SCRUM-KEHYSRAKENTEEN SOVELTAMINEN YKSIN TOTEUTETTAVAAN PROJEKTIIN

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietotekniikan koulutusohjelma

Forssa, kevät 2014

Antti Horkka

Forssa
Tietotekniikan koulutusohjelma
Tietokonetekniikan suuntautumisvaihtoehto

Tekijä Antti Horkka **Vuosi** 2014

Työn nimi Scrum-kehysrakenteen soveltaminen yksin toteutettavaan projektiin

TIIVISTELMÄ

Nykypäivän pienyritys kohtaa jatkuvasti vaihtuvia haasteita asiakkaiden palvelemisessa. Helposti päivitettävät ja yrityksen tarpeisiin räätälöidyt internetsivut ovat yksi parhaimmista tavoista palvella asiakkaita. Suurien ohjelmistotuottajien tarjoamat ohjelmistot ja palvelut ovat kuitenkin usein kokonaisuuksina liian laajoja sekä maallikolle vaikeakäyttöisiä.

Tässä opinnäytetyössä selvitettiin kuinka toteuttamiskelpoinen ajatus on yksinkertaisen ja helppokäyttöisen sisällönhallintaohjelmiston tuottaminen nollapisteestä asiakkaan toiveita varten kootuksi. Työssä perehdyttiin myös ketterän ohjelmistokehityksen toimintamallien soveltuvuuteen yksin toteutettaviin projekteihin.

Työn tarkoitus oli keskittyä nimenomaan ohjelmistotuotannolliseen puoleen, itse ohjelmisto oli opinnäytetyön sivutuote. Ohjelmiston tuli kuitenkin täyttää toimeksiantajan asettamat vaatimukset sekä olla toiminnaltaan selkeä ja jatkokehityskelpoinen.

Avainsanat Ketterä ohjelmistokehitys, Sisällönhallintajärjestelmä, PHP, Scrum

Sivut 12 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

FORSSA
Degree Programme in Information Technology

Author Antti Horkka **Year** 2014

Subject of Bachelor's thesis Adapting the Scrum framework to a solo project

ABSTRACT

The small company of today is faced with constantly changing requirements as to customer service. An easily updateable and tailor-made website is one of the best ways to meet these requirements. Readily commercially available content management software packages are often either too broad for a small company or too complicated for a layman to modify for their specific needs.

The purpose of this thesis was to find out how feasible it would be to design and produce a simple and easy-to-use, tailor-made content management software for a small company. In this thesis I will also examine the adaptability of agile software development methods for solo programming.

The focus of this thesis project was on the software development process itself whereas the produced software was a by-product of the software development process. The software still had to meet the criteria set by the client. It also had to be functionally sound and open for further development.

Keywords Agile software development, content management software, PHP, Scrum

Pages 12 p.

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön toimeksiantaja tarvitsi ohjelmiston, jolla yrityksen internetsivujen päivittäminen ja sisällönhallinta tapahtuu. Ohjelmiston avulla piti pystyä luomaan ja muokkaamaan tekstimuotoisia uutisartikkeleita, kuva-albumeita sekä tarpeen tullen näiden yhdistelmiä.

Toimeksiantaja halusi ohjelmiston olevan heidän tarpeisiinsa räätälöity jolloin valmiit sisällönhallintasovellukset (esim. WordPress) eivät soveltuneet tarkoitukseen. Ohjelmiston tuli olla myös käyttöliittymältään intuitiivinen ja helposti käytettävä.

Ohjelmiston tuli olla ylläpidollisesti halpa. Tästä syystä päätettiin toteuttaa ohjelmisto PHP-kielellä, koska ASP.NET-pohjaisen sovelluksen ylläpitokustannukset olisivat olleet liian korkeat. Valintaan vaikutti myös opinnäytetyön tekijän tausta PHP-ohjelmoijana, jolloin voitiin keskittyä itse ohjelmistokehitykseen sen sijaan, että projektia varten olisi pitänyt opetella uusia ohjelmointimenetelmiä.

Opinnäytetyön aikana perehdyttiin Scrum-kehysrakenteeseen ja sen soveltamiseen yksin toteutettavaan ohjelmointityöhön. Pintapuolisesti tutkittiin myös muita ketterän ohjelmistokehityksen menetelmiä ja työkaluja ja pohdittiin näiden soveltuvuutta projektiin.

Opinnäytetyön lopussa pohditaan menetelmien toimivuutta ja tehokkuutta, sekä määriteltyjen tavoitteiden täyttymistä.

2 KETTERÄ OHJELMISTOKEHITYS YLEISESTI

Vaikka ketterä ohjelmistokehitys on nimikkeenä melko uusi, voidaan sen juuret jäljittää 1950-luvulle asti. Tähän aikaan IBM:n palveluksessa olleet ohjelmistokehittäjät ymmärsivät, että suurien projektien toteuttaminen vanhan vesiputousmallin mukaan ei ollut mielekästä tai edes realistista. (Larman & Basili 2003, 47–56.) Kuitenkin vasta 1990-luvulla ketterä ohjelmistokehitys alkoi saamaan nykyisenkaltaista muotoaan esimerkiksi Scrumin ja Crystal Clear-kehysrakenteiden käytön yleistyessä (Larman 2004, 27).

2.1 Ketterän ohjelmistokehityksen julistus

Vuonna 2001 17 ohjelmistokehittäjää julkaisi allekirjoittamansa ketterän ohjelmistokehityksen julistuksen, joka on sittemmin käännetty 52 kielelle:

”Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja

Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota

Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja

Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta
arvostamme ensiksi mainittuja enemmän.”

(Beck ym. 2001.)

2.2 Lightweight Agile Development

LAD (Lightweight Agile Development) tarkoittaa sellaista ketterän ohjelmistokehityksen ideologiaa, jossa valitaan useista ketteristä ohjelmistokehitysmenetelmistä projektiin parhaiten sopivat, sekä muokataan niitä projektin ja kehittäjien tarpeiden mukaan (Highsmith 2000.) Käytännössä kaikki ketterä ohjelmistokehitys on jollain tasolla LAD:n tunnusmerkit täyttävää, koska liian monia menetelmiä seuratessa ketterän ohjelmistokehityksen mielekkyys ja hyödyllisyys katoaa.

Tämä ohjelmistokehitysprojekti olisi ollut käytännössä mahdoton toteuttaa ilman LAD:n kaltaista menettelytapaa. Tämä johtui suurimmaksi osaksi siitä, että useimmat ketterän ohjelmistokehityksen toimintamallit on suunniteltu tiimityöskentelyä varten.

3 UML

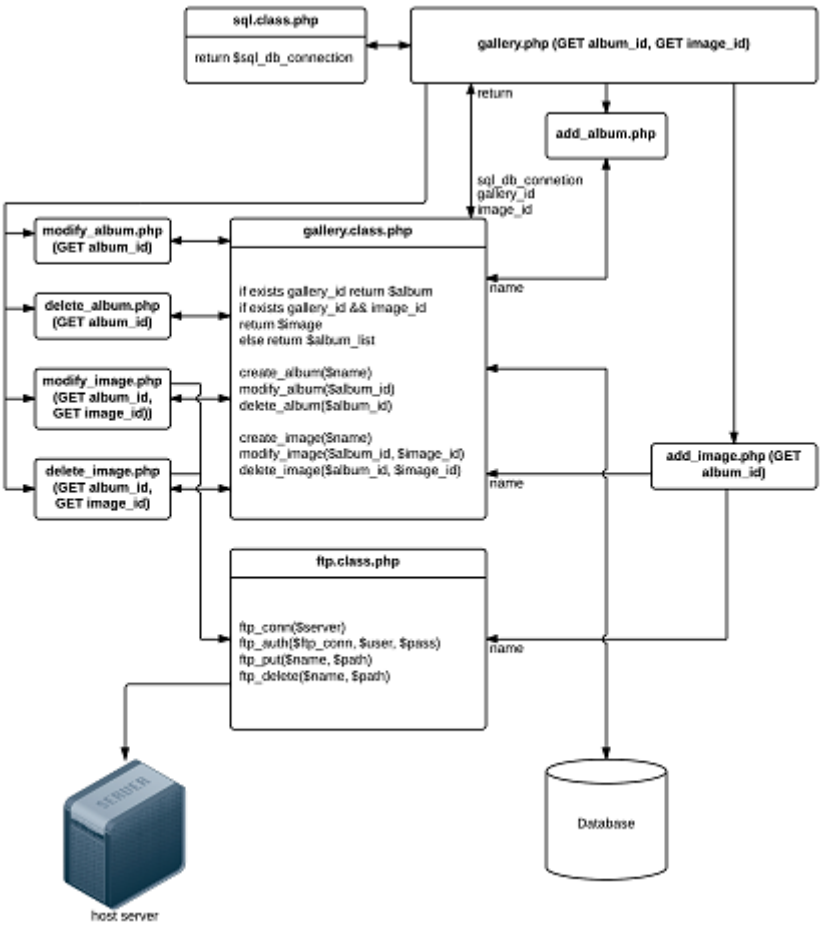
UML (Unified Modeling Language) on ISO-standardoitu mallinnuskieli ohjelmistojärjestelmille. UML ei sisällä menetelmiä järjestelmien toteuttamiseen vaan ainoastaan järjestelmien mallintamiseen ja kehitysprosessin dokumentointiin. Standardin uusin versio on UML2.0, joka sisältää 13 erilaista kaaviomallia jaoteltuna kolmeen kategoriaan. (Object Management Group 2014.)

3.1 Rakennekaaviot

Rakennekaaviot sisältävät työkalut järjestelmän pysyvän arkkitehtuurin mallintamiseen. Niiden avulla pystytään projektin alussa hyvin paikallistamaan mahdollinen redundanssi suunniteltavasta tuotteesta sekä virtaviivaistamaan etenkin tuotteen luokka- ja oliorakennetta. (Object Management Group 2014.)

Rakennekaavioihin kuuluvat luokkakaavio, oliokaavio, komponenttikaavio, komposiittirakennekaavio, pakettikaavio sekä

käyttöönottokaavio (Object Management Group 2014). Esimerkkinä rakennekaaviosta on opinnäytetyön kuva 1.

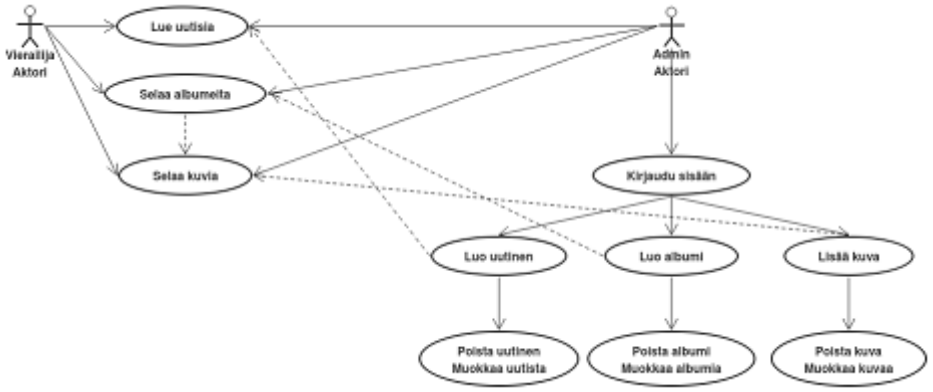


Kuva 1. Mallikaavio ohjelmiston luokkarakenteesta.

3.2 Käyttäytymiskaaviot

Käyttäytymiskaavioilla kuvataan mallin sisäisiä välittömiä tiloja sen suorittaessa sille annettuja tehtäviä. Näillä kaavioilla jäljitetään, miten järjestelmä toimii todellisessa käyttötilanteessa sekä tarkastellaan suorituksen tai tapahtuman vaikutuksia ja tuloksia. (Object Management Group 2014.)

Käyttäytymiskaaviot ovat käyttötapauskaavio, aktiviteettikaavio ja tilakonekaavio (Object Management Group 2014). Mallina käyttäytymiskaavioista on työn kuva 2

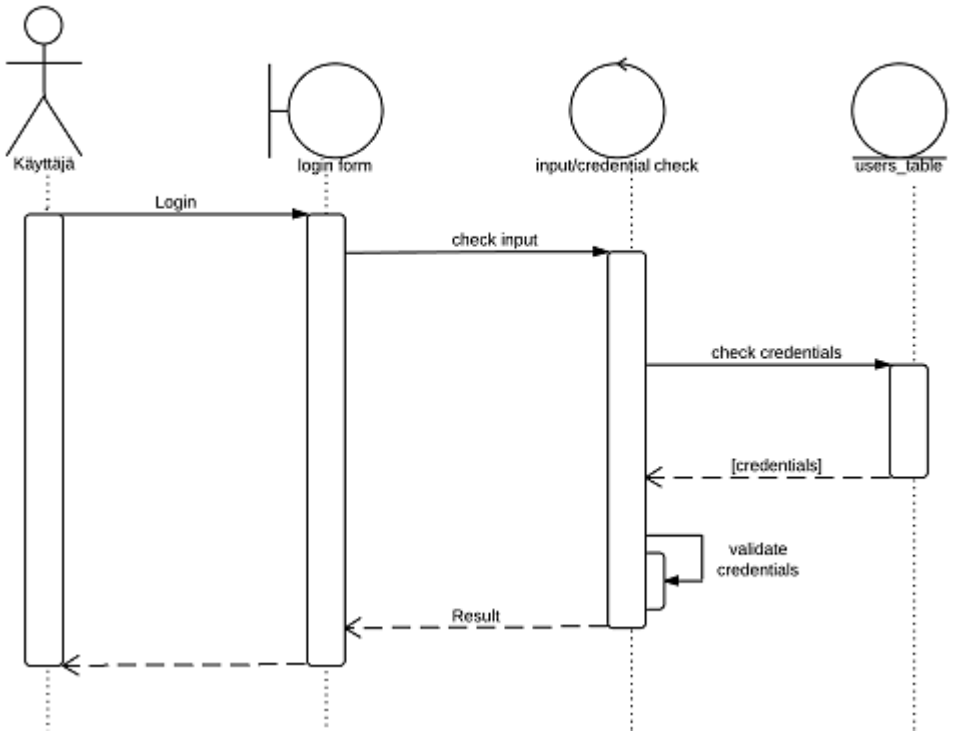


Kuva 2. Käyttötapauskaavio artikkelien CRUD-ominaisuuksista tunnistautuneena ja tunnistautumattomana käyttäjänä.

3.3 Vuorovaikutuskaaviot

Vuorovaikutuskaaviot ovat jatke käyttäytymiskaavioille. Ne auttavat mallintamaan käyttäytymiskaavioiden yksittäisten kohtien sisäisiä toimintoja. Vuorovaikutuskaaviot sisältävät työkalut mallintamaan järjestelmän sisällä toimivien osien, järjestelmän ja ulkoisten järjestelmien osien sekä järjestelmän ja käyttäjän välillä tapahtuvia vuorovaikutuksia. (Object Management Group 2014.)

Vuorovaikutuskaaviot ovat sekvenssikaavio, kommunikointikaavio, aktiviteettikaavio ja tilakonekaavio (Object Management Group 2014). Mallina vuorovaikutuskaavioista on työn kuva 3.



Kuva 3. Sekvenssikaavio sisäänkirjautumisprosessista.

3.4 UML:n soveltaminen projektiin

Tämä projekti oli alun perin suunniteltu vain paperille hyvin suurpiirteisin käsitekartoin. UML:n käytännön kokemuksen puuttuessa opeteltiin sen käyttö tätä projektia varten nollapisteestä alkaen.

Vaikka UML saattaa olla kokeneelle käyttäjälle erinomainen apuväline suurien järjestelmien suunnitteluun, koettiin se korkean oppimiskynnyksen ja monimutkaisuuden johdosta tarpeettoman hankalaksi, jotta se olisi ollut tehostamassa projektin etenemistä. Tuntuu siltä, että UML:n opiskeluun tulisi omistautua kokopäiväisesti, jotta sen pystyisi sisäistämään työtä tehostavalla tavalla. UML kielenä hyötyy kuitenkin siitä, että se on ISO-standardoitu ja täten universaali ja taaksepäin yhteensopiva.

4 SCRUM

Schwaber & Sutherland (2014, 2–3) määrittelevät Scrumin ketterän ohjelmistokehityksen kehysrakenteeksi, joka rakentuu tuotoksista, tapahtumista ja scrumtiimistä. Seuraavissa luvuissa käsitellään tarkemmin Scrumin osia sekä pohditaan niiden soveltuvuutta toteutettavaan projektiin.

4.1 Scrumin tapahtumat

Scrumiin sisältyy monia tapahtumia, joiden merkitys sisältö sekä sijainti aikajanalla on hyvin tarkkaan määritelty. Nämä tapahtumat ovat Sprintti, Sprintin suunnittelu, Päiväpalaveri, Sprintin katselmointi ja Sprintin retrospektiivi. (Schwaber & Sutherland 2013, 7–12.) Monet näistä tapahtumista ovat yksin toteutettavissa projekteissa tarpeettomia tai jopa mahdottomia suorittaa. Tästä huolimatta työssä pyrittiin käyttämään Scrumin tapahtumia mahdollisimman monipuolisesti hyödyksi.

4.1.1 Sprintti

Sprintti on Scrum-kehysrakenteen peruspilari. Sprint on se ajanjakso, jonka aikana scrumtiimin kehitystiimi valmistaa tuotteesta uuden tuoteversion. Yhden sprintin pituus tulee määritellä projektin alussa ja kaikkien sprinttien tulee olla koko projektin ajan samanmittaisia. Sprintti voidaan lopettaa kesken, mikäli sprintin aikana suoritettavat tehtävät tulevat uusien vaatimusten myötä tarpeettomiksi tai mikäli sprintin aikana suoritettavien tehtävien määrä on selvästi liian pieni ja ne saadaan valmiiksi huomattavasti odotettua aiemmin. (Schwaber & Sutherland 2013, 7–8.)

4.1.2 Sprintin suunnittelu

Jokaisen sprintin alussa tulee pitää kokous sprintin suunnittelua varten. Tässä kokouksessa määritellään tulevan sprintin sisältö ja tavoitteet. Sprintin suunnittelussa pyritään vastaamaan kysymyksiin ”mitä sprintin

aikana voidaan toteuttaa” ja ”miten näihin tavoitteisiin päästään”. Ensimmäiseen kysymykseen pyrkivät vastaamaan kehitystiimin jäsenet yhdessä tuoteomistajan (Scrumin roolit esitellään raportissa luvussa 4.3.) kanssa. Tuoteomistaja määrittelee, mitkä kohdat tuotteen kehitysjonosta tulee olla valmiina sprintin lopussa ja kehitystiimi vastaa näihin määrittelyihin omilla esityksillään ja vaihtoehdoillaan. Kun tavoitemääritelmät ovat aselvät, tulee kehitystiimin suunnitella, miten nämä tavoitteet toteutetaan. Kokouksen lopuksi kehitystiimin pitäisi pystyä esittämään tuoteomistajalle sekä scrummasterille suunnitelma siitä, miten he aikovat toimia sprintin aikana saavuttaakseen määritellyt tavoitteet. (Schwaber & Sutherland 2013, 8–10.)

4.1.3 Päiväpalaveri

Yksi tärkeimmistä Scrum-kehysrakenteen työkaluista on päiväpalaveri. Päiväpalaveri on kestoltaan lyhyt, noin 15 minuuttia kestävä palaveri, jossa kehitystiimi keskustelee edellisen päivän aikana tapahtuneesta edistyksestä ja tämän pohjalta suunnittelee tulevan päivän toimenkuvaa. Päiväpalaverissa pyritään myös luotaamaan asioita, jotka saattavat vaikeuttaa tavoitteiden saavuttamista. Päiväpalaveri on tarkoitettu ainoastaan kehitystiimin jäsenille ja tämän asian varmistaminen on scrummasterin vastuulla. (Schwaber & Sutherland 2013, 10–11.)

4.1.4 Sprintin katselmointi

Jokaisen sprintin jälkeen tulee kokoontua sprintin katselmointiin. Tähän kokoontumiseen kuuluvat kaikki scrumtiimin jäsenet sekä toimeksiantajan edustaja. Sprintin katselmoinnissa ei tehdä projektia muokkaavia päätöksiä, vaan sen aikana tulee tarkastella projektin edistymistä sekä tarvittaessa tehdä ehdotuksia tuotteen kehitysjonon muokkaamiseen. Sprintin katselmoinnin aikana tuoteomistajan tulee esitellä toimeksiantajalle, mitkä tuotteen kehitysjonon kohdista on saatu ”valmiiksi”. (Schwaber & Sutherland 2013, 11.) Scrum-kehysrakenteessa on tarkka määritelmä sille, milloin jokin tuotteen kehitysjanan kohta on valmis. Tähän aiheeseen paneudutaan raportin luvussa 4.2.4.

4.1.5 Sprintin retrospektiivi

Sprintin retrospektiivi on scrumtiimin sisäinen, muutaman tunnin kestävä tilaisuus. Tämän tilaisuuden tarkoitus on selvittää, miten scrumtiimi toimi kuluneen sprintin aikana ja miten scrumtiimin toimintaa voisi tehostaa tulevia sprinttejä varten. Retrospektiivin jälkeen scrumtiimillä tulisi olla ymmärrys siitä, mitä muutoksia tullaan tekemään ja miten nämä muutokset parantavat scrumtiimin tehokkuutta. Sprintin retrospektiivin aikana voidaan myös muokata ”valmiin” tuotteen kehitysjonon kohdan määritelmää, mikäli aiemmassa määritelmässä on ollut tulkinnanvaraisuuksia tai epäselvyyksiä. (Schwaber & Sutherland 2013, 11–12.)

4.2 Scrumin tuotokset

Scrumin tuotokset tarkoittavat projektin etenemisen seuraamista helpottavia työkaluja. Tuotosten avulla kehitysprosessia pystytään selkeyttämään ja yhdenmukaistamaan. Tämä mahdollistaa useamman kuin yhden kehitystiimin käytön projektissa. (Schwaber & Sutherland 2013, 12.)

4.2.1 Tuotteen kehitysjono

Tuotteen kehitysjono on lista, johon kerätään kaikki ne asiat, jotka valmiissa tuotteessa tulee olla. Lista muuttuu jatkuvasti tarpeiden ja vaatimusten muuttuessa, joten projektin alussa tuotteen kehitysjonoon kerätään lähinnä tärkeysjärjestyksessä korkeimmalla olevat asiat. Tuoteomistaja hallinnoi tuotteen kehitysjonoa ja varmistaa, että se on aina ajan tasalla ja kaikkien scrumtiimin jäsenten saatavilla. (Schwaber & Sutherland 2013, 12.)

Kaikki tuotteen kehitysjonossa listatut kohdat koostuvat tehtävän arvosta, kuvauksesta, tehtävän tärkeysjärjestyksestä sekä tehtävän työmääräarviosta. Tärkeysjärjestyksessä korkeammalla olevat kohdat ovat yleensä tarkemmin kuvailtuja kuin tärkeysjärjestyksessä alemmaksi järjestetyt kohdat. Työmääräarviot eivät ole määrättyjä tuntimääriä, vaan tuotteen kehitysjonon kohtien sisäisiä suhdelukuja. Näiden työmääräarvioiden mukaan voidaan paremmin arvioida, mitkä kaikki kohdat voidaan toteuttaa yhden sprintin aikana. Työmääräarviota tehdessä kehitystiimin historia ja scrumtiimin kokemus ovat korkeassa arvossa. Projektin edetessä tuotteen kehitysjonon kohtia voidaan järjestellä pienemmiksi kokonaisuuksiksi jotta ”valmiin” tuotteen määritelmä on mielekkäämpi saavuttaa. (Schwaber & Sutherland 2013, 12–13.)

4.2.2 Sprintin kehitysjono

Sprintin kehitysjono on läpinäkyvä ja havainnollistava kuvaus niistä asioista, jotka kehitystiimin tulee saada sprintin aikana valmiiksi. Sprintin kehitysjonoon määritellään myös, mikä on sprintin jälkeen saavutettava tuotteen tuoteversio sekä suunnitelma siitä, miten sprintin kehitysjonon kohdat pystytään sprintin aikana toteuttamaan. Sprintin kehitysjono on tuotteen kehitysjonoa yksityiskohtaisempi kuvaus projektia varten toteutettavista asioista. Tuotteen kehitysjonon tavoin myös sprintin kehitysjono elää sprintin aikana. Sprintin kehitysjonon muokkaamisesta vastaa yksin kehitystiimi ja tästä huolehtiminen on scrummasterin vastuu. (Schwaber & Sutherland 2013, 14.)

4.2.3 Tuoteversio

Tuoteversio on sprintin alussa määriteltyjen ja ”valmiiksi” saatujen tuotteen kehitysjonon kohtien arvojen summa. Tuotteen kehitysjonon kohdan tulee olla täysin ”valmis”, jotta arvo voitaisiin laskea kuuluvaksi tuoteversioon. Tämä vaatimus johtuu siitä, että tuoteversion tulee olla

julkaisukelpoinen tuoteomistajan näin halutessa. Tuotteen kehitysjonon kohdan tulee olla myös sellaisenaan käyttöön otettavissa scrumtiimin määritelmien mukaan. Tuoteversion avulla voidaan helposti seurata sprinttien tehokkuuden muutoksia sekä tarkentaa tuotteen kehitysjanan kohtien työmääräarvioita. (Schwaber & Sutherland 2013, 14.)

4.2.4 Valmiin määritelmä

”Valmiin” tuotteen kehitysjonon kohdan määritelmän tulee olla selvä kaikille scrumtiimin jäsenille, vaikka ”valmiin” määritelmä saattaa vaihdella suuresti eri scrumtiimien kesken tulee sen pysyä aina yhtenäisenä yhden scrumtiimin sisällä. Mikäli saman projektin parissa työskentelee useampi kehitystiimi, tulee ”valmiin” kohdan määritelmä tehdä selväksi jo hyvin varhaisessa vaiheessa projektia. Tällaisen menettelyn ansiosta sprintin jälkeinen tuoteversio on aina yhteneväinen ja tuoteomistajan julkaistavissa. (Schwaber & Sutherland 2013, 15.)

4.3 Scrumtiimi

Scrum-kehysrakenteessa scrumtiimin jäsenille on määritelty tarkat roolit. Nämä roolit ovat tuoteomistaja, scrummaster sekä kehitystiimi. Kullekin roolille on määritelty tarkkaan roolin velvollisuudet ja oikeudet scrumtiimin sisäisessä ja ulkoisessa toiminnassa. (Schwaber & Sutherland 2013, 4.)

4.3.1 Tuoteomistaja

Tuoteomistaja on scrumtiimissä se henkilö, joka yksin ylläpitää tuotteen kehitysjonoa. Hän pitää huolen siitä, että tuotteen kehitysjono on ajan tasalla, selkeästi esitetty sekä kaikin ajoin kaikkien scrumtiimin jäsenten saatavilla. Tuoteomistajan vastuulla on myös se, että kaikki scrumtiimin jäsenet ymmärtävät, mitä kukin tuotteen kehitysjonon kohta sisältää, sekä siitä, että kaikki scrumtiimin jäsenet tietävät mikä on tuotteen kehitysjonon ”valmiin” kohdan määritelmä. Tuoteomistaja voi nimittää myös muita scrumtiimin jäseniä tuotteen kehitysjonon ylläpitoon mutta tuoteomistaja kantaa aina vastuun tuotteen kehitysjonon tilasta. (Schwaber & Sutherland 2013, 5.)

4.3.2 Scrummaster

Scrumtiimin menestyksekkyys on suurilta osin scrummasterin vastuulla. Scrummasterin tulee pitää huolta siitä, että tiimi noudattaa Scrumin periaatteita ja sääntöjä. Scrummaster huolehtii myös siitä, että kaikki Scrumiin kuuluvat tapaamiset järjestetään ajallaan ja suoritetaan oikeaoppisesti. Scrummaster pitää huolta myös ulkopuolisten kontaktien ylläpidosta. (Schwaber & Sutherland 2013, 6.)

Scrummasterin tulee olla tarpeeksi ymmärtäväinen Scrumin teoriasta, jotta hän pystyy tarvittaessa opastamaan myös tuoteomistajaa tuotteen

kehitysjonon ylläpitoon liittyvissä ongelmissa. Scrummasterin tulee myös pystyä auttamaan kehitystiimiä tehokkuuden ja paremman sisäisen järjestyksen löytämiseksi. Scrummasterin tehtäviin kuuluu myös auttaa toimeksiantajaa ymmärtämään Scrumin periaatteet sekä edut. (Schwaber & Sutherland 2013, 6–7.)

4.3.3 Kehitystiimi

Kehitystiimi on se Scrumin rooli, joka toteuttaa tuotteen kehitysjonoon määritellyt tehtävät ja saattaa ne ”valmiiksi”. Kehitystiimissä on tärkeää, että sen jäsenillä ei ole erikseen määrättyjä erikoisaloja, vaan kunkin jäsenen tulee pystyä suoriutumaan kaikista tuotteen valmistumista edistävistä tehtävistä. (Schwaber & Sutherland 2013, 5.)

Kehitystiimi toimii itse omana johtajanaan siten, että tuoteomistaja ja scrummaster eivät voi määrätä miten kehitystiimin tulee toimia sprintin aikana saavuttaakseen sprintin suunnittelussa määritellyt tavoitteet. Kehitystiimin sisällä ei myöskään ole erillistä johtajaa, vaan kaikki kehitystiimin jäsenet ovat keskenään yhdenvertaisia. (Schwaber & Sutherland 2013, 5–6.)

Suuri tekijä koko scrumtiimin tehokkuudessa on kehitystiimin koko. Liian pieni kehitystiimi johtaa mahdollisiin viivästyksiin tuotteliaisuuden ja innovaatioköyhyyden johdosta. Mikäli kehitystiimi kasvaa liian suureksi, on vaarana Scrumin periaatteisiin kuuluvan nopean reagointikyvyn menettäminen. Yleisesti hyväksytty kehitystiimin koko vaihtelee kolmen ja kahdeksan hengen välillä projektin koosta riippuen. (Schwaber & Sutherland 2013, 6.)

5 OHJELMISTON TOTEUTUS

Ohjelmiston toteuttamisessa pyrittiin noudattamaan Scrumin määrittelemiä menetelmiä, sekä yleistä hyvää ohjelmointitapaa. Projektin edetessä etenkin Scrumin monet käytännöt todettiin liian kankeiksi opinnäytetyössä toteutettavaan ohjelmistoon. Highsmithin (2000) määrittämien LAD-periaatteiden mukaan saavutettiin kuitenkin käytäntöjä soveltamalla hyvin mielekäs tapa projektin loppuun saattamiseksi.

5.1 Projektin aloitus

Projektin alkupalaverissa valmiille tuotteelle määriteltiin hyvin vapaamuotoiset ja löyhät vaatimukset. Valmiissa tuotteessa tuli olla mahdollisuus lisätä, lukea, muokata ja poistaa tekstiartikkeleita, kuva-albumeita ja yhteystietolistoja. Toteutettavan ohjelmiston kieleksi valittiin yksimielisesti PHP, koska sen opiskeluun ei tarvinnut käyttää aikaa ja PHP-ohjelmistojen ylläpitokustannukset ovat huomattavasti matalammat kuin esimerkiksi ASP.NET-pohjaisten sovellusten. Alkupalaverissa päätettiin myös, että toimeksiantaja tarkastaa projektin etenemisen annettujen määreiden mukaan noin kuukauden välein ja että

toimeksiantaja voi milloin tahansa pyytää sen hetkisen ohjelmistoversion tarkastettavaksi.

5.2 Projektin sovittaminen Scrum-kehysrakenteeseen

Projektille määriteltyjen vaatimusten mukaan listattiin ensimmäinen versio tuotteen kehitysjonosta. Tämä ensimmäinen versio oli hyvin suurpiirteinen ja sisälsi huomattavan vähän tietoja, jotka auttavat määrittelemään yhden sprintin aikana toteutettavia kohtia. Tämä johtui suurimmaksi osaksi opinnäytetyön tekijän vähäisestä ohjelmistotuotannollisesta kokemuksesta johtuvasta heikosta kyvystä arvioida kunkin listatun kohdan työmäärää.

Aluksi suunniteltiin toteutettavaksi kuhunkin vaadittuun ohjelmiston osaan (tekstiartikkelit, kuva-albumit, yhteystiedot) yksi toiminnallisuus (luku, kirjoitus, muokkaus, poisto) yhden sprintin aikana. Tämä metodi oli kuitenkin huono, koska siitä johtuen ohjelmakoodista tuli huomattavan poukkoilevaa, sekavaa ja hankalasti luettavaa. Ensimmäisen sprintin jälkeen päätettiin siirtyä tapaan tehdä yksi ohjelmiston osa täysin valmiiksi ennen siirtymistä ohjelmiston seuraavaan osaan. Työskentelymetodin vaihdoksesta johtuen ohjelmistokoodiin jäi selvä merkki siitä, mitkä kohdat on toteutettu vanhaa metodia käyttäen.

Scrumin periaatteita noudattaen pyrittiin järjestämään sprinttiin liittyvät tapahtumat, jotta pystyttiin paremmin jäsentelemään kaikki vaadittava tieto tulevista ja menneistä sprinteistä. Varsinkin sprinttien suunnittelu oli projektissa tärkeässä osassa. Aluksi sprinttien suunnittelu suunniteltiin toteuttavaksi yhdessä toimeksiantajan kanssa kuukausittaisen etenemistarkastuksen yhteydessä. Tällainen työskentelymalli olisi sallinut vielä paljon vapauksia muokata kuukauden aikana tapahtuvia sprinttejä sen mukaan mitä tiedettiin mahdolliseksi saavuttaa yhden sprintin aikana. Ensimmäisen tarkastuksen jälkeen tultiin kuitenkin siihen tulokseen, että toimeksiantajan läsnäolo suunnittelutilanteissa ei ollut projektin sujuvuuden kannalta oleellista. Myös aikataulujen sovittaminen yhteistä pitempää suunnittelupalaveria varten oli hankalaa. Näistä syistä sovittiin, että opinnäytetyön tekijällä on vapaat kädet suunnitella projektin eteenpäin viemistä.

Tuoteversiota sovellettiin projektiin vasta työn loputtua. Tällöin laskettiin kunkin sprintin jälkeinen tuoteversio projektin alkutilanteessa annettujen arvojen mukaan ja huomattiin joidenkin sprinttien olleen näennäisesti paljon tehokkaampia kuin toisten. Tämä kuitenkin johtui suurimmaksi osaksi siitä, että työn alussa annettiin tuotteen kehitysjonon kohdille vain yksinkertaiset, lineaariset vaativuusarviot. Projektin lopulla pystyttiin tarkentamaan tuoteversioita vertaamalla alkuperäisiä ja toteutuneita arvioita tehtävien vaativuudesta.

6 POHDINTA

Yksin työskennellessä Scrumin kaltainen kehysrakenne projektien toteuttamiseen tuntuu lähes liian monimutkaiselta ja luonnollista kehitystä

hankaloittavalta. Esimerkiksi sprinttien suunnittelu kestoltaan ja tuoteversion määrältään yhdenmukaisiksi koko projektin ajaksi on käytännössä mahdotonta ilman, että tuotteen kehitysjonon kohtia jakaa merkityksettömän pieniksi paloiksi. Tällainen pienempiin osiin paloittelu saattaa kuulostaa hyvinkin toteutuskelpoiselta, mutta projektin jakaminen liian pieniin osiin hankaloittaa Scrumin perusperiaatetta, jonka mukaan jokaisen sprintin jälkeen tuotteesta tulee olla julkaisukelpoinen versio.

Scrumin roolijako on yksin työskennellessä täysin merkityksetön, mutta jo näin pienessä projektissa on helppo kuvitella, kuinka paljon selkeihin rooleihin jaettu ja tehtäviensä tasalla oleva täysi scrumtiimi hyötyy toisistaan. Varsinkin koko projektiin liittyvä suunnittelutyö on varsin työlästä toteuttaa yksin. Omista suunnitelmistaan ja suoritteistaan on myös huomattavasti hankalampi löytää virheitä, kuin muiden ihmisten suunnitelmista ja suoritteista. Useamman hengen tiimissä tämä ongelma poistuu ja tuloksena on huomattavasti yksilötyötä kriittisempi ja tehokkaampi työskentelytapa.

Tuotteen kehitysjono olikin ainoa Scrumin työkalu, joka omassa työskentelyssäni oli käytössä lähes sellaisena, kuin se Scrumin oppimateriaaleissa määritellään. Ongelmana itselläni oli se, etten kokemattomuudestani johtuen osannut määritellä tuotteen kehitysjonoon listattavia asioita tarvittavalla tarkkuudella. Tämä ongelma tulisi todennäköisesti poistumaan, mikäli tuotteen kehitysjonoa ylläpitäisi kokenut tuoteomistaja.

Projektin edetessä opin ymmärtämään paremmin Scrumin eri työkalujen tarkoitusta. Etenkin sprintin kehitysjonot nousivat projektin loppua kohden arvokkaaksi työkaluksi, vaikka olin projektin alkuvaiheessa pitänyt niitä vain suuntaa-antavina merkkipaaluina. Pian projektin aloittamisen jälkeen ymmärsin myös lopettaa niiden työkalujen käytön, jotka eivät nopeuttaneet työskentelyäni vaan saattoivat jopa hidastaa sitä. Näistä esimerkkinä käy hyvin aiemmin mainitsemani toimeksiantajan kanssa tapahtuva sprintin suunnittelupalaveri ja normaalisti tärkeässä roolissa oleva päiväpalaveri. Myös Scrumin roolijaon jouduin unohtamaan, koska yksilötyöskentelyssä on pakko tehdä kaikkien roolien tehtäviä, jotta projekti etenisi.

Yksin toteutettaviin projekteihin Scrum-kehysrakenne ei sellaisenaan tunnu kovinkaan luonnolliselta työskentelymallilta. Kehysrakenteeseen kuuluvat monet palaverit ja tiukka roolijako aiheuttavat lähinnä etenemisen ja luomistyön takeltelua. Näistäkin ongelmista huolimatta on kuitenkin helppo nähdä, miten asiansa osaava ja kokenut scrumtiimi pystyy suoriutumaan projekteistaan huonosti organisoitua tiimiä tehokkaammin ja paremmin lopputuloksin.

LÄHTEET

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W. Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001. Ketterän ohjelmistokehityksen julistus. Suomentanut Lasse Koskela. Viitattu 10.1.2014 http://agilemanifesto.org/iso/fi

Highsmith, J.A. 2000. Adaptive software development: a collaborative approach to managing complex systems. New York: Dorset House.

Larman, C. 2004 Agile and iterative development: a manager's guide. Boston: Addison-Wesley.

Larman, C. & Basili V.R. 2003. Iterative and incremental development: a brief history. Computer 36 (6), 47–56.

Object Management Group, Inc. 2014. Introduction to OMG's Unified Modeling Language™ (UML®). Viitattu 10.4.2014. http://www.omg.org/gettingstarted/what_is_uml.htm

Schwaber, K. & Sutherland, J. 2013. The Scrum Guide™. Viitattu 5.5.2014. https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf