2020. “Screencast tutorial video understanding”. Teoksessa *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 12526–12535.

March, Salvatore T, ja Gerald F Smith. 1995. “Design and natural science research on information technology”. *Decision support systems* 15 (4): 251–266.

Martin, Nichole A, ja Ross Martin. 2015. “Would you watch it? Creating effective and engaging video tutorials”. *Journal of Library & Information Services in Distance Learning* 9 (1-2): 40–56.

Mayer, Richard E. 2002. “Multimedia learning”. Teoksessa *Psychology of learning and motivation,* 41:85–139. Elsevier.

———. 2008. “Applying the science of learning: evidence-based principles for the design of multimedia instruction.” *American psychologist* 63 (8): 760.

Mestre, Lori S. 2010. “Matching up learning styles with learning objects: What’s effective?” *Journal of Library Administration* 50 (7-8): 808–829.

Miller, Roy, ja Christopher T Collins. 2001. “Acceptance testing”. *Proc. XPUniverse* 238.

“MoviePy documentation”. 2022, heinäkuu. https://zulko.github.io/moviepy/.

Oud, Joanne. 2011. “Improving screencast accessibility for people with disabilities: guidelines and techniques”. *Internet Reference Services Quarterly* 16 (3): 129–144.

“Robot Framework User Guide”. 2022, heinäkuu. https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html.

“ScreenCapLibrary”. 2022, heinäkuu. https://rticau.github.io/ScreenCapLibrary/ScreenCapLibrary.html.

“SeleniumLibrary”. 2022, heinäkuu. https://robotframework.org/SeleniumLibrary/SeleniumLibrary.

Sommerville, Ian. 2001. “Software documentation”. *Software engineering* 2:143–154.

Sweller, John. 1988. “Cognitive load during problem solving: Effects on learning”. *Cognitive science* 12 (2): 257–285.

Toms, RK. 1982. “Human factors in user documentation”. *Information & Management* 5 (3): 149–155.

“WebDriver - Selenium”. 2022, heinäkuu. https://www.selenium.dev/documentation/webdriver/.

Woolfitt, Zac. 2015. “The effective use of video in higher education”. *Lectoraat Teaching, Learning and Technology Inholland University of Applied Sciences* 1 (1): 1–49.

Yu, Chong Ho, Angel Jannasch-Pennell, Samuel Digangi ja Charles Kaprolet. 2009. “An exploratory crossover study of learner perceptions of use of audio in multimedia-based tutorials”. *Journal of Educational Computing Research* 40 (1): 23–46.

Zelkowitz, Marvin V, ja Dolores R. Wallace. 1998. “Experimental models for validating technology”. *Computer* 31 (5): 23–31.

Liitteet

A Käytettyjen ohjelmistojen versionumerot

- `Python – 3.10.4`
- `selenium – 3.141.0`
- `robotframework – 5.0`
- `robotframework-seleniumlibrary – 5.1.3`
- `robotframework-screencaplibrary – 1.6.0`
- `moviepy – 1.0.3`
- `ImageMagick – 7.1.0-36 Q16-HDRI x64`
- `gizeh – 0.1.11`

B Screencast.resource ohjelmistolistaus

```
*** Settings ***
Documentation     Contains keywords for screencast recording and logging narration.
Library     NarrationRecorder     ${SUITE NAME}     WITH NAME     Narrator
Library   SeleniumLibrary     run_on_failure=Nothing
Library   ScreenCapLibrary

*** Keywords ***
Setup recording
    [Documentation]     Sets a proper pace for selenium, prepares the browser and begins
    ...                 recording narrations.
    [Arguments]     ${speed}=0.2 seconds     ${pause}=3
    Set Selenium Speed     ${speed}
    Maximize Browser Window
    Start Video Recording     alias=none     name=${SUITE NAME}_${TEST NAME}     fps=None
    ...                       size_percentage=1.0     embed=True     embed_width=100px
    ...                       monitor=1
    Narrator.initialize test narration     ${TEST NAME}
    Sleep     ${pause}
Stop recording
    [Documentation]     Stops recording and adds the narrations to '.json' file
    [Arguments]     ${pause}=4
    Sleep    ${pause}
    Narrator.append to json
    Stop Video Recording     alias=None
Add narration
    [Documentation]      Logs the narration message and also waits for a while.
    [Arguments]     ${message}     ${pause}=3
    narrator.add narration     ${message}
    Sleep     ${pause}
Highlight element
    [Documentation]     Records the locations of a given element
    ...                 Default sleep time is 0 seconds
    [Arguments]     ${locator}     ${pause}=0
    ${element}=     Get WebElement     ${locator}
    ${boundBox}=     Execute Javascript     return arguments[0].getBoundingClientRect();
    ...              ARGUMENTS     ${element}
    narrator.highlight element     ${boundBox}
    Sleep     ${pause}
```

C NarrationRecorder.py ohjelmistolistaus

```
import json
from datetime import datetime

class NarrationRecorder(object):

    def __init__(self, test_suite):
        self.__start_time = None
        self.__test_data = None
        self.__suite_name = test_suite
        self.__add_suite_if_missing()

    def initialize_test_narration(self, test_name):
        self.__test_data = {"name": test_name, "records": []}
        self.__start_time = datetime.now()

    def add_narration(self, message):
        self.__test_data["records"].append({"type": "narration", "time": datetime.now(), "message": message})

    def highlight_element(self, dom_rect):
        if dom_rect is None:
            raise Exception("No bounding box found")
        y_offset = 118  # Estimated menu bar height
        coordinates = {"left": dom_rect.get("left"), "top": dom_rect.get("top") + y_offset,
                       "right": dom_rect.get("right"), "bottom": dom_rect.get("bottom") + y_offset}
        self.__test_data["records"].append({"type": "element", "time": datetime.now(), "coordinates": coordinates})

    def append_to_json(self):
        self.__datetimes_to_elapsed_times()
        with open("NarrationRecord.json", "r") as file:
            data = json.load(file)
        with open("NarrationRecord.json", "w") as file:
            for item in data:
                if item.get("suite") == self.__suite_name:
                    item.get("tests").append(self.__test_data)
                    json.dump(data, file)
                    return

    def __datetimes_to_elapsed_times(self):
        records = self.__test_data.get("records")
        for r in records:
            timedelta = (r.get("time") - self.__start_time)
            r.update({"time": timedelta.total_seconds()})

    def __add_suite_if_missing(self):
        with open("NarrationRecord.json", "r") as file:
            data = json.load(file)
        try:
            for item in data:
                if item.get("suite") == self.__suite_name:
                    return
        except KeyError:
            pass
        with open("NarrationRecord.json", "w") as file:
            data.append({"suite": self.__suite_name, "tests": []})
            json.dump(data, file)
```

D Hello1.robot-testisarja

```
*** Settings ***
Documentation     Simple example test suite for screencast recording.
Resource      ../Resources/screencast.resource
Library     NarrationRecorder     ${SUITE NAME}     WITH NAME     Narrator
Library   SeleniumLibrary     run_on_failure=Nothing
Library   ScreenCapLibrary
Test Teardown     Close Browser

*** Variables ***
${BROWSER}            Chrome

*** Keywords ***

*** Test Cases ***
Google-haun tekeminen
    Open Browser     http://www.google.com     ${BROWSER}
    Setup recording

    Highlight element     id=L2AGLb
    Add narration     Klikataan ensin huomioikkuna pois.

    Click Button     id=L2AGLb
    Input Text     name=q     Hello world!
    Add narration     Kirjoita haluamasi hakusana ja paina hakupainiketta.

    Highlight element     name=btnK     2
    Click Button     name=btnK

    Stop recording

Bing-haun tekeminen
    Open Browser     https://www.bing.com/     ${BROWSER}
    Setup recording

    Input Text     id=sb_form_q     Hello bing!
    Add narration     Kirjoita haluamasi hakusana ja paina hakupainiketta.

    highlight element     id=search_icon     2
    Click Element     id=search_icon

    Stop recording
```

E Kansiorakenne opasvideon generoinnin jälkeen

```
Overlays
    rectangle_1.png
    rectangle_2.png
    rectangle_3.png
RecordedClips
    Hello1_Bing-haun_tekeminen_1.webm
    Hello1_Google-haun_tekeminen_1.webm
Resources
    Screencast.resource
Results
    HardCodedSubtitles.mp4
    RawVideo.mp4
    Timestamps.txt
Tests
    Hello1.robot
NarrationRecord.json
NarrationRecorder.py
TestDispatcher.py
Utilities.py
```