Lappeenrannan teknillinen yliopisto
LUT School of Engineering Science
Tietotekniikan koulutusohjelma

Kandidaatintyö

Teemu Hokkanen

Windows Phone 8 prototypointityökalun suunnittelu ja toteutus

Työn tarkastaja: Tutkijatohtori Ari Happonen

Työn ohjaaja: Tutkijatohtori Ari Happonen

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
LUT School of Engineering Science
Tietotekniikan koulutusohjelma

Teemu Hokkanen

Windows Phone 8 prototypointityökalun suunnittelu ja toteutus

Kandidaatintyö

2018

38 sivua, 7 kuvaa, 2 taulukkoa

Työn tarkastajat: Tutkijatohtori Ari Happonen

Hakusanat: prototypointi, mobiilisovellus, windows, windows phone 8
Keywords: prototyping, mobile application, windows, windows phone 8

Työn aiheena on prototypointityökalun luominen, jonka avulla on mahdollista tuottaa Windows Phone 8 -sovelluksen prototyyppejä. Tavoitteena on suunnitella ja toteuttaa prototypointisovellus, jonka tuottamissa prototyypeissä hyödynnetään puhelimista löytyviä sensoreita. Tässä työssä keskitytään kameran kautta tehtävään 2D -viivakoodin lukemiseen sekä NFC -tunnisteiden lukemiseen. Toteutuksen tuloksena on sovellus, jolla voidaan tehdä mobiilisovellusprototyyppi. Windows Phone ei ole työn julkaisemishetkenä enää ajankohtainen teknologia, mutta sovellus luo myös XML -muotoisia tiedostoja prototyypin rakenteen mukaan. Näitä tiedostoja hyödyntäen on mahdollista tehdä esimerkiksi mobiilisovellus Android- tai iOS -alustalle, joka lukee kyseisiä tiedostoja ja toteuttaa halutun prototyypin toiminnot.

ABSTRACT

Lappeenranta University of Technology
LUT School of Engineering Science
Degree Program in Computer Science

Teemu Hokkanen

Design and implementation of Windows Phone 8 prototyping tool

Bachelor's Thesis

38 pages, 7 figures, 2 tables

Examiner: D.Sc. (Tech.) Ari Happonen

Keywords: prototyping, mobile application, windows, windows phone 8

Subject of this thesis is to create a prototyping tool, which produces Windows Phone 8 -application prototypes. The goal of this project is to design and implement a prototyping application that produces mobile application prototypes which can interact with 2D -barcodes and NFC -tags. The result of this project is an application that you can use to create a prototype of a mobile application. Windows Phone is not a valid technology when this thesis is published, but the created tool can also produce the schematics of the prototype as an XML -file. This file could be used for example with a new mobile software with Android or iOS, that could read the XML file and simulate the desired prototype.

ALKUSANAT

Tämä työ aloitettiin Lappeenrannassa vuonna 2014 ja se valmistui lokakuussa 2018. Alkuperäisen aikataulun mukaisesti ohjelman toteutus oli tarkoitus suorittaa kesän aikana siten, että se olisi valmis alkusyksystä 2014. Toteutus kuitenkin venyi pitkälle vuoden loppuun ja sen myötä myös loppuraportin kirjoittaminen tapahtui lopullisesti vuonna 2018. Haluan kiittää rakasta avopuolisoani tuesta sekä ohjaajaani Ari Happosta positiivisesta ja kannustavasta tavasta ohjata, jonka ansiosta en koskaan epäröinyt ottaa tarvittaessa yhteyttä, vaikka työ ei edennytkään suunnitellusti.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

HTML	Hypertext Markup Language
NFC	Near Field Communication
PNG	Portable Network Graphics
PDF	Portable Document Format
WPF	Windows Presentation Foundation
2D	Two-dimensional

1 JOHDANTO

1.1 Tausta ja työn tarkoitus

Työn lähtökohtana on prototypointisovelluksen tekeminen. Työn varsinainen aihe on ollut tarjolla tietotekniikan laboratoriossa kandidaatintyön kurssin sivuilla. Valmiin aiheen ajatuksena on, että ohjelmointitaidoton henkilö pystyy tekemään kuvia hyödyntäen prototyypin, joka toimii Windows Phone 8 -puhelimella. Prototyypissä hyödynnetään myös mahdollisuutta käyttää puhelimen Near Field Communication (NFC) -toimintoa ja kaksiulotteisten (2D) -viivakoodien lukemista kameralla. Tällä tavalla henkilö voi ilman ohjelmointitaitoja luoda tehokkaasti omien kuviensa avulla puhelimen sensoreita hyödyntävän prototyypin, jolla henkilö pystyy esittelemään oman ideansa sovelluksesta konkreettisesti puhelimella.

Prototyyppi on rajoitettu esitys tai toteutus sovelluksesta (Preece, Rogers & Sharp 2002, 240–241.). Prototyyppejä käytetään eri tavalla eri aloilla. Esimerkiksi tietotekniikan alalla prototypointia voidaan käyttää tuotteen eri osien tutkimiseen tai ideoiden esittämiseen. (Beaudouin-Lafon & Mackay 2002, 1-2.) Tässä työssä keskitytään työkaluun, jonka avulla käyttäjä voi luoda prototyypin sovelluksesta ilman ohjelmointia.

Toteutettava sovellus erottuu muista samankaltaisista sovelluksista mahdollisuudella luoda prototyyppi, joka käyttää puhelimen omia sensoreita. Älypuhelimet ovat yleistyneet viime vuosien aikana ja esimerkiksi vuonna 2013 Suomessa 60%:sta kotitalouksista löytyi älypuhelin. (Tilastokeskus 2013). Vuoden 2007 iPhonen julkaisun jälkeen puhelimissa alettiin käyttämään enemmän sensoreita (Yan & Chakraborty 2014, 2-3.). Windows Phone 8 –alustaa varten tehdyt prototypointityökalut, kuten WireframeSketcher ja NinjaMock, eivät kuitenkaan tarjoa mahdollisuutta hyödyntää puhelimen sensoreita.

1.2 Tavoitteet ja rajaukset

Tämän työn tavoitteena on suunnitella ja toteuttaa toimiva versio koodigeneraattorista Windows 8 -alustalle. Generaattori luo prototyyppejä Windows Phone 8 -laitteelle, josta

niitä ajetaan. Generaattorin lisäksi älypuhelimen päähän luodaan myös sovellus, joka osaa ajaa kyseisiä prototyyppejä. Pääohjelma, puhelimen ohjelma ja generoitu koodi tehdään C# -kielellä. C# on yleinen kieli Windows Phone 8 ja Windows 8 sovelluksissa (Microsoft 2014).

Pääsovelluksella luotavat prototyypit ovat resoluutiota 480x800. Pääsovelluksella prototyyppiin on mahdollista lisätä navigoimista varten nappeja, kuvia sivujen taustalle, lisäsivuja sekä erilaiset NFC- ja 2D sivut. NFC-sivulla on mahdollista havaita muita luettavia NFC-tarroja. 2D-viivakoodisivulla on mahdollista lukea kameran avulla viivakoodeja, ja nähdä mitä niissä lukee. Pääsovelluksella on mahdollista avata ja tallentaa projekteja sekä luoda prototyyppi, joka on mahdollista avata puhelimesta löytyvän sovelluksen avulla.

1.3 Työn toteutus

Prosessi aloitettiin tutkimalla toteuttamiseen tarvittavia teknologioita. Verkosta etsittiin sopiva määrä tietoa erilaisilta tahoilta ja sen perusteella tehtiin tarvittavat päätökset. Tiedonlähteinä käytettiin esimerkiksi Microsoftin omaa dokumentaatiota sekä useampien eri sivustojen artikkeleita Windows Phone -sovelluskehityksestä. Tarkoituksena oli saada nopeasti yleinen käsitys siitä, mitä teknologioita yleensä käytetään.

Ennen toteutusta opiskeltiin työn toteutukseen käytettävää teknologiaa verkosta löytyvien materiaalien perusteella. Perehtymisen jälkeen toteutusvaihe aloitettiin harjoitusversion toteutuksella. Harjoitusversion jälkeen tehtiin tarvittavat suunnitelmat, jonka jälkeen siirryttiin itse toteutukseen. Lopuksi toteutuksen jälkeen työprosessista kirjoitettiin loppuraportti, jossa käydään läpi ohjelmistoprototypointia eri näkökulmista.

1.4 Työn rakenne

Toisessa luvussa käydään läpi prototypointiin liittyvää kirjallisuutta. Alaluvussa 2.1 käydään läpi prototyypin määritelmä ja sen tarkoitus. Seuraavissa alaluvuissa 2.2 ja 2.3 esitellään prototypoinnin historiaa sekä käydään läpi prototypoinnille ominaisia erilaisia tarkkuuksia. Seuraavaksi alaluvuissa 2.4 ja 2.5 käydään läpi erilaiset prototypointistrategiat

sekä prototyyppien elinkaareen ja toteutuksen nopeuteen liittyvät asiat. Tämän jälkeen esitellään erilaisia olemassa olevia prototypointikeinoja tietotekniselle alalle rajattuna. Erilaisten keinojen jälkeen esitellään, minkälaisia keinoja nykyään käytetään. Alaluvussa 2.7 käydään läpi suosituimpia sovelluksia mobiilialustoilla ja lopuksi esitellään johtopäätökset kirjallisuuskatsauksessa esitellystä materiaalista.

Kolmannessa luvussa perehdytään sovelluksen suunnitteluun sisältyviin vaiheisiin. Ensimmäisessä alaluvussa esitellään ja perustellaan suunnitteluun käytetyt keinot. Toisessa alaluvussa käydään läpi ohjelman toteutusta varten tehty vaatimusmäärittely. Lopuksi käydään läpi arkkitehtuurisuunnitelma.
Neljännessä luvussa käydään läpi sovelluksen toteutus ja siihen liittyvät aiheet. Ensimmäisessä alaluvussa esitellään sovellukselle asetetut tavoitteet. Tämän jälkeen tarkastellaan työn toteutusprosessia aikajanan avulla sekä käydään läpi toteutuksen eri kokonaisuudet. Alaluvussa 4.3 käydään läpi kaikki, mitä lopullisen ohjelmaan on tehty.

Seuraavaksi alaluvussa 4.4 analysoidaan toteutunutta sovellusta suhteessa asetettuihin tavoitteisiin. Tämän jälkeen listataan ja kuvaillaan työhön liittyneet haasteet. Aliluvussa 4.6 tehdään johtopäätökset sovelluksen toteutuksesta, jonka jälkeen viimeisessä aliluvussa pohditaan mahdollisia tulevaisuudessa tehtäviä parannuksia sovellukseen. Viidennessä luvussa pohditaan tulevaisuutta yleisemmällä tasolla. Lopuksi tehdään yhteenveto ja kerrataan lyhyesti raportin oleellisimmat osuudet.

2 KIRJALLISUUSKATSAUS

Kirjallisuuskatsauksessa käydään läpi yleisellä tasolla ohjelmistoprototypoinnin määritelmä sekä siihen liittyvää termistöä. Lisäksi kappaleessa esitellään useampi olemassa oleva prototypointisovellus Windows Phone -alustalle. Lopuksi kirjallisuuskatsauksen teoriaa sekä havaintoja esitellyistä sovelluksista peilataan tämän työn toteutukseen.

2.1 Prototyypin määritelmä ja tehtävä

Ohjelmistoalalla prototyyppi on toteutus, joka jollain tasolla vastaa jotain ohjelmistoon toteutettavista osa-alueista. Käyttötarkoituksesta riippuen prototyypit voivat olla toteutustasoltaan lähellä tai kaukana tavoiteltua lopputulosta. Niiden avulla on mahdollista tutkia suunnitelmaa ohjelmasta ja sitä kautta löytää mahdollisia ongelmakohtia ja parempia toteutustapoja. Tämä on mahdollista saavuttaa prototypoinnin avulla pienemmällä ajan käytöllä kuin mitä kokonaisen ohjelman tekeminen vaatisi. Prototypointi on usein merkittävässä roolissa ohjelmistojen kehittämisessä. (Arent, Arnowitz & Nevin 2007, 41-42, 48-49.)

2.2 Ohjelmistoprototypoinnin historia

Historiallisesti muilla aloilla prototyyppien käyttäminen ei ole uusi asia. Leonardo Da Vinci jätti jälkeensä suuren määrän paperille piirtämiään prototyyppejä jo 1400-luvulla. 1800- ja 1900-luvuilla elänyt Thomas Edison hyödynsi prototypointia ja sen avulla pystyi valmistamaan uusia tuotteita uskomattomalla nopeudella. Edisonin käyttämiä lähestymistapoja prototypointiin hyödynnetään myös ohjelmistoalalla. (Arent, Arnowitz & Nevin 2007, 45-47.) Ohjelmistoalalla termiä prototypointi alettiin käyttää 1980-luvun alussa ohjelmistokehittäjien keskuudessa. Silloin uskottiin, että prototyyppejä tulisi käyttää dokumenttina, joka helpottaa sovelluskehittäjien ja käyttäjien välistä toimintaa. Kyseisenä aikakautena käsiteltiin suuria ohjelmiston tuottamiseen liittyviä ongelmia, jotka aiheuttivat paljon kustannuksia. (Budde & Zullighoven 1990.)

Budden & Zullighovenin mukaan ohjelmien tekeminen maksoi liian paljon ja ne olivat laadultaan heikkoja. Ongelmien ratkaisujen pohtimisen yhteydessä keksittiin, että

prototypoinnista voi olla apua. Aiheesta järjestettiin tuolloin konferenssi, jossa mietittiin kysymyksiä ”Mitä tarkoitetaan termillä prototypointi” ja ”Voiko prototypointia todella tehdä”. Todisteet prototypoinnin hyödyllisyydestä tulivat myöhemmin ja 90-luvulla prototypointi oli yleisesti yritysten käytössä järjestelmien kehittämisessä. (Budde & Zullighoven 1990.)

2.3 Ohjelmistoprototypoinnin tarkkuus

Prototypoinnin tarkkuudella tarkoitetaan sitä, kuinka tarkasti prototyyppi vastaa ulkonäöltään ja toiminnaltaan toteutettavan ohjelman käyttöliittymää (Arent, Arnowitz & Nevin 2007, 125). Kirjassa Interaction Design – beyond human-computer interaction Preece et al. esittelevät kaksi prototypoinnin tarkkuuden tasoa: korkean ja matalan tarkkuuden prototypointi. Prototyyppejä luodessa täytyy valita kulloinkin sopiva tarkkuuden taso (Beaedouin-Lafon & Mackay, 2002, 4). Korkeamman tarkkuuden prototyypin toteuttamiseen kuluu enemmän aikaa, joten oikean valinnan tekeminen on tärkeää (Arent, Arnowitz & Nevin 2007, 122).

Korkean tarkkuuden prototypoinnissa prototyyppi näyttää samalta, mitä voi odottaa lopullisessa sovelluksessa. Korkea tarkkuus liittyy myös siihen, että ohjelma toimii ja käyttäytyy lopullisen ohjelman tavoin. Tätä hyödyntämällä on mahdollista saada esimerkiksi käyttäjiltä palautetta ohjelmasta ennen sen valmistumista (Arent, Arnowitz & Nevin 2007, 125). Korkean tarkkuuden prototyyppien luomiseen tarvitsee apusovelluksen. (Preece, Rogers & Sharp, 2007, 535) Tällaisia prototyyppejä käytetään esimerkiksi testaamiseen, jolloin prototyypistä saadaan enemmän hyötyä kuin vastaavasta matalan tarkkuuden tuotoksesta (Arent, Arnowitz & Nevin 2007, 129).

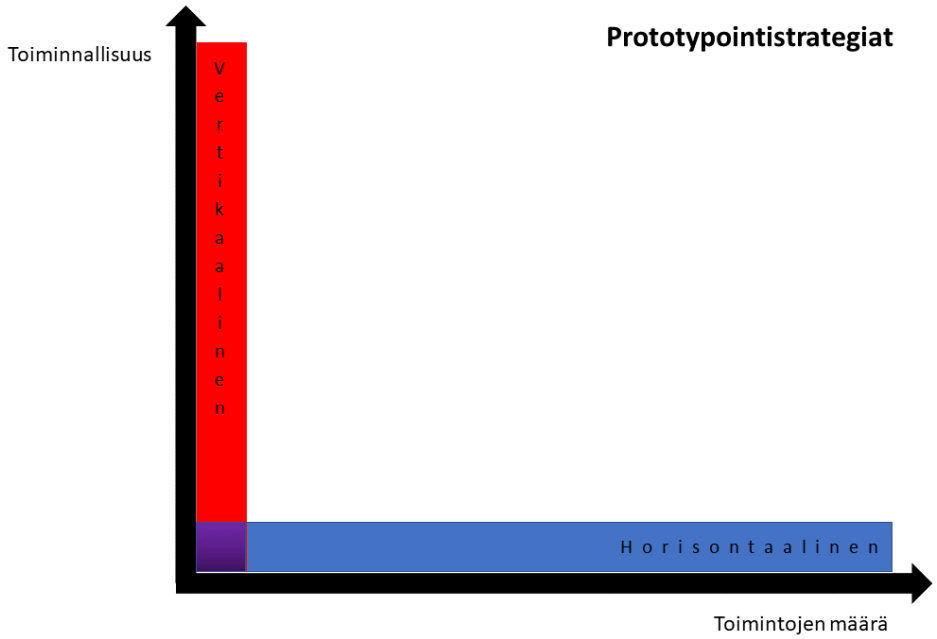
Matalan tarkkuuden prototyypi ei näytä lopulliselta tuotteelta. Ne ovat yksinkertaisia, halpoja ja nopeita tehdä. Matalan tarkkuuden prototyyppejä on myös yksinkertaista muokata uusien ideoiden mukaisesti. Kyseinen ominaisuus on tärkeää suunnittelun alkuvaiheessa. Matalan tarkkuuden prototyyppejä ei ole koskaan tarkoitus ottaa osaksi lopullista järjestelmää. (Preece, Rogers & Sharp, 2007, 531)

2.4 Prototypointistrategia

Teoksessa Usability Engineering Nielsen (1993, 94-95) jakaa prototyypit kahteen ulottuvuuteen: vertikaalinen prototypointi ja horisontaalinen prototypointi. Ensin mainitussa ulottuvuudessa vähennetään toimintoja siten, että jäljellä on muutama valittu funktio. Nämä toiminnot on toteutettu lähes lopulliseen mittakaavaan asti, jotta niitä voidaan testata todellisissa olosuhteissa. (Nielsen 1993, 94-95.) Esimerkiksi käyttöliittymän prototypoinnissa tämä tarkoittaa sitä, että luodaan toiminnot käyttäjän yhdestä valinnasta siten, että se toimii lopullisen vaatimuksen tavoin asetettujen rajoitteiden puitteissa. (Beaudouin-Lafon & Mackay 2002, 12-14.) Vertikaalisia prototyyppejä ei yleensä säilytetä, koska ne on tehty aikaisessa vaiheessa projektia ennen suurta osaa suunnittelua ja ne keskittyvät vain yhteen suunnittelun kysymykseen (Beaudouin-Lafon & Mackay 2002, 4.)

Horisontaalisessa prototypoinnissa taas luodaan kaikki ohjelman osiot, mutta niihin ei tehdä toimintoja. (Nielsen 1993, 94-95.) Käyttöliittymästä tehdystä prototyypistä voidaan esimerkiksi saada hyvä kuva kokonaisuudesta. Tällä tavoin sovellusta voidaan prototypoida tehokkaasti käyttämällä erilaisia sovelluksia. Horisontaaliset prototyypit päätyvät usein lopulliseen sovellukseen. (Beaudouin-Lafon & Mackay 2002, 12-14.)

Teoksessa Prototype Development and Tools (2002, 12-14) Beaudouin-Lafon ja Mackay kutsuvat horisontaalista ja vertikaalista prototypointia prototypointistrategioiksi. Niiden lisäksi teoksessa esitellään tehtäväsuuntautunut prototypointi sekä skenaarioon pohjautuva prototypointi. Ensimmäiseksi mainittu strategia tarkoittaa sitä, että prototyyppi luodaan toteuttamaan tietyt sille määritellyt tehtävät. Jälkimmäinen strategia on samanlainen tehtäväsuuntautuneen prototypoinnin kanssa, mutta siinä prototyyppi luodaan noudattamaan aitoa skenaariota. Prototyyppien luonne on kuvattuna kuvassa 1. (Beaudouin-Lafon & Mackay 2002, 12-14.)



Kuva 1.

2.5 Ohjelmistoprototypoinnin nopeus ja prototyyppien elinkaari

Yksi tarkasteltava näkökulma prototypointiin liittyen on aika, joka prototyypin luomiseen kuluu. Kuten aiemmin prototypoinnin tarkkuuksia käsitellessä todettiin, prototyyppiin kuluu enemmän aikaa sen mukaisesti, kuinka tarkasti se tehdään. Pikaprototypoinnissa pyritään tekemään prototyyppi nopeasti tarkkuuden kustannuksella, kun taas huolellisessa prototypoinnissa käytetään enemmän aikaa tarkemman lopputuloksen saamiseksi. Molempia tapoja käytetään vaihtelevasti eri vaiheessa ohjelman kehittämistä. (Arent, Arnowitz & Nevin 2007, 157-158.)

Prototyypeillä on myös eri tarkoituksiin eri elinkaaria. Pikaprototypoinnissa prototyyppi luodaan lyhyessä ajassa eikä sitä käytetä enää testaamisen jälkeen. Tätä tapaa hyödynnetään yleensä suunnittelun alkuvaiheessa. Tällä tavalla prototyypin tekemiseen kuluvaa aikaa ja rahaa säästetään. Pikaprototypointia voidaan tehdä joko tietokoneen avulla tai ilman. Iteratiiviset prototyypit kehitetään vastaamaan suunnittelua ja niitä on tarkoitus kehittää suunnittelun iteraatioissa. Evoluutioprototyypit ovat erikoistapauksia iteratiivisista prototyypeista, joka tarkoittaa, että ne kehitetään lopulta osaksi tuotetta. Tämä tapa vaatii enemmän suunnittelua kuin pikaprototyypit ja iteratiiviset prototyypit.

(Beaudouin-Lafon & Mackay 2002, 5-6.) Elinkaari voi olla yksi tekijä, jonka mukaan päätetään prototypointiin käytettävä aika (Arent, Arnowitz & Nevin 2007, 159).

2.6 Prototypointikeinoja

Kynä ja paperi -prototypointi on nopea keino prototypoida esimerkiksi interaktiivista järjestelmää (Beaudouin-Lafon & Mackay 2002, 14.) tai käyttöliittymää (Szekely 1995, 4.). Kyseinen menetelmä on helppokäyttöinen ja sen avulla voidaan ilmaista monenlaista tietoa. Kynä ja paperi –prototypoinnin käyttäminen rohkaisee myös tiimityöhön, erityisesti jos prototyyppiä luodaan suuremmalle alustalle. Kynä ja paperi –prototypoinnin heikkouksia ovat käyttäytymisen kuvaaminen ja ohjelmien ajamisen mahdollisuuden puuttuminen. (Szekely 1995, 4.) Niitä on mahdollista simuloida vaihtamalla kuvia tai ohjaamalla esitystä äänellä (Arent, Arnowitz & Nevin 2007, 179).

”Card sorting” on prototypointikeino, jonka avulla selvitetään esimerkiksi parasta tiedon säilytys paikkaa tai navigointirakennetta. Kyseistä keinoa käytetään suunnittelun alkuvaiheessa ja siihen osallistuu tulevaa sovellusta käyttäviä henkilöitä. Prototypointi tapahtuu siten, että tapahtuman vetäjä antaa käyttäjälle lappuja, joissa on esimerkiksi internet sivuston sisältöön liittyviä termejä. Käyttäjä järjestää laput omasta mielestään järkeviin kokonaisuuksiin. Lopuksi ohjelman suunnittelijat analysoivat tuloksista sitä, miten käyttäjät hahmottavat heille esitetyn informaation ja käsitteet. (Arent, Arnowitz & Nevin 2007, 176.)

Videoprototypoinnissa käytetään videota kuvaamaan sovelluksen käyttämistä. Tarkoituksena on parantaa suunnitelmaa uusien ideoiden keksimisen sijaan. (Beaedouin-Lafon & Mackay, 2002, 17.) Videoprototypointi voi olla sisällöltään esitys ohjelman toiminnasta. Keinoa käytetään yleensä aikaisessa vaiheessa ohjelman kehittämisessä. (Arent, Arnowitz & Nevin 2007, 183-184.) Wizard of Oz –prototypoinnissa simuloidaan sitä, että käyttäjä käyttää ohjelmaa. Toinen henkilö valvoo ja ohjaa käyttäjän toimintaa luoden illuusion tietokoneohjelmasta. (Beaedouin-Lafon & Mackay, 2002, 16.) Tätä tapaa voidaan suorittaa esimerkiksi ääniohjauksella tai fyysisellä toteutuksella (Arent, Arnowitz & Nevin 2007, 184.)

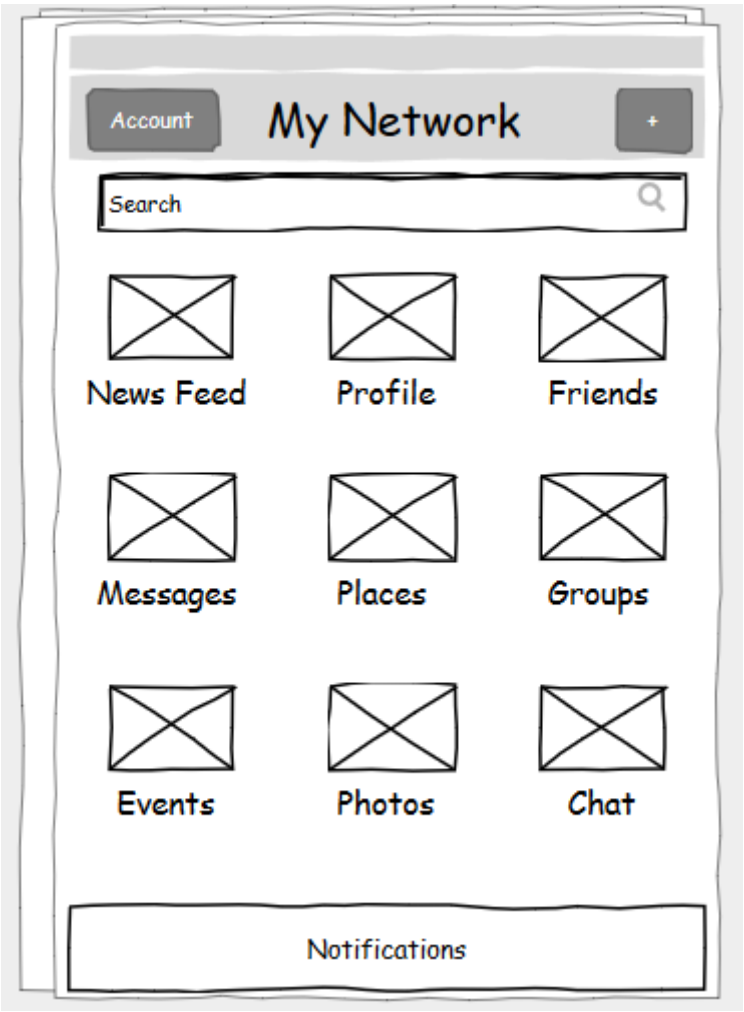
Ohjelmoiva prototypointi tarkoittaa sitä, että luodaan esimerkiksi JavaScript -ohjelmointikielellä prototyyppi. Tällaisia prototyyppejä on usein tarkoitus hyödyntää myöhemmissä vaiheessa ohjelman kehittämistä joko testeissä tai osana lopullista ohjelmaa. Ohjelmoivaan prototypointiin kuluu aikaa, joten on hyvä tietää ohjelman rakenne pääpiirteittäin toteutuksella (Arent, Arnowitz & Nevin 2007, 185.).

2.7 Olemassa olevia sovelluksia mobiilialustalle

Prototypointia varten löytyy paljon erilaisia mobiilisovelluksia, kuten esimeriksi lähes aidon näköisiä prototyyppejä tuottava Justinmind Prototyper tai Picodo, jonka avulla saadaan matalan tarkkuuden sovelluksia, jotka näyttävät käsin piirretyiltä. Toteutettava sovellus eroaa kuitenkin muista vastaavista sovelluksista NFC ja 2D-viivakoodin hyödyntämisessä. Windows Phone 8 -alustalle ei löytynyt montaa sovellusta, joiden avulla prototyyppejä olisi mahdollista ajaa puhelimella.

2.7.1 Pidoco

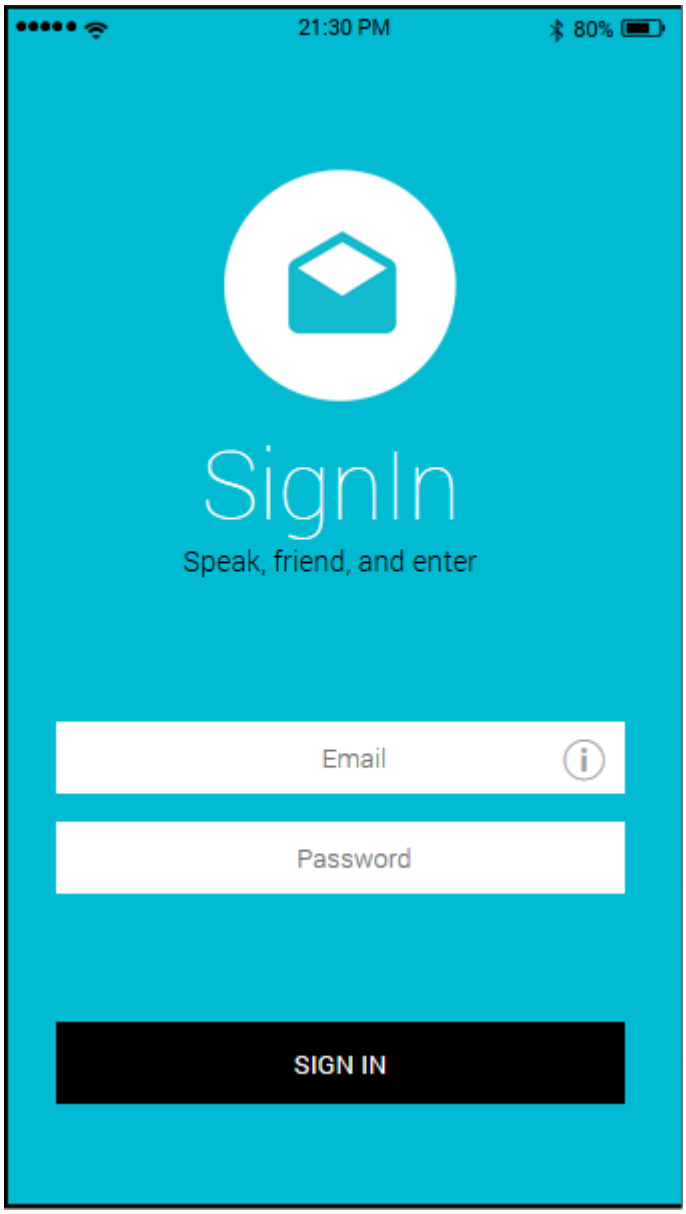
Selainpohjainen ohjelma tarjoaa mahdollisuuden luoda matalan tarkkuuden prototyyppejä tietokoneen lisäksi eri mobiilialustoille. Älypuhelimilla ohjelma on saatavilla puhelimien sovelluskaupoista. Windows Phone 8 -alustalle ohjelmaa ei kuitenkaan saa. Sovellusta on mahdollista ajaa selaimessa tai puhelimessa. Sovelluksessa prototyyppi luodaan raahaamalla ja pudottamalla piirretyn näköisiä komponentteja ja esimerkki lopputuloksesta löytyy kuvasta 2. Prototyypin saa testikäyttöön esimerkiksi PDF tai HTML -muodossa. (Pidoco 2014.)



Kuva 2.

2.7.2 Justinmind Prototyper

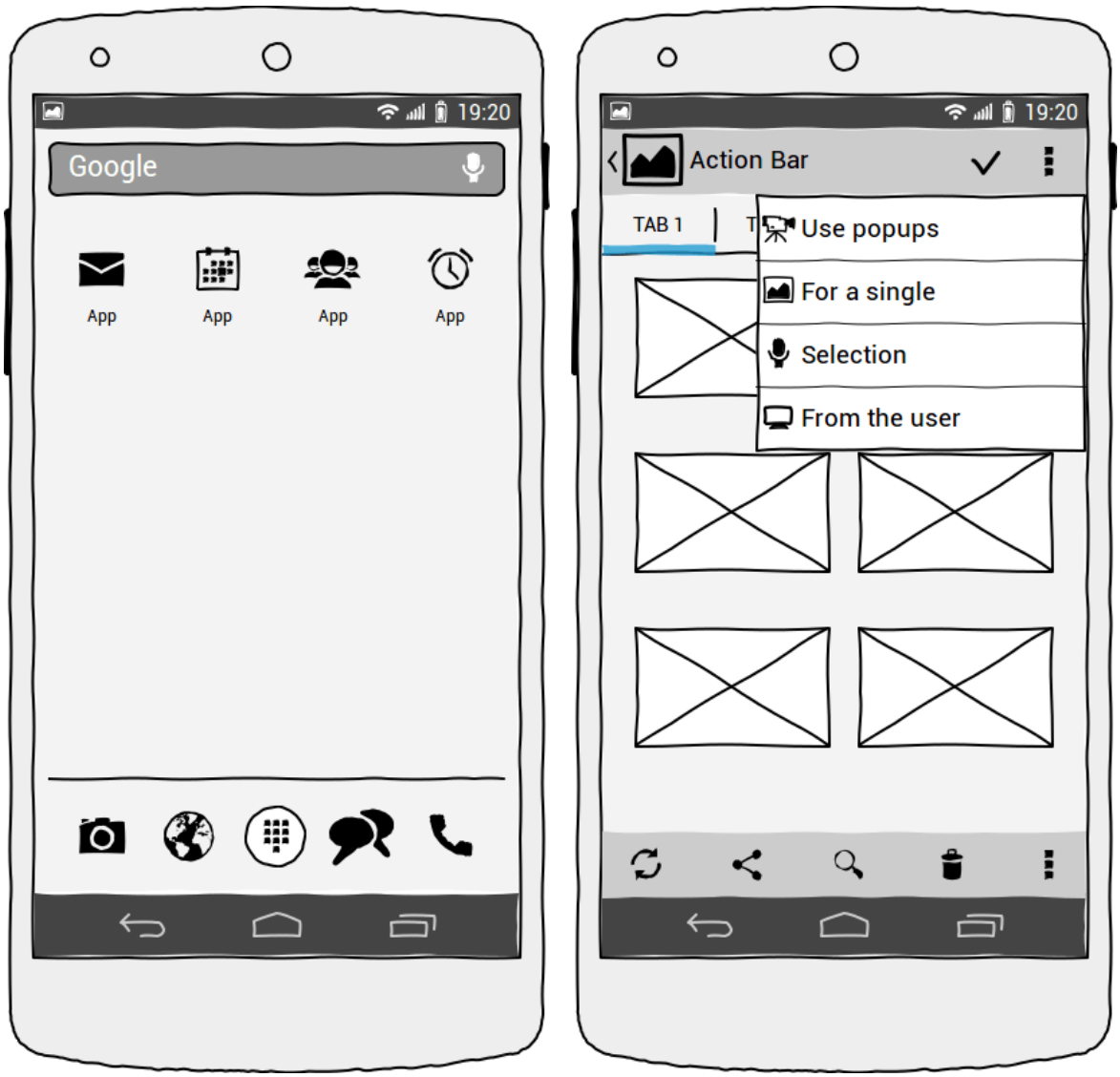
Työpöydällä ajettava sovellus tarjoaa monipuoliset vaihtoehdot erilaiseen prototypointiin ”raahaa ja pudota” -tyyppisellä ratkaisulla. Valmiista kuvista voi valita piirretyn näköiset vaihtoehdot tai esimerkiksi Android käyttöjärjestelmän mukaiset osat, kuten kuvasta 3 voi havaita. Prototyper tukee myös vain Android- ja iOS –käyttöjärjestelmiä, mutta ohjelmalla luotaviin prototyyppeihin on saatavilla Windows Phone -toiminnallisuuksia. Älypuhelimille prototyyppejä on mahdollista saada vain HTML/JavaScript muodossa. Prototyyppiin on mahdollista lisätä erilaisia tapahtumia, joiden avulla voidaan simuloida toimintoja. Esimerkiksi nappiin voi lisätä tapahtuman, jonka mukaan nappia painettaessa jokin toinen osio näkymästä häviää. (Justinmind 2014.)



Kuva 3.

2.7.3 WireframeSketcher

WireframeSketcher on työkalu, jonka avulla voidaan tehdä prototyyppejä Windows Phone –alustalle. Prototyyppi luodaan työpöytäsovelluksen avulla, josta valmiit prototyypit saadaan joko PDF- tai HTML-muodossa, mitä on mahdollista esitellä Windows Phone 8 -laitteella. WireframeSketcherissä on omat valmiit työkalut matalan tarkkuuden prototyyppien luomiseen, mutta heidän sivustolta löytyy myös yhteisön tekemiä kuvia, joita hyödyntäen voi tehdä lähes aidon näköisiä sovelluksia. Kuvassa 4 on esimerkki tehdystä prototyypistä.



Kuva 4.

2.7.4 NinjaMock

NinjaMock –prototypointisovelluksella tehdyt prototyypit näyttävät aina käsin piirretyiltä. Prototyypit luodaan websovelluksella ja ne on mahdollista luoda PDF, PNG tai HTML –muotoon. Valmiita projekteja on mahdollista ajaa puhelimella avaamalla websovelluksesta saatavan QR-koodin linkin. Prototyyppien luomisessa on mahdollista käyttää valmiita haluamansa alustan työkaluja ja kuvia. Prototyyppiin on mahdollista myös piirtää kuvia, joita voi käyttää esimerkiksi nappeina. Esimerkkinä kuvassa 5. on käytetty Windows Phone 8 -tyylisiä näppäimiä prototyypissä.

Kuva 5.

2.8 Johtopäätökset kirjallisuus- ja työkalukatsauksesta

Prototypointi on yleisesti hyödylliseksi todettu toimintamalli. Siinä tulee huomioida toteutettavan ohjelman tarpeet ja suunnitella mahdollinen prototypointi sen mukaisesti. Kirjallisuuskatsauksessa esitellyt näkökulmat näkyvät myös aiemmassa kappaleessa esitellyissä mobiilisovelluksissa, esimerkiksi sovelluksien käyttöliittymien tarkkuuksissa. Tämän työn toteutettavana oleva sovellus erottui toteuttamishetkellä muista sovelluksista siten, että kyseisellä sovelluksella on mahdollista prototypoida tarkemmalla tasolla mobiilisovelluksen toiminnallisuutta sensoreiden avulla. Ohjelma tarjoaa käyttäjälle muihin sovelluksiin verraten monipuolisen mahdollisuuden tehdä prototyyppeja Windows Phone 8 -alustalle.

3 SOVELLUKSEN SUUNNITTELU

3.1 Suunnittelu

Tämän kandidaatintyön suunnitteluvaiheessa huomioitiin ohjelmoijan kokemattomuus käytettävästä ohjelmointikielestä. Suunnittelu alkoi tästä syystä alkuraportin ohessa ohjelmointikielen opiskelemisella ja vaatimusmäärittelyn tekemisellä. Sovelluksesta pyrittiin tekemään yksinkertainen, jotta toteutukseen kuluva aika pysyy järkevässä mittakaavassa, joten ohjelmointityön helpottamiseksi alkuraportin jälkeen täytyi löytää avuksi kirjastoja toteutukseltaan haastaville toiminnoille, kuten NFC:n käyttöön ja 2D-viivakoodien lukemiseen.

Ohjelmointikielen opiskelun ja vaatimusmäärittelyn jälkeen toteutettiin esimerkkiprototyyppi, joka on Windows Phone 8 -sovellus. Kyseiseen prototyyppiin haluttiin sisällyttää kaikki mahdolliset toiminnot, jotta sitä voidaan käyttää pohjana generoitavalle koodille. Sen avulla oli mahdollista määrittää tarkemmin, mitä sovelluksella tulee pystyä tekemään ja minkälaisen tiedoston se luo. Tämän jälkeen toteutettiin testisovellus, jonka avulla harjoiteltiin tarvittavien toimintojen toteuttamista. Testisovelluksen jälkeen suunnittelu jatkui arkkitehtuurisuunnitelman tekemisellä, jossa hyödynnettiin testisovelluksesta opittuja asioita ja valmiita toimintoja. Esimerkkiprototyypin ja testisovelluksen avulla selvitettiin myös mahdollisia rajoitteita, joita ohjelmointikielestä tai alustasta johtuen saattaisi tulla vastaan.

3.2 Vaatimusmäärittely

Suunnittelun ensimmäinen vaihe oli kerätä listalle kaikki mahdolliset sovellukseen liittyvät vaatimukset. Tämän rajoitetun vaatimusmäärittelyn tarkoitus oli luoda mahdollisimman kattava kuvaus sovelluksesta, jotta kokonaisuutta on helpompi hahmottaa. Vaatimusmäärittelyä oli tarkoitus päivittää testisovelluksen toteuttamisen jälkeen. Vaatimusmäärittely sisältää ainoastaan pääsovelluksen määritykset, koska tässä vaiheessa sovelluksen toteuttamistapa oli vielä epäselvä.

3.2.1 Rajoitteet

Sovellukseen liittyvät rajoitukset koskivat alustoja, joissa ohjelmia on mahdollista käyttää. Prototyyppiin lisättävän kuvan koko lukittiin, jotta toteutukseen ei tarvitse ottaa tässä vaiheessa eri kokoisia laitteita huomioon.

Taulukko 1. Rajoitteet

ID	Rajoite	Rajoitteen kuvaus
1	Windows	Sovellus tehdään Windows -käyttöjärjestelmälle
2	Prototyypit WP8	Sovellus luo prototyypejä Windows Phone 8 -puhelimille
3	Kuvan koko	Lisättävien kuvien koko täytyy olla 480x800

3.2.2 Vaatimukset

Vaatimukset kohdistuivat myös pääsovellukseen. Vaatimuksiin sisältyi tässä vaiheessa oleellisimmat osat toteutuksesta. Vaatimuksia oli tarkoitus päivittää sitä mukaan, kun niitä nousee toteutuksessa esille.

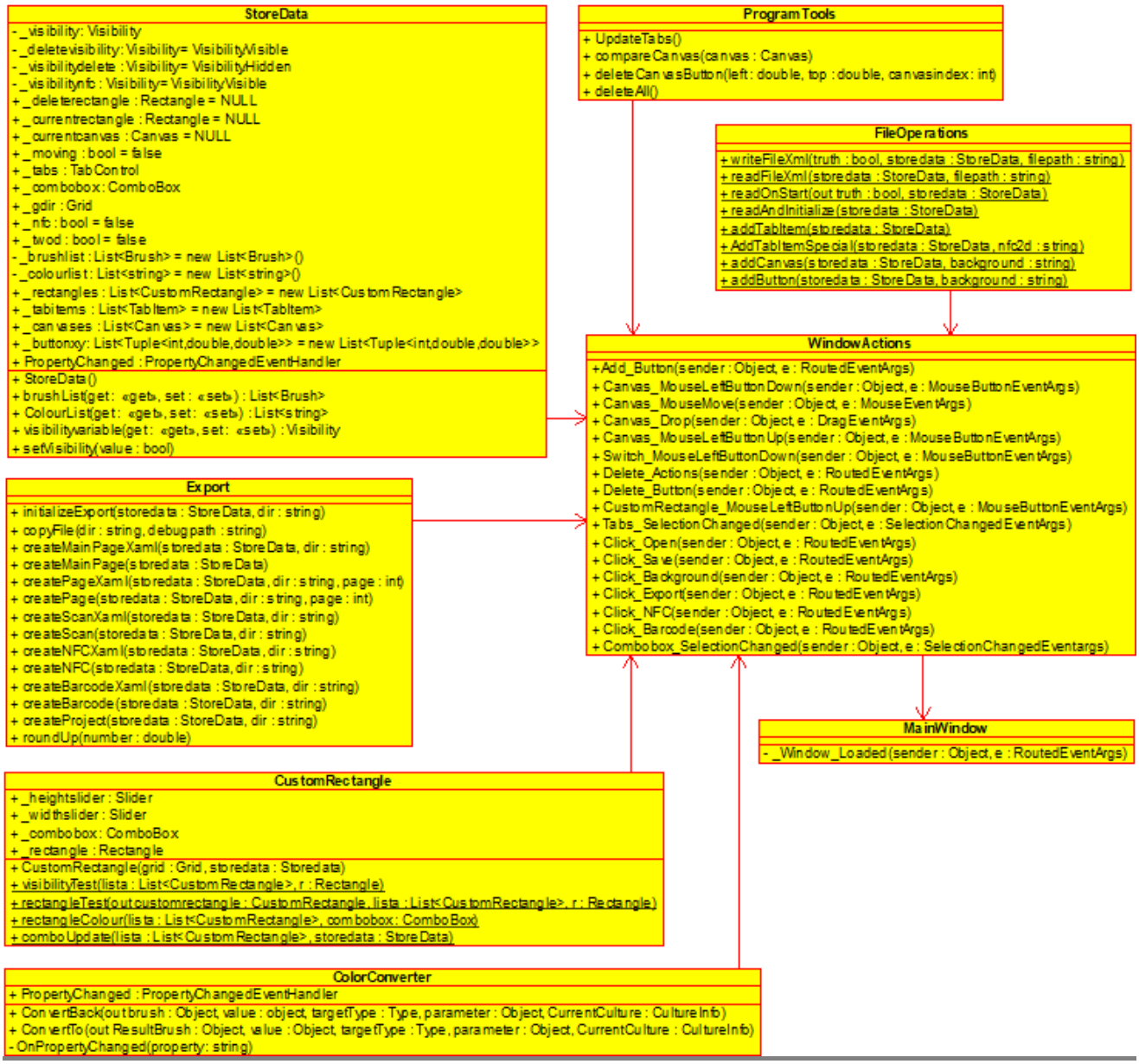
Taulukko 2. Vaatimukset

ID	Vaatimus	Vaatimuksen kuvaus
1	Projektin tallennus	Ohjelmalla on mahdollista tallentaa keskeneräinen projekti
2	Projektin lataaminen	Ohjelmalla on mahdollista avata tallennettu projekti
3	Työntekoalusta ohjelmassa	Kanvaasi ohjelman käyttöliittymässä, jolle käyttäjä luo haluamansa prototyypin
4	Kuvien lisääminen	Prototyyppiin pitää pystyä lisäämään kuvia eri sivuille. Ohjelma ”tehdään” kuvilla.
5	Näppäimien lisääminen	Prototyypin sivuille on oltava mahdollista lisätä nappeja
6	NFC-toiminto	Prototyyppiin tulee olla mahdollista lisätä NFC -tageja lukeva

		toiminto
7	2D-viivakoodi	Prototyyppiin tulee olla mahdollista lisätä 2D-viivakoodin lukeminen
8	Navigointi	Prototyypin nappeja painaessa tulee olla mahdollista navigoida toiselle sivulle
9	Nappien poistaminen	Prototyyppiin lisättyjä nappeja tulee voida poistaa
10	Sivut	Prototyyppiin tulee olla mahdollista lisätä eri sivuja
11	Nappien koko	Nappien koko tulee olla määritettävissä
12	Ohjelman tuotos	Ohjelman tulee tuottaa natiivi Windows Phone 8 -sovellus

3.3 Arkkitehtuurisuunnitelma

Suunnittelun viimeinen vaihe oli arkkitehtuurisuunnitelman toteuttaminen. Arkkitehtuurisuunnitelma luotiin lopullisen ohjelman toteuttamisen apuvälineeksi. Suunnitelman avulla tarkoitus oli tehdä ohjelman rakenteesta laadukas, jotta sitä on helppo jatkokehittää. Suunnitelma ei sisällä tarkkoja kuvauksia luokkien välisistä riippuvuuksista.



Kuva 6.

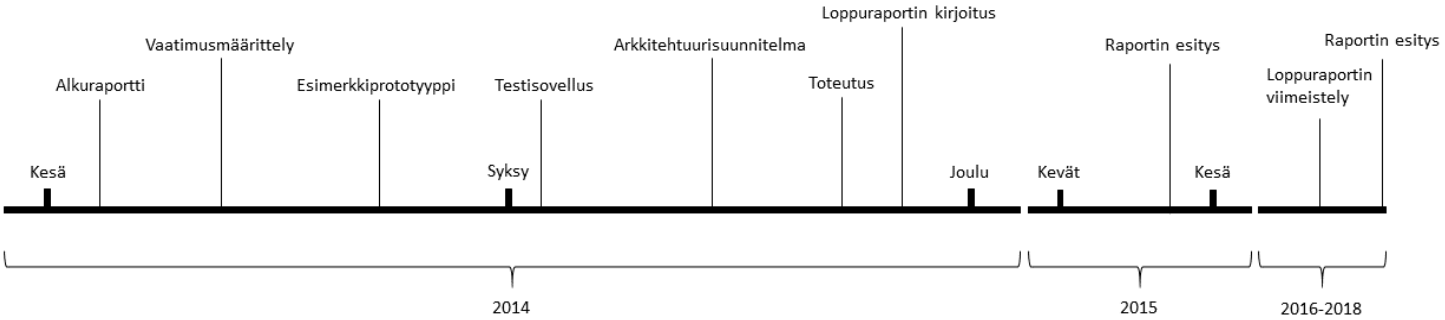
4 SOVELLUKSEN TOTEUTUS

4.1 Tavoitteet

Sovelluksella piti olla mahdollista luoda prototyyppejä Windows Phone 8 –puhelimille. Prototyypin piti toimia älypuhelimilla ja hyödyntää niiden NFC ja kamera toimintoja. Sovelluksen testaamista ja kehittämistä jatkettiin siihen asti, että sovelluksella on mahdollista tehdä kaikkia toimintoja hyödyntävä toimiva prototyyppi monin eri tavoin. Sovelluksen oli myös toimittava lähtökohtaisesti kaikilla Windows 8 tietokoneilla. Sovelluksen piti myös toimia niin, että prototyyppien tekeminen onnistuu ilman ohjelman kaatumista.

4.2 Toteutusprosessi

Ohjelman toteutus alkoi esimerkkiprototyypin tekemisellä. Tämän jälkeen toteutettiin testisovellus, johon toteutettiin suurin osa toiminnoista. Osa toiminnoista oli kuitenkin vajaita ja ohjelma oli yksittäisten toiminnallisuuksien yhdistelmä ehyen ohjelman sijaan. Tästä syystä lopullinen ohjelma toteutettiin edellä mainittujen sovellusten ja arkkitehtuurisuunnitelman jälkeen. Kuvassa 7 kuvataan koko työn toteutusjärjestys ja aikataulu.



Kuva 7.

4.2.1 Esimerkkiprototyyppi

Alun lyhyen suunnittelun jälkeen siirryttiin suunnittelemaan esimerkkiprototyyppiä. Esimerkkiprototyyppi oli mobiilisovellus, johon ohjelmoitiin yksi tapaus jokaisesta

toiminnosta, jota pääsovelluksen avulla on mahdollista lisätä. Tällä tavoin saatiin pohja pääohjelman ”vie” –toiminnolle, jonka avulla luodaan prototyypin lähdekoodi. Pääohjelma lisää esimerkkiprototyypin koodiin dynaamisesti sisällön sen hetkisen projektin mukaisesti.

Esimerkkiprototyypin toteuttaminen aloitettiin yksinkertaisesta Microsoft Visual Studion Windows Phone -projektista, josta siistittiin kaikki ylimääräinen pois. Tämän jälkeen esimerkkiprototyyppiin lisättiin tausta ja muutama nappi. Seuraava vaihe oli lisätä prototyyppiin toinen sivu, jotta nappeihin voitiin lisätä navigointi. Esimerkkiprototyypin toiminnallisuus pidettiin tarkoituksella mahdollisimman yksinkertaisena, joten pääsivulle ei lisätty mitään muuta. Tässä vaiheessa lisättiin myös toinen sivu, joka erosi pääsivusta ainoastaan nimensä puolesta. Kyseisen sivun tarkoitus on olla pohjana muille sovelluksen sivuille, kuin ensimmäiselle eli pääsivulle.

Esimerkkiprototyyppi sisältää myös NFC –sivun, jota tarvitaan yksinkertaista NFC-toimintoa varten. NFC-sivu pohjautuu esimerkkitiedostoon NFC:n käytöstä artikkelista ”Use NFC tags with Windows Phone 8” sivustolla developer.nokia.com, mitä on muokattu kandidaatintyön tarkoituksiin (Microsoft 2014). Sivulla on nappi, jota painamalla puhelin alkaa etsimään NFC-luettavia kohteita. Puhelimen löytäessä luettavan NFC-laitteen, ohjelma navigoi toiselle sivulle automaattisesti.

Viimeinen sivu esimerkkiprototyypissä on 2D-viivakoodin lukemista varten. 2D-viivakoodin lukutoiminto pohjautuu sivustolla ”developer.nokia.com” julkaistuun artikkeliin ”Generating and scanning barcodes using ZXing on Windows Phone” ja artikkelissa esitettyä koodia on muokattu kandidaatintyötä varten erilaiseen muotoon (Microsoft 2014). Sivu käyttää puhelimen kameraa ja näyttää siitä tulevaa kuvaa. Kun kamera löytää viivakoodin, se ilmoittaa siitä kahdella palkilla kuvassa. Sivulle lisättiin teksti, joka ilmoittaa viivakoodin sisällön. Tekstin alle lisättiin myös ”Avaa linkki” –nappi, jota voi käyttää, jos viivakoodin sisältö on linkki www-sivulle.

4.2.2 Testisovellus

Seuraavaksi tehtiin esimerkkiprototyyppiä hyödyntäen testisovellus. Testisovellus sisältää toteutukset kaikista toiminnoista, jotka ovat lopullisessa sovelluksessa. Testisovelluksen oli

tarkoitus olla harjoitus, jonka avulla itse työ voidaan toteuttaa. Testisovellus rakennettiin kokeilemalla ja harjoittelemalla toimintojen tekemistä vaatimusmäärittelyn mukaisesti.

Testisovelluksen tekeminen aloitettiin luomalla tyhjä C# Windows Presentation Foundation –projekti. Tämän jälkeen siihen luotiin Grid –elementti, joka jaettiin kolmeen osaan. Vasemmalle ja oikealle jäi tilaa työkaluille, kun taas keskelle jää älypuhelimen näytön kokoinen työskentelyalue. Työskentelyalueeksi luotiin Canvas –elementti, jonka taustan väriksi valittiin valkoinen. Sovelluksen yläreunaan luotiin valikkoja, joihin lisättiin mahdollisuudet tiedoston avaamiselle, tallentamiselle ja viemiselle.

Seuraava vaihe oli mahdollistaa napin, eli tässä vaiheessa kuvan, lisääminen Canvas -elementtiin. Tässä vaiheessa projektia ei ollut vielä tiedossa, että WPF tarjoaa oman luokan neliöiden luomiseen. Kuvien lisäämisnapin jälkeen siirryttiin tekemään sovelluksen ehkä tärkeimpiä ominaisuuksia, eli nappien raahaamis- ja pudottamistoimintoja. Tähän kului paljon aikaa, sillä verkosta löytyneet toimintoon liittyvät esimerkit eivät auttaneet ja useiden erilaisten tapojen ja kokeilujen jälkeen asiassa ei päästy eteenpäin. Lopulta kuitenkin ohjelma saatiin toimimaan oikein.

Raahaa ja pudota –toimintojen jälkeen keskityttiin vielä nappeihin liittyviin ominaisuuksiin ja toimintoihin. Nappeja oli tarkoitus olla mahdollista poistaa tai niiden kokoa tuli voida muuttaa. Tässä vaiheessa ongelmaksi muodostui kuvan koon muuttaminen. Ongelman motivoimana etsimällä löytyi WPF:n oma neliö -luokka, joka tarjosi useita etuja kuvaan verrattuna, kuten esimerkiksi koon ja värin muokkaaminen. Tässä vaiheessa kaikki nappeihin liittyvät kuvaelementit korvattiin neliöillä, ja sen jälkeen napeille luotiin ohjaimet niiden korkeuden, leveyden ja värin muokkaamiseen. Canvas –elementiltä löytyvien nappi olioiden ominaisuuksia oli yhdistettävä erilaisiin raahaustyökaluihin ja valikkoihin. Tämä tarkoitti WPF DataBinding –toimintojen hyödyntämistä. Kyseisessä vaiheessa meni keskimääräisesti pidempi aika, sillä aihetta täytyi opiskella paljon, ennen kuin ohjelma saatiin toimimaan oikein.

Nappien ja niihin liittyvien kontrollien onnistumisen jälkeen ohjelmalla pystyttiin luomaan käytännössä yksi prototyypin sivu, joka sisältää halutun määrän nappeja. Tässä vaiheessa testisovelluksen toteuttamisessa siirryttiin projektin tiedostoon tallentamiseen.

Tallennusmuodoksi valittiin XML, koska WPF tarjosi siihen valmiit työkalut ja koska elementtien ja olioiden tallentaminen XML-kielen ja XAML-kielen yhtäläisyyksien takia oli luontevaa. Tallennus toteutettiin siten, että ohjelma käy läpi kaikki Canvas –elementin lapset ja kirjoittaa niistä tiedot tiedostoon. Tällaisia tietoja olivat esimerkiksi sijainti, leveys ja korkeus. Toteutuksessa helpotti WPF:n tarjoama valmis Windows käyttöjärjestelmistä tuttu selaustyökalu tiedoston tallentamista varten.

Tallentamistoiminnon jälkeen siirryttiin tiedoston lataamiseen. WPF tarjoaa omat työkalut myös XML-tiedostojen lukemiseen. Ensin käytettiin työkaluja ja varmistettiin, että ne lukevat oikeat tiedot tiedostoista. Seuraavaksi luotiin funktiot, jotka lukevat tiedostojen sisällön ja sen mukaan tuovat käynnissä olevaan ohjelmaan tiedostoon tallennetun projektin. Tässä vaiheessa toteutusta hyödynnettiin samankaltaista tiedostojen selaustyökalua, kuin tallentamisen toteuttamisessa. Tiedostosta ladattava projekti täytyi luoda täysin tyhjälle pohjalle, joten ohjelma käynnistetään uudelleen lataamisvaiheessa. Ennen kuin ohjelma sammuu, kopioi se halutun ladattavan tekstitiedoston sovelluksen omaan kansioon. Sovellus lukee ensimmäiseksi kyseisen tiedoston.

Tässä vaiheessa toteutus ajautui eri suuntaan, kuin mitä suunnitelmassa oli määritetty. Esimerkkisovelluksen koodia ei ollut mahdollista generoida toimivaksi mobiilisovellukseksi ilman Microsoftin Visual Studiota, eli sitä ei ollut mahdollista hyödyntää alkuperäisen suunnitelman mukaisesti. Tässä vaiheessa esimerkkisovellus oli kuitenkin jo tehty ja testisovelluksen toteutukseen oli kulunut runsaasti aikaa, joten kehitystä jatkettiin alkuperäisestä suunnitelmasta poikkeavalla ratkaisulla.

Seuraavaksi toteutettiin esimerkkisovellukseen pohjautuva ”Export” –toiminto, jonka avulla on mahdollista luoda ohjelman auki olevasta projektista toimiva mobiilisovellus. Aluksi tavoitteena oli saada testisovellus kirjoittamaan ja kopioimaan tiedostot toimivaksi kokonaisuudeksi. Tämä tarkoitti sitä, että luotava prototyyppi sisältää aina samat napit ja saman taustan. Osa luotavista tiedostoista oli mahdollista kopioida suoraan haluttuun hakemistoon. Kopioituihin tiedostoihin sisältyi myös ZXing –kirjasto, jonka kaikkien tiedostojen kopiointiin hyödynnettiin valmista koodia, jonka avulla toistorakenteessa käydään läpi jokainen hakemisto ja niiden alihakemistot ja kopioidaan sieltä löytyvät tiedostot.

Osa tiedostoista täytyi kirjoittaa, koska niihin lisätään dynaamisesti nappeja ja taustakuvia. Tiedostoon kirjoittaminen yksinkertaisesti WPF:n XmlTextWriter –luokkaa hyödyntäen. Luokan avulla esimerkkiprototyypin lähdekoodi kirjoitetaan rivi riviltä luotavaan tiedostoon. Avatun projektin mukaisten nappien lisääminen vie –toimintoon toteutettiin heti, kun testisovelluksella luotava prototyyppi sisälsi oikeat asiat ja toimi oikein.

Seuraavaksi vuorossa oli niiden työkalujen ja ominaisuuksien lisääminen, jotka vielä puuttuivat testisovelluksesta. Ensin mahdollistettiin sivujen lisääminen ja poistaminen prototyyppiin. Tämä tapahtui WPF:n TabControl ja TabItem –luokkien avulla. Lopuksi toteutettiin toiminto, jonka avulla käyttäjä voi lisätä kuvia prototyypin sivujen taustalle. Tästä eteenpäin uudet toiminnot ja parannukset tehtiin uudelle pohjalle kirjoitetulle pääsovellukselle.

4.2.3 Pääsovellus

Pääsovelluksen toteutus perustui testisovelluksista opittuihin asioihin ja toimintoihin. Lisäksi pääsovellukseen lisättiin puuttuvia oleellisia toimintoja. Toteutuksessa noudatettiin testisovelluksen jälkeen toteutettua arkkitehtuurisuunnitelmaa. Pääsovelluksen toteuttaminen aloitettiin kirjoittamalla ohjelman pohja uudelleen. Tämä tapahtuu nopeasti ja suurimmat muutokset olivat koodin järjestely ja ScrollViewer –elementin lisääminen. Kyseisen elementin avulla Canvas –elementtiä voi palkin avulla selata ylös ja alas, jos sovellusta käyttää esimerkiksi pieninäyttöiseltä kannettavalta tietokoneelta. Kaikki näkyvät osuudet ohjelmasta sijoitettiin uudelleen ja niiden ulkoasua muutettiin.

Seuraava vaihe oli arkkitehtuurisuunnitelman mukaisten luokkien lisääminen sovellukseen. Luokkien toteutukset, funktiot ja attribuutit tehtiin järjestyksessä, joka mahdollistaa niiden testaamisen. Kaikki edellä mainitut osuudet toteutettiin seuraavassa järjestyksessä:

- Nappien lisääminen
- Drag & drop
- Nappien poistaminen
- Nappien koon muuttaminen

- Yläpalkin valikko
- Sivujen lisääminen ja poistaminen
- Projektin tallentaminen ja lataaminen
- Prototyypin luominen nykyisestä projektista

Osa toiminnoista tuotiin suoraan testisovelluksesta ja osaan toiminnoista tehtiin pieniä muokkauksia. Suurimmaksi osaksi koodia järjestettiin ja yhtenäistettiin, mutta esimerkiksi tiedostoja käsittelevät osiot muutettiin toimimaan missä tahansa hakemistossa. Neliön raahaamisessa muutettiin myös neliön putoaminen siten, että sen vasen ylänurkka tulee kursorin osoittamaan paikkaan neliön keskustan sijasta.

Ohjelman yleisen parantamisen ja korjaamisen jälkeen siirryttiin lisäämään puuttuvia tärkeitä toimintoja ja ominaisuuksia. Ensimmäinen lisätty toiminto oli nappien värin muuttaminen. Tämä toiminto on tarpeellinen, jotta prototyyppejä tehdessä napit eivät häviä samanväriseen taustaan. Seuraavaksi ohjelmaan lisättiin mahdollisuus lisätä erikoissivut NFC ja 2D-viivakoodia varten. Erikoissivuille ei ole mahdollista lisätä mitään tässä versiossa. Tämän jälkeen pääsovellukseen lisättiin mahdollisuus valita jokaiselle napille oma navigointikohde. Kohteeksi voi valita projektiin luotujen sivujen lisäksi ”Default” –vaihtoehdon, jolloin kohde on automaattisesti seuraava sivu.

Tässä vaiheessa sovelluksiin oli luotu useita uusia mahdollisuuksia, joita ei ollut kuitenkaan mahdollista tallentaa tai viedä mihinkään muotoon. Tästä syystä seuraavaksi uusien toimintojen hyödyntämismahdollisuus lisättiin myös tiedostoon tallentamiseen. Tämä tapahtui yksinkertaisesti lisäämällä XML-tallennukseen toimintoja uusien elementtien luomiseen. Tiedostoon tallentamisen jälkeen siirryttiin tiedostosta lataamiseen, joka tarvitsi vastaavat lukutoiminnot uusille elementeille. Niiden toteuttaminen onnistui ongelmitta ja seuraavaksi siirryttiin ”Export” –toiminnon päivittämiseen.

Ohjelma oli tässä vaiheessa suurimmaksi osaksi toiminnoiltaan valmis, mutta lopputuloksen kannalta yksi tärkeimmistä funktioista puuttui vielä. Pääsovelluksen täytyi voida luoda prototyyppi, joka vastaa täysin ohjelmalla tehtyä projektia. Tässä vaiheessa pääsovelluksella oli mahdollista luoda sovellus, joka sisälsi vain yhden sivun ja napit auki olevasta projektista. Tästä syystä seuraavaksi pääsovelluksen ”Export” –luokkaan lisättiin

toiminnot usean sivun luomiselle. Samalla luokkaan lisättiin navigointimahdollisuudet NFC ja 2D-viivakoodi sivuille.

4.3 Toteuma

Lopullinen sovellus on Windows 8 –käyttöjärjestelmällä toimiva työpöytäsovellus. Sovelluksen käyttöliittymä rakentuu neljästä osasta, jotka ovat vasen ylävalikko, vasen valikko, työskentelyalue ja oikea valikko. Sovelluksen ikkuna skaalautuu pienempiin näyttöihin, koska työskentelyaluetta on mahdollisuus rullata ylös sekä alas ja valikkojen elementit tiivistyvät pieneen tilaan tarvittaessa. Ikkuna käynnistyy täyttäen koko ruudun ja siinä on automaattisesti tyhjä pohja auki.

Vasemmasta valikosta löytyy omat säädöt työskentelyalueelle luotavien nappien leveyksille ja korkeuksille. Valikosta löytyy myös painike työskentelyalueen valittuna olevan napin poistamiselle. Painikkeen alta löytyy laatikko, josta voidaan valita eri värejä työskentelyalueen napeille. Alimmasta laatikosta valitaan, mille sivulla valittua nappia painamalla on tarkoitus navigoida.

Työskentelyalue löytyy keskeltä sovellusta ja se on kokoa 480 x 800, joka vastaa kooltaan osaa Windows Phone 8 –puhelimien näytöistä. Työskentelyalueella on mahdollista raahata ja pudottaa sinne lisättäviä nappeja. Työskentelyalueen taustaksi on mahdollista lisätä haluamansa kuva, kuten esimerkiksi yhden kuvan oman mobiilisovelluksen sivuista. Alueen ylälaidasta löytyy jokaiselle välilehdelle oma palkki, jota painamalla voi valita työskentelyalueelle eri välilehtiä. Oikealta löytyvästä valikosta löytyy painike napin lisäämiselle työskentelytasolle. Sen alapuolelta löytyy painike valittuna olevan välilehden poistamiselle.

Ylävalikossa on kaksi eri valikkoa, joista ensimmäinen sisältää mahdollisuudet tyhjän projektin avaamiselle sekä projektin tallentamiselle ja lataamiselle. Edellä mainittujen valintojen lisäksi valikosta on mahdollisuus lisätä työskentelytasolle kuva ja luoda sen hetkisestä projektista MS Visual Studio Windows Phone 8 –projekti. Toisesta ylävalikon valikosta löytyy mahdollisuudet sivujen eli välilehtien lisäämiseen työskentelytasolle.

Sivuja voi lisätä kolmea eri tyyppiä, jotka ovat normaali sivu, NFC-sivu ja 2D-viivakoodisivu.

Sovelluksella on mahdollisuus luoda kuvien ja nappien avulla MS Visual Studio Windows Phone 8 –projekti, joka sisältää kaikki pääsovelluksen avulla lisätyt ominaisuudet. Visual Studion avulla projekti voidaan siirtää puhelimelle, jossa mobiilisovellus on pääsovelluksella suunniteltu ja luotu prototyyppi. Sovelluksella on mahdollisuus lisätä mielivaltainen määrä normaaleja sivuja ja yhdet NFC- ja 2D-viivakoodi –sivut. Normaalilla sivulla on mahdollista painaa nappeja, jotka navigoivat yhdelle sovelluksen sivuista. NFC-sivulla on mahdollista havaita NFC-lukukelpoisia kohteita, joita löytäessä sovellus ilmoittaa ”NFC ok”. 2D-viivakoodi –sivulla koodien skannaaminen käynnistetään painamalla nappia, jonka jälkeen näyttöön ilmestyy kameran kuvaama näkymä. Kameraa suunnattaessa viivakoodia kohti, sovellus tunnistaa viivakoodin ja ilmoittaa sen sisällön.

4.4 Toteuman analysointi

Sovellus sisältää tavoitteissa mainitut yksinkertaiset toiminnot ja ominaisuudet. Suurin puute ohjelmassa on kuitenkin sen jaettavuus. Ohjelma olisi ollut paras toteuttaa siten, että se tuottaisi mobiilisovellukselle luettavan xml-tiedoston sen sijaan, että se tuottaa mobiilisovelluksen Visual Studion projektin muodossa. Tähän päätökseen päädyttiin siksi, että testisovellusta kehitettiin liian pitkään ja pitkälle. Mobiilisovelluksen erillinen toteutus olisi vienyt liian paljon aikaa. Testisovelluksen kehittäminen olisi pitänyt lopettaa muutaman perustoiminnallisuuden jälkeen, jonka jälkeen olisi ollut mahdollista suunnitella kunnolla koko sovellus. Sen lisäksi, jos toiminnallisuudet rajattaisiin vielä tarkemmin, olisi paras toteutus ollut mahdollista saavuttaa määräaikaan mennessä.

Suurimmat puutteet sovelluksen toiminnallisuudesta löytyvät toimintavarmuudesta. Sovelluksen testaamista on jatkettu vain siihen asti, että sovelluksella on ollut mahdollista tehdä kaikkia toimintoja hyödyntävä prototyyppi. Tällä hetkellä tiedossa on yksi ongelma, jonka takia ohjelma saattaa joskus jäädä käynnistymättä. Tästä syystä ohjelma ei täysin vastaa tavoitteita, joiden mukaan ohjelman tulisi toimia ilman kaatumisia. Ohjelma hyödyntää tiedostoon tallennettua arvoa käynnistyksessä ja siihen liittyen on todennäköisesti olemassa tilanteita, joissa ohjelman logiikka pettää.

4.5 Haasteet

Kandidaatintyöhön liittyviä haasteita oli eniten alkuvaiheessa. Sekä suunnitteluun että toteutukseen vaikutti kokemattomuus ohjelmointikielessä. Suunnittelun osalta se tarkoitti sitä, että oli vaikeaa tehdä kattava suunnitelma toteutuksesta. Se vaikeutti myös huomattavasti tarvittavan ajan arvioimista, joka oli yksi tämän työn suurimpia ongelmia. Näiden syiden takia testisovelluksen ja pääsovelluksen suunnittelu ei onnistunut ja toteutus venyi yli aikataulun.

Ohjelmoinnin kanssa tuli vastaan aika ajoin haasteita, joiden ratkaisemiseen kului pidempi aika. Suurimpia haasteita olivat esimerkiksi raahaa, pudota ja vie toiminnot. Raahaamiseen ja pudottamiseen C#-kielellä löytyi jonkun verran esimerkkejä internetistä, mutta mikään ei kuitenkaan vastannut toteutettavan sovelluksen tarpeita. WPF:sta löytyy luokka kyseisille toiminnoille, mutta tiedon siirtäminen luokasta löytyvien funktioiden välillä ei toteutettavassa sovelluksessa toiminut. Lopulta toteutus oli sekoitus internetistä löytyvistä esimerkeistä, WPF dokumentaation esimerkeistä ja omasta ratkaisusta.

”Export” –luokka oli ongelma suuren kokonsa takia, koska kaikkien osien oli toimittava täsmälleen oikein, jotta prototyypin luominen onnistuisi. Luokan avulla luotiin useita erilaisia tiedostoja ja kirjoitettiin XAML-koodia. Toteutuksessa kirjoitetaan koodilla koodia, joka vaati kahden syntaksin sääntöjen noudattamisessa suurta tarkkuutta. ”Export”-luokan tekemistä vaikeutti myös se, että helpoin tapa testata luokan toimivuutta oli luoda ja tarkastella sovelluksen luomaa erillistä prototyyppiprojektia. Tämä vaati joka testauskerran jälkeen uuden projektin avaamista ja tulkitsemista.

4.6 Johtopäätökset toteutuksesta

Toteutus sekä testisovelluksen ja pääsovelluksen osalta onnistui kohtuullisesti. Aikataulun kannalta toteutus venyi asetettujen rajojen ulkopuolelle. Tämä johtui suurimmaksi osaksi testisovelluksen toteuttamisen venymisestä. Lopputuloksena oli kuitenkin suurimmaksi osaksi tavoitteita vastaava sovellus, johon jäi kuitenkin paljon kehitettävää. Testisovelluksesta oli hyötyä pääsovelluksen toteutuksen ja suunnittelun kannalta, mutta sitä kehitettiin työhön nähden liian pitkälle. Tästä syystä sovelluksen tuottamia

prototyyppejä on mahdollista saada puhelimelle ainoastaan Visual Studion kautta. Toteutettavien ominaisuuksien määrää olisi kannattanut karsia ja keskittyä enemmän työn aiheessa esiteltyihin vaatimuksiin. Tällä tavoin lopputulos olisi voinut olla lähempänä alkuperäistä tavoitetta.

4.7 Parannukset sovellukseen

Sovellusta tehdessä nousi esille paljon uusia ajatuksia ja tarpeita sovelluksen toteuttamiseen liittyen, mistä osa lisättiin sovelluksen vaatimusmäärittelyyn ja osa jätettiin tulevaisuuden kehitysideoiksi. Sovelluksessa on useita tärkeitä ja oleellisia ominaisuuksia ja toimintoja, jotka täytyy muuttaa ja lisätä. Osa muutoksista on niin suuria, että osia sovelluksen perustoiminnoista tulisi korvata tai jättää pois uusista versioista. Toteutettavat parannukset voidaan jakaa neljään kategoriaan:

- Parannukset, joiden avulla sovelluksen toimintavarmuutta parannetaan ja vakavia virhetiloja vähennetään
- Parannukset, joiden avulla sovelluksen käytettävyyttä parannetaan huomattavasti
- Parannukset, joiden avulla sovelluksesta tulee hyödyllinen
- Parannukset, joiden avulla sovelluksen mahdollista suurempaa potentiaalia yritetään saavuttaa

Tärkeimmät sovellukseen lisättävät ominaisuudet tulevaisuuden kannalta liittyvät sovelluksen laatuun ja toimivuuteen. Sovelluksen toimintavarmuutta pitää parantaa lisäämällä sovellukseen virheenkäsittelyä ja korjaamalla tiedostosta lukemiseen liittyvän ongelman, joka saattaa joissakin tilanteissa estää ohjelman käynnistymisen. Sovelluksessa on vielä useita osuuksia, joissa virheenkäsittely on vajaa tai se puuttuu kokonaan. Kriittisempien korjauksien jälkeen sovellusta pitäisi testata laajemmin useammilla käyttäjillä, jotta suurimmat käyttäjäkokemusta haittaavat virhetilat saataisiin karsittua.

Sovellukseen liittyy myös parannuksia, joista useat ovat yleisiä toimintoja suurimmassa osassa kaupallisista sovelluksista. Esimerkiksi tällä hetkellä neliöiden kokoa vaihdetaan kahdesta eri palkista, kun se yleensä tehdään neliötä itseään venyttämällä. Käyttöä nopeuttaisi myös huomattavasti neliöiden ”kopioi ja liitä” -toimintojen lisääminen.

Sovelluksen tulisi myös tukea useampia tiedostomuotoja kuvien lisäämiseen, jottei käyttäjien tarvitse turhaan muokata tiedostoja juuri oikeaan muotoon.

Sovellus on tällä hetkellä hyvin yksinkertainen ja sen tuottamat prototyypit pystyvät esittämään yksinkertaisia sovelluksia. Sovellukseen tulisi lisätä lisää erilaisia mahdollisuuksia käyttää NFC ja 2D-toimintoja, sillä tällä hetkellä NFC ja 2D-sivuja ei ole mahdollista muokata käyttäjän tarpeiden mukaisesti millään tavalla. Sovellukseen olisi myös tarpeellista lisätä mahdollisuudet kaikkien normaalien toimintojen ja animaatioiden lisäämisille, joita mobiilisovellukset ja muut prototypointiohjelmat käyttävät.

Sovellus eroaa muista kilpailijoistaan puhelimen sensorien käyttämisen ansiosta. Tähän ominaisuuteen tulisi keskittyä ja niihin liittyviä toimintoja ja mahdollisuuksia laajentaa. Ensin nykyisten NFC ja 2D-viivakoodin hyödyntämistä on laajennettava reilusti, jonka jälkeen voidaan siirtyä luomaan toimintoja muille puhelimen sensoreille. Toiminnat tulee kuitenkin pysyä riittävän yksinkertaisina, jotta koodin generoiminen pysyy mahdollisena. Tällä hetkellä sovellusta rajoittaa myös riippuvaisuus MS Visual Studiosta. Tulevaisuudessa yksi tärkeimmistä kehityskohteista on mobiilisovelluksen luominen, joka lukee pääsovelluksien luomia XML-tiedostoja ja niiden avulla mahdollistaa prototyyppien suorittamisen puhelimella.

5 POHDINTA JA TULEVAISUUS

Sovellus sisältää suurimmaksi osaksi kaikki suunnitellut toiminnot, mutta tässä muodossa se ei ole vielä hyödyllinen. Siitä puuttuu vielä paljon pääsovelluksen käyttämiseen ja prototyypin ominaisuuksiin liittyviä asioita. Sovelluksen jatkokehityksessä täytyy kuitenkin ensin keskittyä hyvän perustan luomiseen ja kriittisimpien korjauksien tekemiseen.

Työ oli lopputulokseltaan onnistunut, mutta aikataulultaan epäonnistunut. Suurimmaksi ongelmaksi nousi testisovelluksen toteutuksen venyminen. Toteutusvaiheessa nousi ongelmaksi se, että rajauksia ja tavoitteita ei oltu käyty tarpeeksi kattavasti läpi. Siitä syystä sovelluksien kehittämisen valmiiksi saaminen venyi. Suunnitteluvaiheessa olisi pitänyt rohkeammin tehdä tarkemmat suunnitelmat ja rajaukset, vaikka ohjelmointikieli oli tuntematon. Kokemattomuuden huomioiminen olisi ollut järkevämpää toteuttaa rajaamalla vaatimuksia kuin lisäämällä työtä

Sovellusta on mahdollista jatkokehittää muille mobiilialustoille, koska sillä voi tallentaa prototyyppejä XML -muotoon. Työpöytäsovellusta on mahdollista kehittää siihen pisteeseen, että se olisi saatavilla Microsoft Storesta. Sovelluksen jatkokehittäminen saattaa jatkua, jos muilla kursseilla on mahdollista toteuttaa mobiilisovellus esimerkiksi android -alustalle.

6 YHTEENVETO

Työn lopputuloksena on sovellus, jolla on mahdollista tehdä prototyyppeja Microsoft Visual Studion projektin muotoon. Prototyyppeihin voi lisätä painikkeita, joita painaessa voidaan avata toinen näkymä prototyyppisovelluksessa. Taustalle on mahdollista lisätä kuvia, jotta prototyypit saa näyttämään enemmän oikealta sovellukselta. Lisäksi sovellukseen on mahdollista lisätä 2D -viivakoodin lukemiseen tarkoitettu näkymä sekä NFC-tunnisteesta aktivoituva näkymä. Ohjelmalla on myös mahdollista tallentaa projektit XML -muotoon, joita hyödyntäen ohjelmaa voisi hyödyntää esimerkiksi muiden alustojen sovellusten kanssa.

7 LÄHTEET

Arnowitz J., Arent M. & Berger N. 2007. Effective Prototyping for Software Makers. San Diego: Elsevier, Inc.

IEEE 1990. Prototyping revisited. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=113653&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D113653. Luettu 11.6.2014.

Justinmind 2014. Build the right apps from the start. http://www.justinmind.com/prototyper/features-build. Luettu 12.6.2014.

Microsoft 2014. App architecture. http://msdn.microsoft.com/en-us/library/windowsphone/develop/br211361.aspx. Luettu 11.6.2014.

Microsoft 2014. Windows Phone: Generating and scanning barcodes using ZXing on Windows Phone. https://social.technet.microsoft.com/wiki/contents/articles/27366.windows-phone-generating-and-scanning-barcodes-using-zxing-on-windows-phone.aspx. Luettu 30.6.2014

Microsoft 2014. Windows Phone: Use NFC tags with Windows Phone 8. https://social.technet.microsoft.com/wiki/contents/articles/27293.windows-phone-use-nfc-tags-with-windows-phone-8.aspx

Nielsen, J. 1993. Usability Engineering. San Diego: Academic Press.

Pidoco 2014. The Rapid Prototyping Tool. https://pidoco.com/en. Luettu 12.6.2014.

Preece, J., Rogers, Y. & Sharp, H. 2002. Interaction Design: Beyond Human - Computer Interaction. New York: John Wiley & Sons, Inc.

Samsung 2014. Samsung Galaxy S5 http://www.samsung.com/global/microsite/galaxys5/specs.html Luettu 31.1.2015

Schrage, M. 1996. Cultures of Prototyping: New York: ACM

Szekely P. 1995 User interface prototyping: Tools and techniques. In: Taylor R.N., Coutaz J. (eds) Software Engineering and Human-Computer Interaction. SE-HCI 1994. Lecture Notes in Computer Science, vol 896. Berlin: Springer

Tilastokeskus 2013. Tiede, teknologia ja tietoyhteiskunta. http://www.stat.fi/tup/suoluk/suoluk_tiede.html. Luettu 11.6.2014.

Zhixian, Y. & Dipanjan, C. 2014. Semantics in Mobile Sensing. California: Morgan & Claypool