Ohjelmistokehitysprosessin kuvaus

Ville Heikkinen
2006

Opinnäytetyö

Satakunnan Ammattikorkeakoulu – Tekniikka, Pori

Tietoliikennetekniikan koulutusohjelma

TIIVISTELMÄ

OHJELMISTOKEHITYSPROSESSIN KUVAUS

Heikkinen, Ville
Satakunnan ammattikorkeakoulu
Tekniikka Pori
Tietotekniikan koulutusohjelma
Tietoliikennetekniikan suuntautumisvaihtoehto
Joulukuu 2006
Kivi, Karri
UDK: 004.41, 004.42, 658.56
Sivumäärä: 28
Avainsanat: ohjelmistotuotanto, ohjelmointi, tuotekehitys

Opinnäytetyön tarkoituksena oli luoda ohjelmistokehitysympäristö Yoso Oy:lle sisältäen työkulun kuvauksen prosessina, työkalut työn eri vaiheisiin sekä ohjeet työn etenemisestä ja työkalujen käytöstä. Itse prosessi sisältää sekä kohdeprosessien kuvauksen, jossa käytetään BPEL/BPML -notaatiota että järjestelmän määrittelyosion, jossa käytetään UML-notaatiota ja malleja.

Tärkeimpiä tavoitteita oli yhdenmukainen toimintatapa, joka on perustana laadun valvonnalle ja kehittämiselle sekä tehokkuudelle. Tuloksena syntyi määritelty ja uudelleenkäyttöä tukeva sovellusten ja tuotteiden kehityslinja, jota pystytään kehittämään ja laajentamaan tarpeiden mukaan. Tässä projektissa ympäristö luotiin ainoastaan Java-kielelle.

Tarkasti kuvatulla toimintatavalla ja sitä tukevilla työkaluilla pyrittiin entistä nopeampaan ohjelmiston kehitykseen ja vakiomuotoiseen kuvaustapaan ja toiminnan kuvaamisen sekä määrittelyjen että suunnittelun osalta. Ympäristö kehitettiin huomioiden testauksen, kokoonpanohallinnan ja koodin generoinnin lisääminen toimintaan ja prosessiin.

ABSTRACT

DESCRIPTION OF SOFTWARE DEVELOPMENT LINE

Heikkinen, Ville
Satakunta University of applied sciences
School of Engineering and Technology Pori
Degree Programme in Information Technology
Option of Telecommunications Engineering
December 2006
Kivi, Karri
UDC: 004.41, 004.42, 658.56
Number of pages: 28
Key words: software production, programming, product development

The purpose of this work was to create a software development environment to Yoso Oy containing the work process, the tools for the separate stages of the work and instructions for the progress of the work and for the use of tools. The process itself contains both a description of target processes which uses the BPEL/BPML notation and system declaration section which uses the UML-notation and models.

One of the most important objectives was a uniform way of action which is a foundation for the quality control, development and efficiency. The development line of the applications and products which defines and supports the reuse and which it is possible to develop and to extend according to the needs was created as a result. In this project the environment was created only to the Java language.

With the way of action that has been described carefully and with the tools which support it quicker development of the software than before and the standard description way and both for the definitions of the describing of the operation and for the planning were striven for. The environment was developed paying attention to the adding to the operation and process of the testing, assemblage control and generation of code.

SISÄLLYS

1 JOHDANTO

Nykypäivänä ohjelmistotuotannossa uudet teknologiat mahdollistavat lähes kaiken tiedon automaattisen käsittelyn, paikoin jopa oppivan automaattisen toiminnan ansiosta päätösten tekemisen. Valmiit kehittämistuotteet tukevat toimintojen automatisointia myös prosessi-näkökulmasta.

Kuitenkin suurelta osin yritysten toimintamallit, toiminta ja työkalut eivät tue toisiaan. Kaikkialla ei edes olla vastaavaa ongelmaa uskallettu tarkastella. Yksittäinen suuri tekijä tässä ongelmien viidakossa on löytää tarpeeseen sopiva työkalu, joka vielä maksaisi itsensä takaisin edes jollain aikataululla. Räätälöityjen tuotteiden hinnat ovat erittäin kalliita, Pk-yrityksille usein jopa ylitsepääsemättömiä. Ne eivät tunnu istuvan toimintaan, eivätkä yleisestikään uusien työkalujen käyttöönotto suju ilman perinteistä muutoksen epäilyn alaiseksi saattamista, erityisesti tulevan käyttäjäkunnan suunnalta.

Tässä työssä paneudutaan perusongelmaan: mitä liiketoiminta todella tarvitsee tuekseen ja työkaluikseen ja miten ne saadaan nopeasti ja kustannustehokkaasti valmiiksi.
Työkalun eli tietojärjestelmätuotteen kannalta on useita seikkoja, jotka nopeuttavat tuotteen saamista markkinoille ja alentavat merkittävästi kustannuksia. Useimmat teorioista tähtäävät uudelleenkäytettävyyteen tai toimintojen keskittämiseen ja sitä kautta saman asian tekemiseen samoilla tavoin. Tekniseltä kantilta ongelmaa lähestyttäessä löytyy useita arkkitehtuurisia teorioita, joilla ongelmaa on ratkottu.

2 TEORIATAUSTA

2.1 UML

2.1.1 Yleistä

Yhtenäistetty mallinnuskieli UML (Unified Modeling Language) on Grady Booch:n, James Rumbauch:in ja Ivar Jacobssonin vuonna 1995 julkaisema graafinen mallinnuskieli, jolla kuvataan ohjelmistokehityksen eri vaiheita. UML standardoitiin Object Management Groupin (OMG) toimesta vuonna 1997. Tällä hetkellä UML:stä on käytössä versio 1.5 ja 2.0 on tulossa lähitulevaisuudessa.

UML on oliosuuntautunut mallinnuskieli, jonka elementit ja kaaviot perustuvat oliosuuntautuneeseen ajatusmalliin. Oliosuuntautumisella tarkoitetaan tekniikkaa, jolla pyritään tuottamaan luonnollisella tavalla malleja kokonaisuuksista. Nämä mallit ovat helposti selitettävissä, muutettavissa ja laajennettavissa. Oliosuuntauneella mallintamisella luodut mallit ovat helposti toteutettavissa olio-ohjelmointikielillä.

UML:n tavoitteena on pystyä kuvaamaan mikä tahansa järjestelmä oliopohjaisilla kaavioilla. Tällaisia järjestelmiä voi olla esimerkiksi tietojärjestelmät, tekniset järjestelmät ja hajautetut järjestelmät. /1, s. 4-8/

2.1.2 UML ohjelmistokehityksessä

UML voidaan karkeasti jakaa neljään osaan: näkymät, kaaviot, mallinnuselementit ja yleiset merkinnät. /1, s. 11/

2.1.2.1 Näkymät

Näkymät näyttävät mallinnettavasta järjestelmästä eri puolia ja pitävät sisällään useita kaavioita. Järjestelmän kokonaiskuva koostuu useasta näkymästä joista jokainen kuvaa järjestelmä tiettyä ominaisuutta. Näkymien avulla mallinnuskieli yhdistetään kehityksessä käytettyyn työmenetelmään tai toimintaohjeisiin.

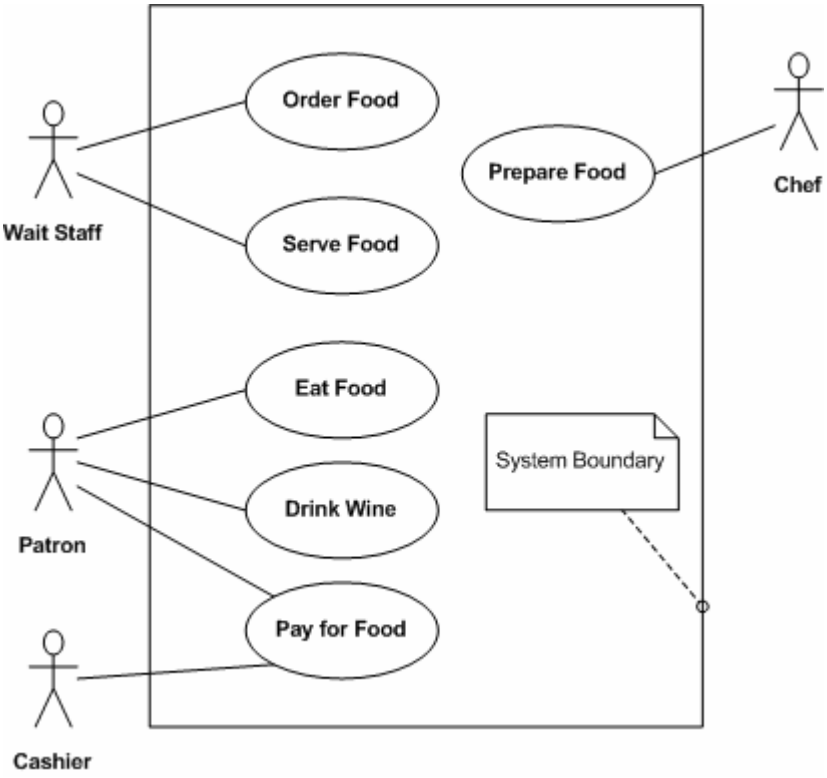
UML:ssä on viisi erilaista näkymää.

- Käyttötapausnäkymä kuvaa järjestelmän toimintaa ulkopuolisen toimijan näkökulmasta.
- Looginen näkymä kuvaa järjestelmän sisäisen toiminnallisuuden pysyvien ja muuttumattomien toimintojen ja rakenteiden kannalta.
- Komponenttinäkymässä kuvataan lähdekoodimoduulien rakennetta.
- Samanaikaisuusnäkymää käytetään kuvattaessa moniajojärjestelmien yhteistyötoiminnan synkronointia ja kommunikaatiota.
- Käyttöönottonäkymä kuvaa järjestelmän käyttöönottoa eri tietokoneissa ja oheislaitteissa. /1, s. 12-14/

2.1.2.2 Kaaviot

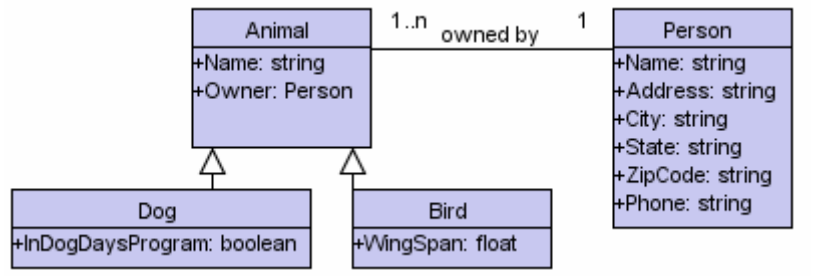
Kaaviot ovat näkymien sisältöä kuvaavia kuvioita. Nykyinen UML:n versio 1.5 pitää sisällään yhdeksän eri tarkoituksiin soveltuvaa kaaviota. Näitä kaavioita yhdistelemällä muodostetaan järjestelmän kokonaiskuva. /1, s. 14-15/

Käyttötapauskaaviossa hahmotellaan ulkoisten toimijoiden ja järjestelmän välisiä yhteyksiä. Yksi käyttötapaus kuvaa yhtä järjestelmässä olevaa toimintoa. Kuvaus tehdään yleensä käyttötapauselementtiin sanallisesti, mutta myös toimintokaavioita voidaan käyttää hyväksi käyttötapausta määriteltäessä. Käyttötapaukset tulee kuvata toimijan näkökulmasta eikä niiden tule puuttua siihen, miten toiminto tullaan toteuttamaan. /1, s. 15/



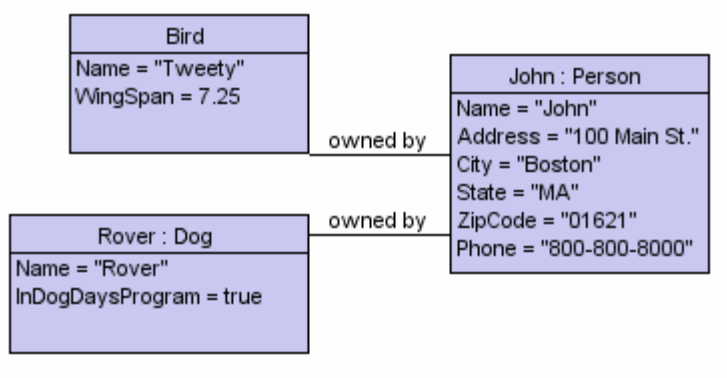
Kuva 1 Käyttötapauskaavio /2/

Luokkakaaviossa kuvataan järjestelmän luokkarakennetta. Kukin luokka vastaa tiettyä järjestelmän käsittelemää asiaa. Luokkien välissä voi olla erilaisia yhteyksiä, joilla kuvataan niiden riippuvuuksia, periytymisiä ja assosiaatioita toisistaan. /1, s. 15/



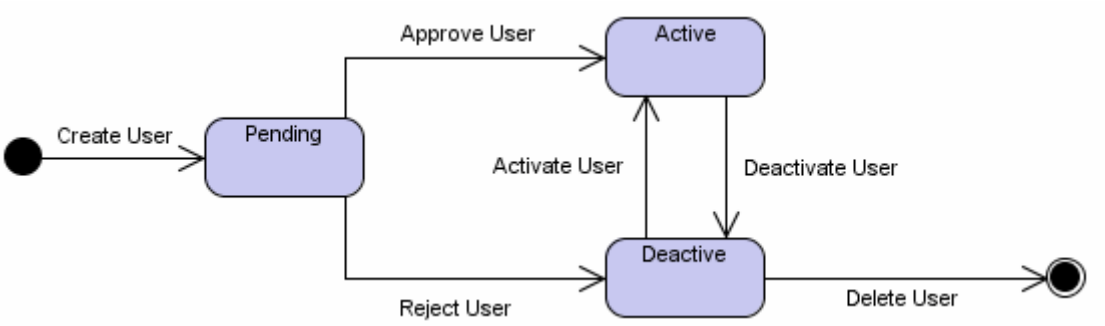
Kuva 2 Luokkakaavio /3/

Oliokaavio on hieman samantyyppinen kuin luokkakaavio. Oliokaaviossa tuotetaan esimerkki luokkakaaviosta. Oliokaaviossa on luokkien ilmentymiä eikä varsinaisesti luokkia. Oliokaavioilla pyritään selkiyttämään monimutkaisia luokkakaavioita. /1, s. 17/



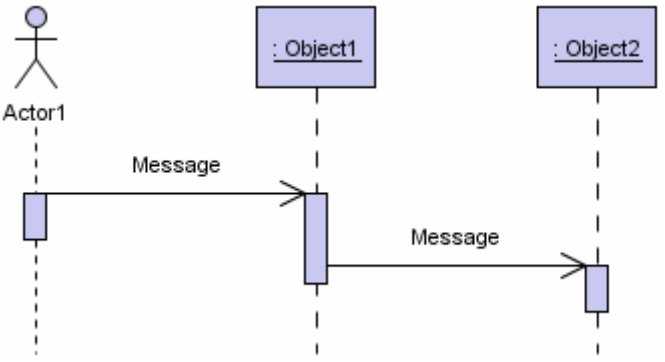
Kuva 3 Oliokaavio /4/

Tilakaaviolla täydennetään yleensä luokan kuvausta. Kaaviossa kuvataan kaikki mahdolliset tilat, joihin oliot voivat joutua sekä mitkä tapahtumat aiheuttavat minkäkin siirtymän. Tilakaaviota ei ole tarpeellista tehdä jokaiselle luokalle vaan ainoastaan siinä tapauksessa, että luokalla on selkeästi useita eri tiloja joiden käyttäytyminen muuttuu tilasta toiseen. /1, s. 17/



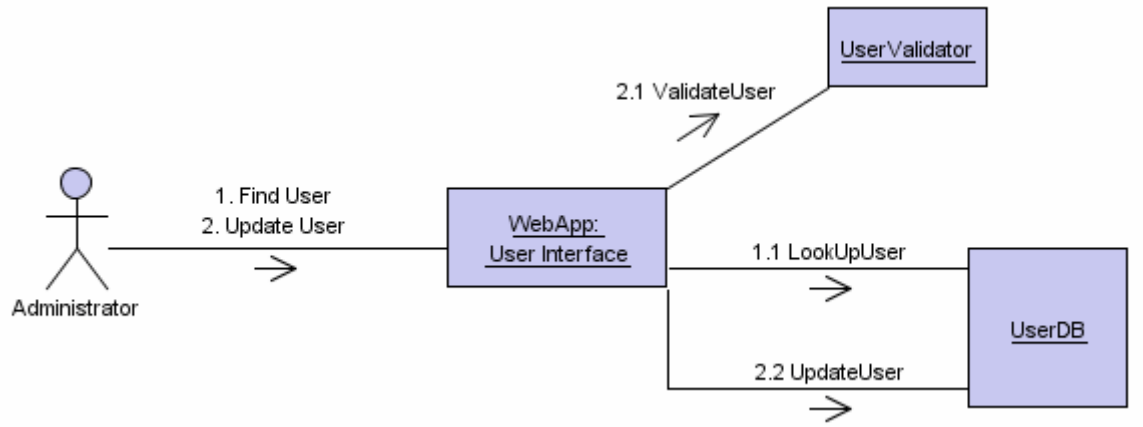
Kuva 4 Tilakaavio /5/

Viestiyhteyskaavion tarkoituksena on havainnollistaa olioiden välistä kommunikaatiota. Kaavio koostuu olioista ja niiden välisistä viivoista, jotka kuvaavat viestiyhteyksiä. /1, s. 18/



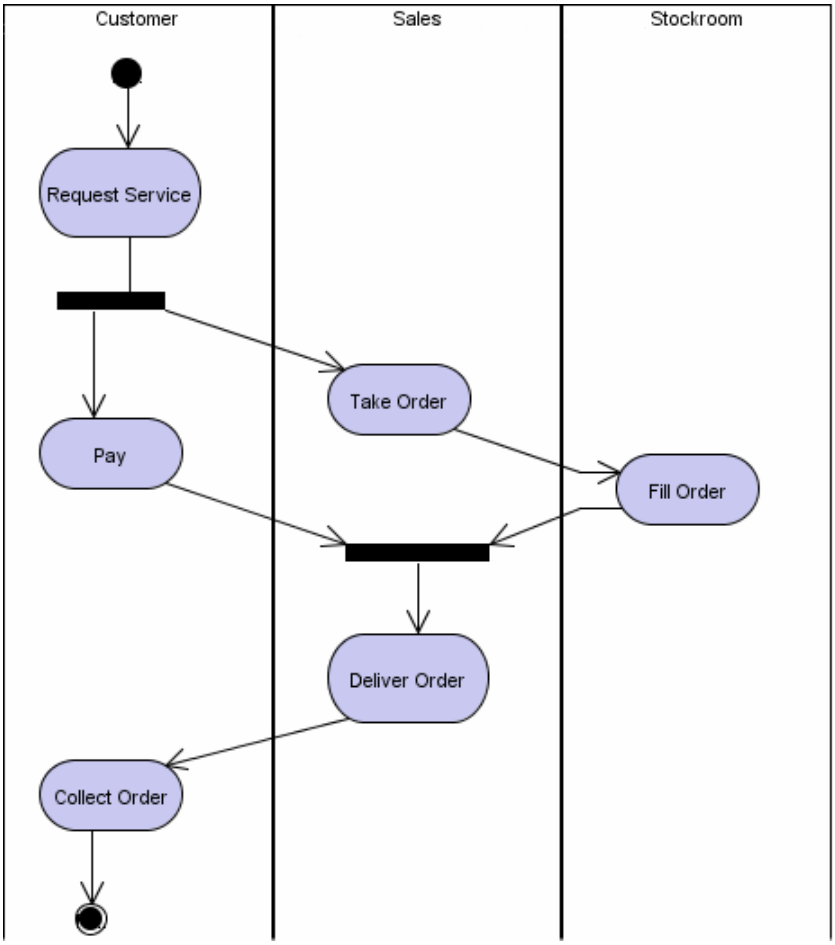
Kuva 5 Viestiyhteyskaavio /6/

Yhteistyökaavio on tarkoitukseltaan samanlainen kuin viestiyhteyskaavio, mutta sillä erotuksella, että viestiyhteyskaaviossa voidaan kuvata myös olioiden suhteet toisiinsa nähden. /1, s. 18-19/



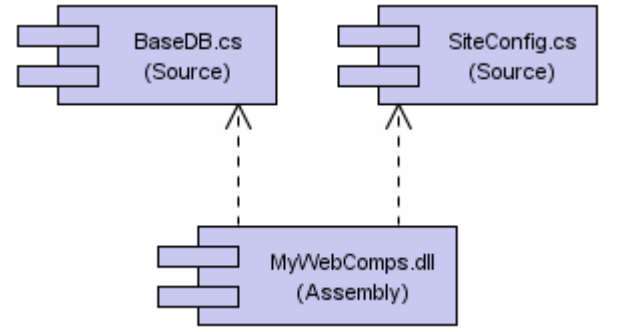
Kuva 6 Yhteistyökaavio /7/

Toimintokaaviolla kuvataan tapahtumien kulkua ajansuhteen. Toimintokaavio koostuu tiloista, jotka sisältävät määritelmän suoritettavasta tapahtumasta. Tapahtumatilasta poistutaan kun tapahtuma on suoritettu toisin kuin tilakaaviossa, joissa siirtymä tapahtuu vasta, kun viesti on vastaanotettu. /1, s. 19-20/



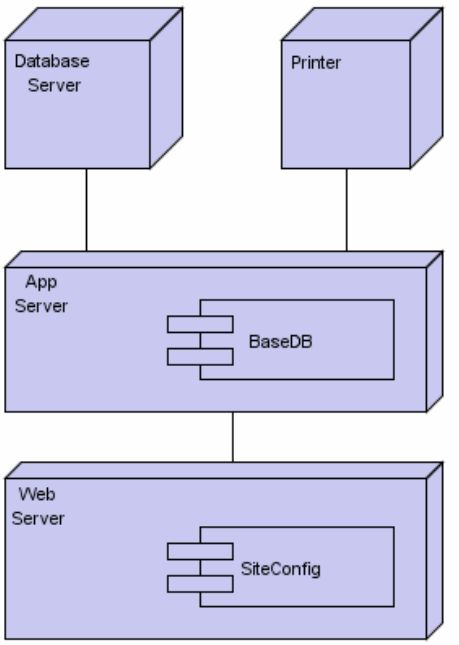
Kuva 7 Toimintokaavio /8/

Komponenttikaavio esittää lähdekoodin rakenteen koodikomponentteina. Komponentit pitävät sisällään tiedot sisältämistään luokista. Komponenttikaavio helpottaa järjestelmän yleisen rakenteen hahmottamista. /1, s. 20/



Kuva 8 Komponenttikaavio /9/

Käyttöönottokaaviolla kuvataan järjestelmän fyysinen arkkitehtuuri. Kaavioon piirretään kaikki järjestelmän tietokoneet ja oheislaitteet yhteyksineen. /1, s. 21/



Kuva 9 Käyttöönottokaavio /10/

2.1.2.3 Mallinnuselementit

Mallinnuselementit kaavioissa vastaavat yleisiä olio-ohjelmoinnin käsitteitä, kuten olioita, luokkia ja viestejä sekä niiden välisiä yhteyksiä. Näitä elementtejä voidaan käyttää erityyppisissä kaavioissa, mutta elementtien merkitys ei muutu. /1, s. 21-23/

Kuva 10 Mallinnuselementit /11/

2.1.2.4 Yleiset merkinnät

Yleisillä merkinnöillä lisätään mallinnuselementteihin lisäinformaatiota kuten kommentteja, ohjeita tai lisämerkityksiä. /1, s. 23-24/

2.2 BPMN

2.2.1 Yleistä

BPMN (Business Process Modeling Notation) on BPMI:n (The Business Process Management Initiative) kehittämä graafinen notaatio, jolla kuvataan liiketoimintaprosessien vaiheet alusta loppuun saakka. Ensimmäinen 1.0 versio julkaistiin toukokuussa 2004.

BPMN:n kehityksen perimmäisenä tarkoituksena on ollut kehittää notaatio, joka on helposti ymmärrettävä ja luettava kaikille sitä käyttäville. BPMN määrittää liiketoimintaprosessikaavion, joka pohjautuu räätälöityyn vuokaaviotekniikkaan, jolla voidaan luoda graafisia malleja liiketoiminnan operaatioista. Liiketoimintaprosessimalli on joukko graafisia komponentteja, jotka kuvaavat eri toimintoja sekä niiden suoritusjärjestystä. [Wh04]

2.2.2 Merkistö

BPMN merkistö jaetaan neljään kategoriaan: flow objects (vuo-objektit), connection objects (yhteysobjektit), swimlanes (uimaradat) ja artifacts (artefaktit).

Flow-objektit jaetaan karkeasti kolmeen ydin ryhmään: Events (tapahtuma), Activity (toiminto) ja Gateway (yhdyskäytävä). /12/

- Event:iä kuvaa ympyrä. Sillä on yleensä syy ja seuraus. Tällaisia ovat alku, välivaihe ja loppu.

Kuva 11 Event /12/]

- Activity kuvataan suorakaiteella, jonka kulmat ovat pyöristetty. Se on jokin työ, mitä yrityksessä tehdään.

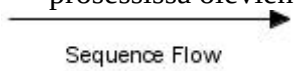
Kuva 12 Activity /12/

- Gateway:n merkki on timantin muotoinen. Ne kuvaavat prosessissa olevia poikkeamia ja yhtymisiä, kuten valintoja, haarautumisia ja yhdistymisiä.

Kuva 13 Gateway /12/

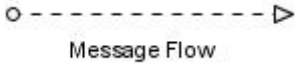
Flow-objektit yhdistetään toisiinsa yhteysobjekteilla. Tällaisia on kolmenlaisia: sequence flow, message flow ja assosiaatio. /12/

- Sequence flow kuvataan yhtenäisellä nuolella. Sequence flow:lla kerrotaan prosessissa olevien aktiviteettien suoritus järjestys.



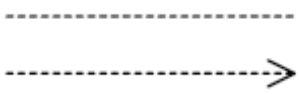
Kuva 14 Sequence flow /12/

- Message flow kuvataan katkoviivanuolella jolla on avoin kärki. Sillä kerrotaan kahden prosessiin osallistuvan toimijan välisestä viestin vaihdosta.



Kuva 15 Message flow /12/

- Assosiaatio kuvataan pisteviivanuolella. Sillä osoitetaan jonkin artefaktin yhteyttä vuo-objektiin.



Kuva 16 Assosiaatio /12/

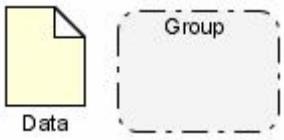
Swimlane on tapa järjestää ja organisoida tapahtumia ja toimintoja visuaalisiksi kokonaisuuksiksi. Swimlane-tyyppisiä objekteja on kahdenlaisia: pool (allas) ja lane (rata). /12/

Pool kuvaa prosessiin osallistuvaa osapuolta. Lane jakaa pool:in pienempiin osiin.

Kuva 17 Pool /12/

Kuva 18 Lane /12/

Artifact:eilla tuodaan prosessiin joustavuutta ja laajennettavuutta. Tällaisia ovat mm. data object ja group. /12/



Kuva 19 Artifact /12/

2.3 Model Driven Development

Malliohjattu kehitys on OMG:n (Object Management Group) kehittämän MDA:n (Model Driven Architecture -standardin) johdosta tullut suosituksi tavaksi kehittää ohjelmistoja. MDA mahdollistaa ohjelmiston suunnittelun ja toteutuksen pelkkien mallien pohjalta.

MDA:n mukaan järjestelmästä toteutetaan ensin täysin alustariippumaton malli. Mallin tulee olla erittäin hyvin määritelty, jotta sitä voidaan ajaa ja validoida. Alustariippumattomasta mallista voidaan automaattisesti generoida alustakohtainen malli, jolla kuvataan järjestelmän toteutus tietyllä alustalla. /13/

2.4 Tuote- ja tuotantolinja

Tuotelinjalla pyritään ohjelmistoarkkitehtuurien laajamittaiseen uudelleenkäyttöön. Tuotelinja-arkkitehtuurin pyrkimyksenä on toteuttaa samankaltaisia ohjelmistoja käyttäen samaa perusrakennetta. Tavoitteena on saavuttaa uusien ohjelmistojen kokoamisen tehostumista ja vähentää suunnittelun ja toteutuksen vaatimaa työtä. Tuotelinja-arkkitehtuurissa käytetään sovelluksissa komponentteja ja erilaisia kehyksiä.

Tuotantolinja on määrämuotoinen tapa kuvata koko ohjelmistokehitysprosessi. Se on kuvaus kaikista ohjelmistokehityksen vaatimista vaiheista. Tuotantolinja voi sisältää useita tuotelinjoja kulkemassa rinnan. Tuotantolinjalla saavutetaan hyötyä ohjelmistojen laadun parantumisena, kehityksen nopeutumisena ja koodin uudelleenkäytettävyytenä. /14/

2.5 Service Oriented Architecture

Service Oriented Architecture (SOA) eli palvelukeskeinen arkkitehtuuri on tapa organisoida uusien ja ennestään olemassa olevien sovellusten rakenne siten, että ne muodostuvat keskenään kommunikoivista palveluista. Palvelut tunnistetaan yleensä liiketoimintaprosessien avulla, mikä kuvastaa SOA:n yhtä päätavoitetta, eli luoda palveluja, jotka muodostuvat olemassa olevista liiketoiminnan kokonaisuuksista. Tällaisia palveluja voisi olla esimerkiksi tilausten tekeminen ja laskuttaminen. /15/

2.6 Design Patterns

2.6.1 Yleistä

Design Pattern:in eli suunnittelumallin määritelmä:

”Suunnittelumalli nimeää, tarjoaa taustatietoa ja selittää käytännössä hyväksi todetun tavan ratkaista jokin järjestelmissä usein esiintyvä arkkitehtuuri- tai suunnitteluongelma. Ratkaisu on yleinen kokoelma olioita ja luokkia, joita sovelletaan ja muovataan ratkaisemaan ongelma erilaisissa käytännön tilanteissa.” /16, s. 6/

Suunnittelumallit ovat siis kokoelmia erilaisia menetelmiä tietynlaisten ongelmien ratkaisuiksi. Mallit sinällään eivät ole koodia, vaan kuvauksia ratkaisuista.

Suunnittelumallien tavoitteena on, ettei kaikkea tarvitsisi keksiä uudelleen, vaan vanhat hyviksi todetut ratkaisut voidaan hyödyntää. /17/

2.6.2 Luokittelu

Suunnittelumallit luokitellaan yleisesti tarkoituksen ja ympäristön mukaan. Käyttötarkoituksen mukaan mallit voidaan jakaa vielä kolmeen luokkaan:

Toiminnalliset-/käyttäytymismallit (behavioral patterns): Toiminnalliset luokkamallit käyttävät perintää jakamaan toiminnot eri luokkiin. Toiminnallisten oliomallien perusmenetelmänä on sitä vastoin olioiden yhdistäminen. Ne auttavat ratkaisemaan esimerkiksi seuraavanlaisia ongelmia: Kuinka olion tila voidaan tallentaa ja palauttaa tarvittaessa myöhemmin tai kuinka välittää tietoja muille olioille ilman, että oliot tietävät toistensa luokkia. /16, s. 6/

Luontimallit (creational patterns): Luontimallit auttavat ratkaisemaan mm. seuraavanlaisia käytännön ongelmia: Kuinka luoda olioita ympäristössä, jossa liittymänä on joukko tapauskohtaisia abstrakteja luokkia tai kuinka varmistetaan, että luokalla on vain yksi ilmentymä, joka on kaikkien muitten luokkien saatavissa. /16, s. 6/

Rakennemallit (structural patterns): Rakennemalleilla kuvataan olioiden välisiä suhteita ja sitä, kuinka olioita yhdistelemällä saadaan uusia toimintoja. Ne auttavat ratkaisemaan esimerkiksi muistinkäytön vähentämiseen tai tehokkuuden lisäämiseen liittyviä ongelmia, kun joudutaan luomaan miljoonia olioita tai kuinka voidaan sovittaa yhteen kaksi yhteensopimatonta järjestelmää. /16, s. 6-7/

Suunnittelumalliin voidaan ajatella kuuluvan neljä osaa:

- Mallin nimi kuvaa muutamalla sanalla ongelmaa, ratkaisua ja seurauksia. Hyvin valittu nimi helpottaa mallin hyödyntämistä jatkossa.
- Ongelma. Kappaleessa kuvaillaan kohteena oleva ongelma, sen tyypillisiä ilmenemisalueita sekä sen perinteisiä ratkaisutapoja. Joskus ongelma voi sisältää listan ehtoja, joiden tulee täyttyä, ennen kuin on järkevää käyttää mallia.
- Ratkaisu. Kappale kuvailee ratkaisun rakenteen jollakin yleisellä esitystavalla, yleensä olioperusteisin menetelmin sanallisesti tarkennettuina UML-kaavioina.
- Seuraukset. Kappaleessa analysoidaan mallin hyviä ja huonoja puolia sekä esitetään tilanteita, joihin malli ei sovellu. /16, s. 7-8/

2.7 Ketterät menetelmät

Ketterät menetelmät perustuvat ajatukseen, että koska vaatimusten muuttuminen kehityksen aikana on todennäköistä, ei ole kannattavaa suunnitella etukäteen liian raskasta mallia. Menetelmiä on useita, mutta niille kaikille on yhteistä toimivan

ohjelmiston ensisijaisuus, suora viestintä eri osapuolten välillä ja nopea muutoksiin reagointi.

Useimmiten ketterät menetelmät pyrkivät jakamaan ohjelmistokehityksen lyhyisiin iteraatiokierroksiin ja näin minimoimaan riskejä. Iteraatiot sisältävät projektisuunnittelun, vaatimusanalyysin, ohjelmistosuunnittelun, koodauksen, testauksen ja dokumentoinnin, joita tarvitaan uusien toimintojen julkaisemiseen. Tällä pyritään saamaan julkaisemiskelpoinen versio ohjelmistosta jokaisen iteraatiokierroksen jälkeen. /18/

3 PROJEKTIN TARKOITUS

Yoso Oy on perustettu vuoden 2005 marraskuussa ja silloin yrityksessä työskenteli yksi työntekijä, mutta jo muutaman kuukauden kuluttua kaikki kolme omistajaa olivat siirtyneet kokopäiväisesti Yoson palvelukseen. Tällöin tehtiin strateginen päätös kuvata Yoson toiminta ja ohjeistaa se. Toiminnan kuvaaminen tehtiin, jotta yrityksen oman toiminnan tehokkuutta pystyttiin tarkastelemaan kriittisesti. Yoson yhtenä päätoimialueena on ict-konsultointi, jonka alaisuudessa toteutetaan tuotekehitystä usealla eri toimialalla. Tämä projekti keskittyy erityisesti tuotekehityksen toimintojen kuvaamiseen ja myös niiden kehittämiseen.

Lähtötilanteessa kuvattiin projektin vaatimukset:

- Nopea ja ohjeistettu prosessi
- Toimintaa tukevat työkalut prosessin eri vaiheisiin
- Työkalujen integroiduttava keskenään prosessin etenemisen mukaan
- Iterointi oltava mahdollista ja työkalujen tuettava sitä
- Käytetään yleisiä standardeja ja avointa lähdekoodia mahdollisimman laajasti
- Kuvaukset ja ohjeet voitava julkaista sähköisesti yrityksen extranetissä

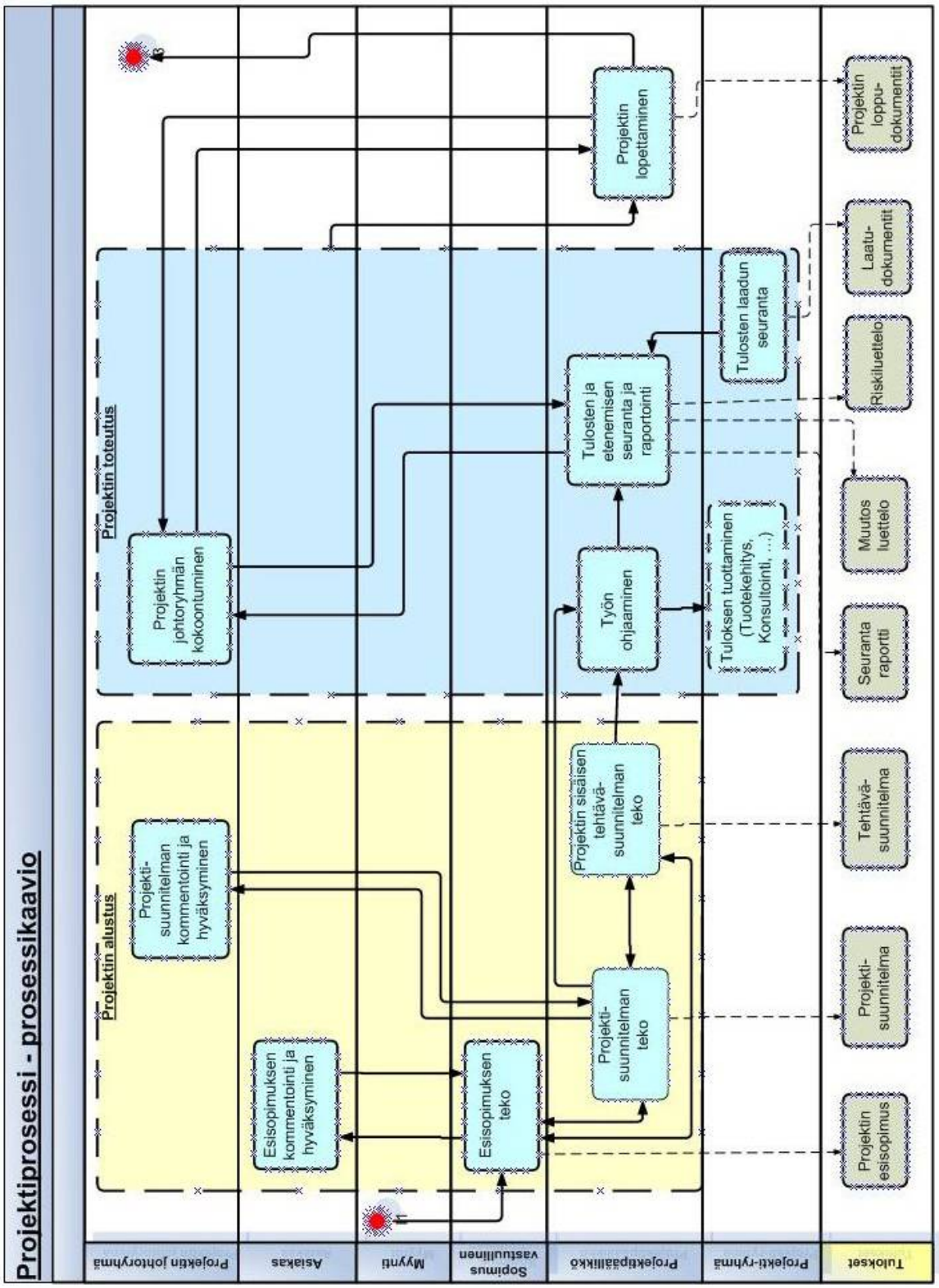
Yosolla oli ennestään kokemusta vastaavista ohjelmistokehitysprosesseista ja vaatimuksia kerättäessä otettiin huomioon nämä kokemukset ja visiot paremmasta tavasta toimia. Projekti lähti pääasiassa hakemaan ratkaisuja edellä mainittuihin vaatimuksiin ja projektin sovittiin olevan samalla Ville Heikkisen opinnäytetyö insinööritutkintoa varten. Projektissa saavutettuja tuloksia lähdettiin testaamaan ohjelmistoprojektissa, joka tuli mukaan projektin myöhemmässä vaiheessa.

4 TUTKIMUS- JA PROJEKTIMENETELMÄT

Opinnäytetyö päätettiin toteuttaa projektimuotoisena. Projekti päätettiin viedä läpi noudattamalla hieman muokattua versiota Yoso Oy:n projektiprosessista. Projektiprosessiin on kuvattu kaikki prosessin eri vaiheet helpottamaan sen hallintaa ja läpi viemistä.

Projekti aloitettiin alustavalla suunnittelulla ja sopimuksen tekemisellä kaikken osapuolien välille. Seuraavaksi seurasi vaatimusten kartoittaminen, josta siirryttiin itse

prosessin kuvaamiseen ja työkalujen istuttamiseen. Myöhemmin mukaan liitettiin myös muutoksena pilottihanke.



Kuva 20 Projektiprosessi /19/

Yoson projektiprosessi yksityiskohtineen keskittyy käytännössä projektin etukäteiseen suunnitteluun, ja suunnitelman tiukkaan seurantaan. Suunnitelmasta poikkeamat hoidetaan omassa osaprosessissa ja suunnitelmaa päivitetään sen mukaan. Tarkempi kuvaus prosessista kuuluu salassapidon piiriin.

4.1 Projektityökalut

Mallipohjat projektin eri dokumenteista (projektisuunnitelma, muutostenhallinta, riskienhallinta) helpottavat projektin hallinnollisten dokumenttien laatimista ja yhdenmukaistaa dokumentaatiota.

Projektinhallintaan järjestettiin tutori ohjaamaan projektin hallinnoimista ja oikeaoppista läpivientiä. Tutori oli tarpeen, koska projektipäälliköllä ei ollut aiempaa kokemusta projekteista. Tutoriksi määrättiin Olli Niemi Yoso Oy:stä.

Varsinaisina työkaluina käytettiin Microsoftin toimisto-ohjelmistoista Word -tekstinkäsittelyohjelmaa ja Excel -taulukkolaskentaohjelmaa.

Esisopimus vastaa tässä projektissa opinnäytetyön aloitussopimusta joka tehtiin Satakunnan Ammattikorkeakoulun, Yoso Oy:n ja Ville Heikkisen välillä.

Projektisuunnitelmassa kerrotaan tärkeimmät projektia koskevat tiedot. Se sisältää seuraavat osiot: 1) toimeksianto ja tavoitteiden täsmennys, 2) projektin vaiheistus (päävaiheet ja ensimmäisen vaiheen tarkennetut osatehtävät) 3) aikataulu ja tarkistuspisteet (väliraporttien aiheet) 4) projektin organisaatio.

Tehtäväsuunnitelmassa määritetään tehtävät ja aikataulu tarkemmin. Aikataulutuksella pyrittiin saamaan projekti valmiiksi ennalta sovitun ajan kuluessa sekä helpottamaan sen etenemisen seuraamista. Tehtäväsuunnitelmaa ja aikataulua tarkennettiin jatkuvasti projektin edetessä. Projektisuunnitelman mukaisesti pienet aikataulumuutokset eivät vaatineet ohjausryhmän hyväksyntää.

Riskiluettelossa listataan mahdolliset riskit, jotka voivat vaikuttaa projektin kulkuun epäsuotuisasti. Tässä projektissa riskien lukumäärä ja niiden todennäköisyys olivat vähäisiä, koska niiden vaikutusten minimoimiseen pyrittiin jo alusta lähtien.

Ohjausryhmä kokoontuu määrätyin väliajoin ja käsittelee kokouksissa projektiin liittyviä asioita kuten sen etenemistä ja muutosten hyväksymiset. Ohjausryhmän tehtävänä oli myös ohjata ja opastaa projektin tekijää projektin läpiviemisessä.

Edistymisraportilla informoidaan projektin etenemisestä ohjausryhmälle. Se pitää sisällään mm. projektin tilanteen, työmäärien seurannan, tehdyt työt ja seuraavalle jaksolle suunnitellut työt.

Muutostenhallinta-dokumentissa listataan projektissa tapahtuneet ja siihen olennaisesti vaikuttaneet muutokset ja niiden vaikutukset aikatauluun. Muutokset tuli hyväksyttää ohjausryhmällä.

4.2 Prosessin kuvaaminen

Projekti päätyi käyttämään kuvausten tekemiseen standardia BPMN-notaatiota. Piirtämistyökaluna käytettiin Microsoft Visio -ohjelmistoa. Notaationa välineessä käytettiin internetistä löytynyttä open source -merkistöä. Prosessin suunnittelemiseen ja toteuttamiseen tarvittavat tiedot kerättiin haastattelemalla Yoso Oy:n henkilökuntaa ja pitämällä workshopeja. Haastattelemalla saaduista tiedoista lähdettiin mallintamaan prosessia. Prosessin kuvaus annettiin välillä henkilökunnan kommentoivaksi, jonka jälkeen siihen tehtiin tarvittavia parannuksia. Workshopien jälkeen projektissa kuvattiin ja hyväksyttiin tulokset.

4.3 Ohjelmistojen evaluointi

Ohjelmistojen evaluointi aloitettiin listaamalla tarvittavat ominaisuudet, joita ohjelmistojen tulisi pitää sisällään. Lista tehtiin haastattelujen pohjalta.

Listan pohjalta kartoitettiin olemassa olevien ohjelmistojen vaihtoehdot. Kartoittamista suoritettiin haastattelemalla suhdeverkostoa, käyttämällä internetin hakupalveluja sekä keskustelufoorumeja. Ohjelmista laadittiin ominaisuustaulukko jota verrattiin haluttujen ominaisuuksien listaan.

Parhaat vaihtoehdot valittiin tarkempaan tarkasteluun. Ohjelmat asennettiin tuotantoympäristöön ja tutkittiin niiden soveltumista tuotantolinjan vaatimuksiin.

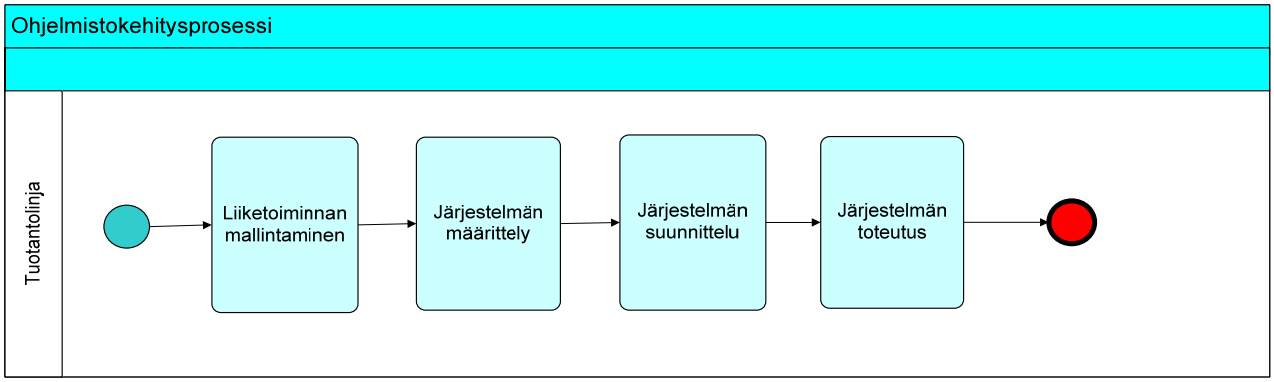
4.4 Prosessin testaaminen

Prosessin toimivuutta haluttiin testata toiminnassa, joten yhden Yoso Oy:n ohjelmistohankkeen kuvaukset otettiin pilotti-projektiksi. Ohjelmistoa lähdettiin kehittämään prosessia seuraten ja sitä täydentäen. Testaamisen aikana tulleet kehitysehdotukset kirjattiin ylös, jotta niistä voidaan tehdä jatkokehitysprojekteja.

5 TULOKSET

5.1 Prosessi

Tuloksena syntyi BPMN-notaatiolla piirretty ohjelmistokehitysprosessi, josta karkea versio alempana. Alkuperäinen versio kuuluu salassapitosopimuksen piiriin.



Kuva 21 Prosessin karkea kuva

5.2 Työkalut

5.2.1 Projektin hallinnolliset dokumentit

Projektin hallinnolliset dokumentit syntyivät projektin kuluessa ja pitävät sisällään kaikki projektin tiedot. Alempana on lista näistä dokumenteista.

- Projektisuunnitelma
- Tehtävälista
- Tilanneraportit
- Riskienhallinta
- Muutostenhallinta

5.2.2 Kehitystyökalun evaluointi

Työkaluja evaluoitaessa tehtiin haettavasta sovelluksesta vaatimuslista ja löydetyistä sovelluksista laadittiin ominaisuuslista.

Kehitys-sovelluksen vaatimuslista

- Open source
- UML
- Java
- Koodin generointi
- Tuki muille kielille
- Jatkuva kehitys
- Hyvät tukipalvelut

Ominaisuuslista

Ominaisuus	Sovellus1	Sovellus2	Sovellus3
Open source	Kyllä	Ei	Kyllä
Ilmainen	Ei	Kyllä	Kyllä
UML	Kyllä	Kyllä	Ei
Java	Kyllä	Kyllä	Kyllä
Koodin generointi	Kyllä*	Kyllä	Ei
Tuki muille kielille	Kyllä	Ei	Kyllä
Jatkuva kehitys	Kyllä	Kyllä	Ei
Hyvät tukipalvelut	Kyllä	Kyllä	Ei
	*Generointi vain Javalle		

5.2.3 Kehitystyökalun ohjeet

5.2.3.1 Asennusohjeet

Asennusohjeet tehtiin helpottamaan kehitysympäristön asentamista toimintakuntoon. Ohjeet sisälsivät tarkan kuvauksen itse sovelluksen sekä sen tarvitsemien muiden komponenttien asennuksen eri vaiheista. Ohjeistus pyrittiin tekemään sellaisella tarkkuudella, että kuka tahansa pystyisi ohjeiden avulla asentamaan kehitysympäristön toimintakuntoon ilman ulkopuolisen apua. Ohjeissa käytettiin paljon kuvakaappauksia havainnollistamisen tehostamiseen.

5.2.3.2 Käyttäjän ohjeet

Käyttäjän ohjeet sisältävät kehitysympäristön käyttöä helpottavia ohjeita ja sääntöjä, joilla saadaan aikaan yhdenmukainen tapa sovellusten kehittämiseen. Se kertoo mm. hakemistorakenteen, tiedostojen nimeämismallin sekä koodin kommentoimiseen liittyvät asiat. Yhdenmukaiset säännöt helpottavat ohjelmistokehittäjien työtä sekä mahdollistavat työn siirrettävyyden helpottumisen toiselle kehittäjälle.

5.3 Jäsenrekisterisovelluksen kuvaukset

Jäsenrekisteristä tuloksina opinnäytetyöhön syntyi alla olevat kaaviot. Ne luotiin prosessin avulla. Dokumentit kuuluvat salassapitosopimuksen alaisuuteen.

- Luokkamalli
- Tietomalli
- Käyttötapauskaavio

6 TULOSTEN TARKASTELU JA OMAT KOKEMUKSET

6.1 Projekti

Opinnäytteen tekeminen lähti liikkeelle aiheen hankkimisesta, joka sitten löytyi Yoso Oy:stä. Keskusteluissa selvisi, että Yoso tarvitsi tarkasti kuvatun ohjelmistokehitysprosessin, jota sitten lähdimme kuvaamaan. Sovittiin, että opinnäytetyö tehdään projektimuotoisena, jotta sen seuraaminen ja hallitseminen olisi helpompaa. Projektimuotoinen toiminta ei ollut työn tekijälle tuttua, joten Yoso järjesti projektityöhön liittyvää opastusta ja koulutusta. Yosolla oli olemassa valmis projektiprosessi jonka mukaisesti projektia lähdettiin viemään eteenpäin.

Alussa projektista tehtiin tarvittavat sopimukset osapuolien välille sekä projektisuunnitelma. Projektisuunnitelman pohjalta tehtiin tehtävälista, joka piti sitten aikatauluttaa. Aikatauluttaminen tuotti suuria vaikeuksia kokemattomuuden vuoksi, joten se vaati jatkuvaa päivittämistä projektin edetessä. Tehtäviin tarvittava aika tuli helposti arvioitua liian lyhyeksi ja käytettävissä olevat resurssit liian suuriksi. Aikataulutusvaikeudet johtuivat kokemattomuuden lisäksi myös projektipäällikön opintojen keskeneräisyydestä. Koulu ja kokeet veivät enemmän aikaa, kuin osasi arvioida. Projektista tehtiin myös riskienhallintasuunnitelma, jonka merkitys tässä kyseisessä projektissa ei ollut oleellinen.

Koska projekti käsitteli ohjelmistosuunnittelua, eikä projektin tekijällä ollut siitäkään aikaisempaa kokemusta tai koulutusta, tuntui projektin valmiiksi saattaminen alkuun mahdottomalta urakalta. Kasvava kiinnostus aiheeseen ja projektin mahdollistavat jatkoprojektit innostivat hakemaan tietoa ja opettelemaan ohjelmistosuunnittelun alkeet. Yoso Oy:n henkilökunta opasti ja helpotti tätä urakkaa, koska heillä oli jo vankkaa asiantuntemusta ja työkokemusta alalta.

Projektin etenemistä seurattiin kuukausittaisissa ohjausryhmän kokouksissa sekä edistymisraporteilla. Kokouksissa käytiin läpi mitä oli saatu aikaan ja mitä kohtia pitäisi vielä hioa sekä seuraavan kuukauden tehtävät. Kokoukset olivat hyödyllisiä tekijälle, koska siellä sai välitöntä palautetta tuloksista, varsinkin sinä aikana, kun tekijä opiskeli eri paikkakunnalla kuin työnantaja sijaitsi. Yhteyttä pyrittiin pitämään myös muuna aikana sähköpostilla, Skypellä ja muilla sähköisillä välineillä, ne eivät kuitenkaan korvaa täysin kasvokkain tapahtuvaa asioiden käsittely.

Projektiin tuli muutama suurempi muutos, jotka osaltaan sekoitti hieman aikataulua ja toi uusia haasteita tekijälle. Muutokset kirjattiin ylös muutostenhallintadokumenttiin. Yoso oli hankkimassa käyttöönsä yhtiön ulkopuolelta sovelluskehystä ja sen keskeneräisyydestä johtuen prosessi rajattiin tässä vaiheessa käsittämään vain kaksi ensimmäistä vaihetta. Toinen iso muutos oli jäsenrekisterisovelluksen kuvauksien tekemisen ottamisen mukaan prosessin testaamiseksi.

Projektin hallinnointi paljastui hyvin paljon aikaa vieväksi toiminnaksi. Raporttien kirjoittaminen, aikataulun ja tehtävien päivittäminen kulutti aikaa enemmän kuin etukäteen osattiin arvioida. Tämä johti siihen, että dokumenttien päivittäminen alkoi projektin edetessä laahata perässä, mikä johti sitten myöhemmässä vaiheessa hankaluuksiin. Kaikkia asioita, mitkä olivat muuttuneet, ei enää muistanut kun päivittäminen tuli ajankohtaiseksi. Jos päivittämiseen olisi varattu enemmän aikaa ja tehnyt sen säännöllisesti, olisivat työhön käytettävät resurssit pysyneet pienempinä ja dokumentit oikeellisina.

6.2 Prosessi

Projektin yksi tärkeimmistä tuloksista oli itse ohjelmistokehitysprosessi. Tuloksena syntyi nopea ja tarkka kuvaus siitä, kuinka Yoso Oy:ssä aletaan ohjelmistoja kehittää niin, että se olisi mahdollisimman tehokasta, nopeaa ja kilpailukykyistä. Prosessilla mahdollistetaan sovelluksen saamisen nopeasti testattavaan versioon. Muista projekteista on helppo tuoda valmista koodia, jolloin vältytään osien uudelleen kirjoittamiselta. Prosessin tuli olla myös kevyt, ettei sen seuraaminen vaadi turhaa resurssien haaskaamista ja tämä tavoita saavutettiin.

Prosessia lähdettiin piirtämään käyttäen standardia BPMN-merkistöä. BPMN:n hyödyt ovat luettavuudessa ja sillä saadaan aikaan yhdenmukaisia ja uudelleen käytettäviä tuloksia. Piirtäminen eteni hitaasti, koska ohjelmistokehitykseen liittyy useita asioita ja niiden selvittäminen oli aikaa vievää sellaiselle, joka oli ensimmäistä kertaa asian kanssa tekemisissä. Pääasiallisena tietolähteenä toimi Yoso:n oma henkilökunta, joilla oli näkemys siitä, mitä kaikkea prosessissa tulee olla. Haastattelemalla heitä, lukemalla alan kirjallisuutta ja käyttämällä Internetiä tietoa alkoi kasaantua riittävästi, että sai karkean näkemyksen siitä miten edetä. Prosessia kehitettäessä pyrittiin ottaa huomioon parhaat puolet olemassa olevista kehitysmalleista ja tekniikoista ja yhdistämään ne toimivaksi kokonaisuudeksi. Näistä on kerrottu tarkemmin kappaleessa kaksi. Ensimmäinen tavoite oli saada listattua kaikki prosessin tehtävät ja prosessiin osallistuvat osapuolet. Tämän jälkeen prosessista valmistui ensimmäinen karkea versio, jossa oli piirrettynä osapuolet ja heille osoitetut tehtävät. Tämä versio kävi jokaisella Yoso:n työntekijällä arvioitavana ja kommentoitavana. Saadusta palautteesta syntyi seuraava versio, jossa tehtävistä oli muodostettu vuokaavio. Tätä versiota paranneltiin kunnes päästiin tyydyttävään tulokseen.

Prosessi on jaettu neljään osa-alueeseen, joista opinnäytetyössä tehtiin kaksi ensimmäistä. Liiketoiminnan mallintamisen tavoite on selvittää organisaation ydinprosessit, niihin liittyvät roolit, prosessin vaiheissa tarvittavat keskeiset tiedot (tietovarastot) sekä samalla kirjata ylös kehittämistarpeet ja ongelmat. Tuloksena syntyvät kuvaukset organisaation prosesseista. Järjestelmän määrittelemisellä pyritään UML-malleja käyttäen kuvaamaan kaikki järjestelmän toteuttamat toiminnot ja liitännät järjestelmän ulkopuolelle. Vaiheen tuotoksena syntyvä määrittelydokumentti kuvaa siis mitä kaikkea järjestelmällä voi tehdä sekä miten käyttäjä voi ne tehdä. Järjestelmä ei kuvaa sitä, miten toiminnot tulee toteuttaa. Järjestelmän määrittely tehdään tarkasti ja huolellisesti, jotta saadaan työkalulla

generoitua mahdollisimman valmista koodia. Ideaalitapauksessa generoimalla saadaan tehtyä kaikki muu paitsi metodien toiminnallisuus, joka jää sitten itse ohjelmoijan tehtäväksi. Kaksi muuta prosessin vaihetta eli järjestelmän suunnittelu ja toteutus toteutetaan jatkokehitysprojekteissa. Ne tulevat pitämään sisällään mm. sovellusarkkitehtuurin valinnan, tietomallin suunnittelun, koodin generoinnin, itse ohjelmoinnin ja testauksen.

6.3 Sovellukset

Prosessin tarpeisiin etsittiin oikeat työkalut jokaiseen tehtävään ja yksi projektin päätavoitteista oli löytää näihin tehtäviin open source -tuotteet, joka teki tehtävästä haastavan. Työkalujen evaluointiprosessi oli melko työläs, sillä tietoa ja sovelluksia ei ollut paljon markkinoilla. Haussa oli sovellus, jossa olisi kaikki halutut ominaisuudet samassa paketissa. Vaatimukset rajasivat pois kaikki pienet ja kehityksen alkuvaiheessa olevat open source -ratkaisut.

Hakeminen lähti liikkeelle tuotteiden kartoittamisella. Tätä tehtiin puhtaasti internetiä käyttämällä. Hakukoneiden ja keskustelupalstojen avulla löytyi muutamia potentiaalisia vaihtoehtoja. Näistä kerättiin ominaisuuslista, jonka pohjalta tehtiin ensimmäinen karsinta. Jäljelle jääneitä tuotteita kokeiltiin käytössä ja kirjattiin ylös positiiviset ja negatiiviset kokemukset. Loppujen lopuksi vain yksi sovellus erottui muista edukseen ja tuli valituksi. Tämä sovellus piti sisällään kaikki halutut ominaisuudet, paitsi että se oli maksullinen. Hinta oli kuitenkin nimellinen eikä muodostunut ongelmaksi.

Työkalua päästiin testaamaan käytännössä prosessin testausvaiheessa.

6.4 Prosessin testaus

Prosessin testaaminen tuli opinnäytteeseen mukaan muutoksena. Yososssa päätettiin toteuttaa jäsenrekisteriohjelmiston kehittäminen ja se oli sopiva testiprojekti prosessille. Tästä projektista opinnäytteeseen kuului ainoastaan ohjelmiston määritykset ja muut tehtävät rajattiin opinnäytteen ulkopuolelle.

Jäsenrekisterin suunnittelu toteutettiin prosessin määrittämiä tehtäviä seuraamalla. Osa tehtävistä ohitettiin, koska ne eivät olleet oleellisia tässä projektissa. Jäsenrekisterin suunnittelemisessa käytettiin geneeristä organisaatiorakennetta. Rakenteen avulla saatiin aikaiseksi jäsenrekisterin mahdollinen käyttö muiden projektien osana. Suunnittelu aloitettiin käyttötapauksien kuvaamisella jonka tuloksena saatiin käyttötapauskaavio. Käyttötapauksien avulla muodostettiin liiketoimintaluokkamalli, johon kuvattiin luokat, metodit, attribuutit ja niiden suhteet. Tässä tapauksessa kokeiltiin prosessin joustavuutta jättämällä viestiyhteyskaavioiden luonti pois, mikä kostautui toteutusvaiheessa. Viestiyhteyskaavioiden tekeminen olisi helpottanut koodin kirjoittamista. Näiden kaavioiden tekeminen jälkikäteen toi turhaa lisätyötä, mikä olisi ollut estettävissä seuraamalla tarkemmin prosessia.

Työkalut, osana prosessia, teki työstä helppoa ja mallintaminen oli nopeaa. UML-mallit oli helppo siirtää muihin välineisiin, jos tarvetta oli. Jatkokehittämiseen on kuitenkin vielä tarvetta versionhallinnan ja dokumenttienhallinnan osalta, joiden eteen Yososssa on jo lähdetty tekemään työtä.

Prosessi osoittautui toimivaksi ja vaatimukset täyttäväksi kokonaisuudeksi. Sen avulla saa aikaiseksi hyvin ja tarkkaan määriteltyjä ohjelmistoja nopeasti. Prosessi otetaan käyttöön myös muissa Yoson projekteissa ja sen kehitys jatkuu.

7 LÄHDELUETTELO

1. Erikson, H-E & Penker, M. UML, Helsinki: Oy Edita Ab, 2000, 339 s.

2. Wikipedia. Use case diagram [online]. [Viitattu 10.10.2006]. Sivujen toteutus: Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Use_case_diagram/

3. .netCoders. Guide to UML Diagrams. Class diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/classdiagram.aspx

4. .netCoders. Guide to UML Diagrams. Object diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/objectdiagram.aspx

5. .netCoders. Guide to UML Diagrams. Statecharts diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/statechart.aspx

6. .netCoders. Guide to UML Diagrams. Sequence diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/sequence.aspx

7. .netCoders. Guide to UML Diagrams. Collaboration diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/collaboration.aspx

8. .netCoders. Guide to UML Diagrams. Activity diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/activity.aspx

9. .netCoders. Guide to UML Diagrams. Component diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/component.aspx

10. .netCoders. Guide to UML Diagrams. Deployment diagrams [online]. [Viitattu 15.10.2006]. Saatavissa: http://www.dotnetcoders.com/web/learning/uml/diagrams/deployment.aspx

11. White, S. A. 2004. Introduction to BPMN [online], IBM Corporation, [Viitattu 17.10.2006]. Saatavissa: http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf

12. Wikipedia. BPMN [online]. [Viitattu 17.10.2006]. Sivujen toteutus: Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/BPMN

13. Miller, M. & Mukerji, J. 2003. MDA Guide Version 1.0.1 [online]. Object Management Group. [Viitattu 20.10.2006]. Saatavissa: http://www.omg.org/docs/omg/03-06-01.pdf.

14. Software Engineering Institute. A Framework for Software Product Line Practice Version 4.1. [online], Pittsburgh: Carnegie Mellon University. [Viitattattu 22.10.2006]. Saatavissa: http://www.sei.cmu.edu/productlines/framework.html

15. Wikipedia. Service oriented architecture [online]. [Viitattu 17.10.2006]. Sivujen toteutus: Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Service-oriented_architecture

16. Immonen M. 2002. Erikoistyö: suunnittelumallit [online]. Kuopio: Kuopion yliopisto tietojenkäsittelytieteen ja sovelletun matematiikan laitos. [Viitattu 25.10.2006]. Saatavissa: http://www.cs.uku.fi/research/Teho/Suunnittelumallit.pdf

17. Koskimies K., Oliokirja Satku - Kauppakaari, Gummerus kirjapaino, Jyväskylä, 2000, 422 s.

18. Wikipedia. Agile software development [online]. [Viitattu 17.10.2006]. Sivujen toteutus: Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Agile_software_development

19. Yoso Oy. Projektiprosessi 2006. Helsinki: Yoso Oy.