



TAMPEREEN
AMMATTIKORKEAKOULU

TIETOJENKÄSITTELY

OPINNÄYTETYÖ

**TILAUS- JA RESURSSIENHALLINTAJÄRJESTELMÄ AGILE-
MENETELMIÄ HYÖDYNTÄVÄÄN TYÖYMPÄRISTÖÖN**
Case TietoEnator Telecom & Media Oy

Marko Kuusniemi

Tietojenkäsittelyn koulutusohjelma
lokakuu 2008
Työn ohjaaja: Paula Hietala

TAMPERE 2008



Tekijä	Marko Kuusniemi	
Koulutusohjelma	Tietojenkäsittely	
Opinnäytetyön nimi	Tilaus- ja resurssienhallintajärjestelmä agile-menetelmiä hyödyntävään työympäristöön, Case TietoEnator Telecom & Media Oy	
Työn valmistumis- kuukausi ja -vuosi	Lokakuu 2008	
Työn ohjaaja	Paula Hietala	Sivumäärä: 37

TIIVISTELMÄ

Opinnäytetyönä on tehty tilausten- ja resurssienhallintajärjestelmä TietoEnator Telecom & Media Oy:lle. Järjestelmä suunniteltiin ja toteutettiin agile-menetelmiä hyödyntävään työympäristöön. Järjestelmän ensisijaisena tavoitteena oli hallinnoida henkilöresurssien saatavuutta ja työtilauskantaan reaaliaikaisesti.

Tilausten- ja resurssienhallinnan parantamisella tavoitellaan tuotantotehokkuutta. Tämä saavutetaan tilausprosessin yhdentämisellä yhden prosessimallin mukaiseksi, sekä jatkuvan työtilauskannan tasaamisella suhteutettuna saatavilla olevien henkilöresurssien määrään.

Agile-menetelmät ovat vanhaan vesiputousmalliin verrattuna uusi kevyempi tapa toteuttaa ohjelmistotuotantoa. Agile-menetelmien yksinkertaisilla pääperiaatteilla pyritään kohdentamaan ohjelmistotuotannon pääpaino: yksilöihin ja näiden interaktioihin, toimivaan lopputuotteeseen, asiakasyhteistyöhön ja sujuvaan reagointiin muutostilanteissa. Tämän lisäksi toisarvoisten asioiden painotusta vähennetään.

TietoEnatorilla hyödynnetään agile-menetelmien scrum-prosessia. Siinä ohjelmistotuotanto tapahtuu lyhyissä ajanjaksoissa, eli sprinteissä, joissa tiimi tuottaa tuotteenhallinnan priorisoimat toiminnallisuudet. Sprint on usein 30-päivän mittainen ajanjakso, joka koostuu suunnittelupalaverista, päivittäisistä tiedontasauspalaverista, demostrointipalaverista ja sprintin parantamispalaverista.

Opinnäytetyönä toteutettu tietojärjestelmä vastaa ensisijaisiin vaatimuksiin tilausprosessia mukaillen ja keräten sen edessä yksityiskohtaisia tietoja keskitetyksi yhteen tietokantaan. Resurssien hallinnassa järjestelmä hyödyntää scrum-prosessin tuottamaa sprintin aikaista resurssiensaatavuuslukemaa. Vertaamalla tätä arvoa tiimin kuukausikohtaiseen tilauskantaan järjestelmällä pystytään tuottamaan raportteja saatavuuden ja tilauskannan eroista. Näiden raporttien ansiosta saatavuus ja tilauskanta pystytään optimoimaan mahdollisimmat tarkasti lähelle toisiinsa jakamalla tilauskanta kuukausisaatavuuksien mukaan. Järjestelmään pystytään syöttämään myös tiimikohtainen oletusarvio saatavuudesta sekä budjetoitu tilauskanta, joiden avulla tulevaisuuden arviointi onnistuu ilman tulevien kuukausien varmistunutta saatavuutta tai tilauskantaan.

Toteutetun tietojärjestelmän ensimmäinen tuotantokaari päättyi järjestelmän käyttöönoton jälkeisten vikakorjauksien loppuun suorittamiseen. Uusien toiminnallisuuksien vaatimuksia ovat mm. lomalistalaajennus, vuosittaisen ylläpitosopimusten hallinta, järjestelmäkäyttäjän sijaisuuksien hallinta ja muutosten historiatiedon hallinnointi.



Author	Marko Kuusniemi	
Degree Programme	Business Information Systems	
Title	Order and resource management tool for agile work environment, Case TietoEnator Telecom & Media Oy	
Month and year	October 2008	
Supervisor	Paula Hietala	Pages: 37

ABSTRACT

Thesis comprises of the designing and implementing of an order and resource management tool for TietoEnator Telecom & Media Oy. The system was designed and implemented for the use of an agile software development work environment. Primary goal was to create a database system and an unified process for managing orders and work resources.

Aim with this new order and resource management tool was to be more efficient. This is possible by unifying the order process under one process model and frequent optimization of the orders and the available work resources.

Agile software development is a new simpler way of doing software development when comparing it to the old waterfall process model. Agile development has few simple guidelines that aim the focus to: individuals and interactions, working end product, customer collaboration, responding to change. On the other hand less focus is shared for secondary issues.

TietoEnator uses the scrum process of the agile methods. In Scrum the software development is done in teams. These teams commit to produce the requirements that product owner has prioritized for the them to do in short period of time called a sprint. Usually the Sprint is 30-days long and during that time team gathers up for design meeting, daily status meetings, demonstration meeting and retrospective meeting.

The implemented system answers to the primary requirements by following the order process and saving all the essential information to one centralized database along the process. Resource management feature of the system uses the total of a teams available work resources that was produced in sprint planning session. Comparing this value to monthly orders that are pointed for each team, the system can produce reports from the available work resources. With these reports the optimization of the available work resources and the orders is easier. The system also accepts a general available resource hours value for each team and also budgetary orders. With these features system can produce a predicts of the future orders and available work resources.

The first production cycle ended with the final fault corrections that were made after the start-up of the system. New pre-planned features include: holiday list feature, upkeeping yearly order contracts, substituting a system user and change log administration.

Sisällysluettelo

1 Johdanto	5
2 Agile-ohjelmistokehitys	7
2.1 Agile-menetelmät	7
2.2 Scrum	8
2.2.1 Tiimi	9
2.2.2 Scrum Master	10
2.2.3 Daily scrum -palaveri	11
2.2.4 Time-boxing.....	12
2.3 Sprint	12
2.3.1 Sprint planning -palaveri ja product backlog	13
2.3.2 Sprint backlog	14
2.3.3 Sprint review ja retrospective.....	16
2.4 Tavoitellut edut	16
2.5 Agile-menetelmät TietoEnatorilla	17
3 Tilaus- ja resurssienhallintajärjestelmä	18
3.1 Vaatimusmäärittely ja suunnittelu	18
3.2 Valitut tekniikat	23
4 Järjestelmän toteutus	24
4.3 Työvaiheet ja valitut toteutustavat	24
4.3.1 Ydinprosessi.....	24
4.3.2 Ohjelmakoodi.....	27
4.3.3 Tietokanta ja kyselyt.....	29
4.3.4 Tyylien hallinta.....	31
4.3.5 Optimoidut käyttöliittymät.....	32
4.4 Tuki scrum-tiimien resurssienhallintaan	33
4.5 Toteutetun järjestelmän arviointi ja jatkokehitys	34
5 Johtopäätökset	36
Lähdeluettelo	37

1 Johdanto

Opinnäytetyönä toteutettiin tilaus- ja resurssienhallintajärjestelmä agile-menetelmiä hyödyntävään työympäristöön. Tämä tietojärjestelmä toteutettiin TietoEnator Telecom & Media Oy:lle vuoden 2007 aikana. Projektin päätarkoitus oli luoda selkeä työkalu, jonka ympärille voitaisiin rakentaa yhtenäinen käytäntö ja tapa, jolla tilauksia, töitä ja resursseja hallittaisiin agile-menetelmiä hyödyntävässä työympäristössä.

Toteutetulla tietojärjestelmällä hallitaan TietoEnatorin saamia tilauksia, yrityksen sisäistä työskentelyä ja käytössä olevia työresursseja. Tietojärjestelmää käytetään myynti-, tilaus-, tuotanto- ja laskutusvaiheissa. Sen avulla pystytään helposti havainnoimaan ja hallinnoimaan halutun hetken käytössä olevat työvoimaresurssit, työkuorma, keskeneräiset työt, laskuttamaton työ, sekä saamaan näistä reaaliaikaisia raportointeja.

Esimieheni toimi esitutkimuksessa ja vaatimusmäärittelyssä järjestelmän tilaajan ensisijaisena edustajana. Hän on yksi yksikköme kolmesta jaospäälliköstä. Lisäksi vaatimuksia antoi ja tarkensi yksi ohjelmistosuunnittelija, kaksi muuta jaospäällikköä, tuotekoordinaattori sekä yksikön päällikkö.

Ensimmäisessä vaiheessa tietojärjestelmä toteutettiin yhden yksikön käyttöön. Tämä tarkoittaa käytännössä, että järjestelmän käyttäjiä on noin 50. Näistä 40 toimii järjestelmän peruskäyttäjänä, eli tavallisena työntekijänä. Loput 10 käyttävät järjestelmää eritasoisina päälliköinä, töiden tai tilausten hallinnoijina. Kehitetyn tietojärjestelmän ensimmäistä demomallia esiteltiin myös muiden jaosten ja yksiköiden päälliköille. Esittelyn tarkoituksena oli näyttää millainen tietojärjestelmä voisi olla myös heidän käytössään lähitulevaisuudessa, sekä kirjata heidän ideoita ja mahdollisia erityisvaatimuksia.

Teknisiltä vaatimuksiltaan tietojärjestelmästä haluttiin selainpohjainen tietokantasovellus. Näin ollen hyvin pian työn aloittamisen jälkeen esivaatimuksissa mainitut MySQL ja PHP vahvistuivat käytettäviksi tekniikoiksi.

Opinnäytetyöni teoreettinen viitekehys käsittelee scrum-prosessimallin mukaisesti toteutettuja agile-menetelmiä projektihallinnan näkökulmasta. Agile-menetelmät ovat uusi, kevyempi tapa toteuttaa ohjelmistotuotantoa, joilla pyritään suoritustehokkaampaan ja joustavampaan ohjelmistotuotantoon. Agile-menetelmistä käytetään myös nimitystä ketterät menetelmät, mutta tässä työssä käytetään ainoastaan termiä agile-menetelmät. Scrum-prosessimallissa scrum-tiimi toteuttaa 30-päivän jaksoissa, eli sprinteissä, sille määritellyt tehtävät ja demonstroi ne sprintin lopussa. Ero perinteiseen vesiputousmalliin on huomattava, jonka ensimmäinen demonstraatio saattaa olla vuosien päässä tuotannon

aloittamisesta. Varsinaista scrummissa suoritettavaa työtä: suunnittelua, toteutusta tai testausta ei käsitellä tässä työssä. Tämä siitä syystä, että toteutettu tietojärjestelmä tukee ainoastaan scrum-menetelmin työskentelevien tiimien saatavuuden ja työkuorman hallintaa. Varsinaiseen scrum-työskentelyn hallintaan on erilliset työkalut ja menetelmät, joista keskeisimmät esitellään tässä opinnäytetyössä.

Lisäksi opinnäytetyössä esitellään otteita toteutetun tietojärjestelmän vaatimusmäärittelystä ja näiden vaatimusten mallintamisesta sekä teknisestä toteuttamisesta. Koko järjestelmää ei siis käydä läpi, vaan ainoastaan muutama oleellinen kokonaisuus. Valittuja osuuksia käsitellään ohjelmistotuotannon standardien ja vakiintuneiden käytäntöjen kautta.

Työhön ei liitetä erillistä sanastoa, sillä jokainen työssä käytetty termi avataan ensimmäisellä käyttökerralla ja keskeiset käsitteet kuvaillaan tarkemmin.

2 Agile-ohjelmistokehitys

2.1 Agile-menetelmät

Agile-ohjelmistokehitys on yleisnimitys menetelmistä, joilla pyritään keventämään ohjelmistotuotantoa. Perinteinen vesiputousmalli on vuosien saatossa koettu raskaaksi ja kankeaksi nopeiden muutosten edessä. Agile-menetelmien avulla ohjelmistokehitys pysyy reaktioherkässä tilassa, jolloin äkillisiin suunnanmuutoksiin pystytään luontevasti. Näillä menetelmillä pyritään ensisijaisesti tuottamaan toimivaa ohjelmakoodia tarkan ja kattavan dokumentaation sijaan. Agile-menetelmissä keskeistä on myös avoin viestintä. Osansa tähän uuden toimintamallin joustavuuteen tuo tilaavan asiakkaan aikaisempaa aktiivisempi yhteistyö tuottavan tiimin kanssa. (Martin 2003: 3 - 9.)

Agile-manifesti syntyi vuonna 2001. Tuolloin seitsemäntoista sen hetken erilaisten agile-menetelmien johtavaa kehittäjää pitivät kaksipäiväisen konferenssin, jonka tuloksena syntyi *The Agile Manifesto*. Kuvaavasti tämä ryhmä otti nimekseen *The Agile Alliance*, joka toimii edelleen voittoa tuottamattomana järjestönä. (Highsmith 2001.)

Manifestin neljä keskeistä pilaria kuvaavat varsin lyhyesti ja ytimekkäästi sen perustan, joihin onnistunut agile-kehitys nojaa.

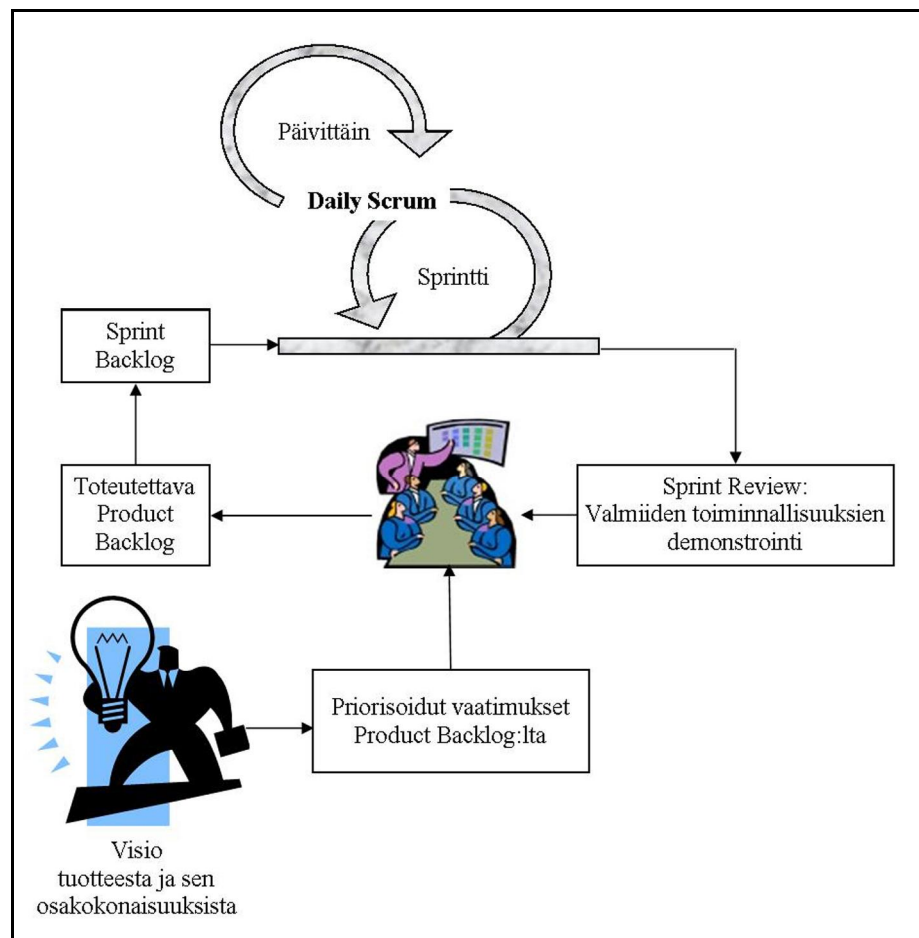
Agile-manifestin pilarit:

- Arvostamme yksilöitä ja interaktioita enemmän kuin prosesseja ja työkaluja.
- Arvostamme toimivaa ohjelmistoa enemmän kuin perusteellista dokumentaatiota.
- Arvostamme asiakasyhteistyötä enemmän kuin sopimusneuvotteluita.
- Arvostamme muutokseen reagoimista enemmän kuin suunnitelman noudattamista.

(Agile Alliance 2001.)

2.2 Scrum

Scrum on yksi prosessimalli, jolla ohjataan agile-menetelmin toteutettavaa ohjelmistotuotantoa (Kuvio 1). Scrum koostuu yhdestä tai useammasta tiimistä, jotka toteuttavat tuotehallinnan priorisoimia vaatimuksia. Nämä vaatimukset tulevat product backlogilta, jota product owner eli tuotehallitsija hallinnoi. Scrum-tiimit toteuttavat siis product backlogilta valittuja toiminnallisuksia maksimissaan 30-päivän sprinteissä. (Schwaber 2004: 5 - 6).



Kuvio 1: Scrum prosessi (mukaillen Schwaber 2004: 9)

Kussakin sprintissä järjestetään sprint planning –palaveri, jossa tiimi suunnittelee scrum masterin johdolla tuotehallinnan priorisoidut vaatimukset yksityiskohtaisiksi tehtäviksi, joilla haluttu toiminnallisuus saadaan aikaan. Samalla määritellään myös näiden tehtävien työmääräarviot ja ne listataan sprint backlogiin. Sprint backlog on siis tiimin yksityiskohtainen tehtävälista kyseiselle sprintille. Tämän jälkeen tiimin jäsenet alkavat suorittaa näitä tehtäviä, raportoiden edistyksensä sprint backlogiin ja kertomalla lyhyesti työnsä etenemisestä tiimilleen päivit-

täisessä daily scrum -palaverissa. Tiimi päättää itsenäisesti omasta toiminnasta. Tiimin scrum master ohjaa tiimiä hallinnollisella tasolla, pyrkien ainoastaan toimittamaan tiimille sen tarvitsemat resurssit, poistamaan mahdollisia esteitä ja valmentamaan tiimin jäseniä toimimaan scrum-prosessin mukaisesti.

Sprint on ohi, kun tiimi on järjestänyt sprint review -palaverin tuotteenhallinnalle, toimittanut tuotoksensa ja lopuksi kokoontunut tiimin kesken sprint retrospective -palaveriin. (Schwaber 2002: 7 – 10.) Sprint review- ja retrospective-palaveria kuvataan tarkemmin luvussa 2.3.3.

2.2.1 Tiimi

Scrum-tiimit ovat yleisesti pieniä, alle kymmenen hengen ryhmiä, jotka koostuvat suunnittelijoista, ohjelmoijista ja testaajista. Rakenteensa vuoksi scrum tiimejä kutsutaankin usein cross functional -tiimeiksi. Tiimien jäsenmäärä ja koostumus voi vaihdella sprintistä riippuen, mutta jokaisen sprintin alussa jokainen tiimin jäsen sitoutuu saavuttamaan sprint planning -palaverissa sovitut päämäärät.

Suosittelava tiimikoko on seitsemän, plus miinus kaksi ja jokaisessa tiimissä tulisi olla vähintään yksi kokeneempi ohjelmoija, joka sprintin kuluessa opastaa nuorempia ohjelmoijia. Isommat tiimit tulisi jakaa kahteen erilliseen tiimiin ja alle kolmen hengen tiimejä ei pitäisi perustaa lainkaan, sillä molemmissa tapauksissa scrummilla saavutettava tehokkuus kärsii. Ihannetilanteessa jokaiseen tiimiin kuuluisi myös erillinen testaaja, joka vastaa lopputuotteen laadusta, sekä erillinen tekninen dokumentoija. Jos tiimissä ei ole erikseen näihin tehtäviin erikoistunutta osajaa, on tiimi silti vastuussa näiden tehtävien suorittamisesta. Näin ollen ohjelmoijat testaavat ja dokumentoivat tuotoksensa itse. (Schwaber 2002: 36 - 38.)

Tiimin varsinaisten jäsenten lisäksi tiimin ympärillä toimii myös muita tahoja, jotka osallistuvat ainoastaan tuki- ja hallintotehtäviin tai ainoastaan tarkkailevat tiimin toimintaan. Tiimin jäseniä kutsutaan Agilemaailmassa sioiksi ja näitä muita osallistujia kanoiksi. Tämä pohjautuu tarinaan, jossa kana ehdotti sialle, että perustetaan ravintola, jossa tarjoillaan kinkkua ja munia. Hyvin ymmärrettävästi sika olisi tässä nahkojaan myöten lautasella, kun kana ainoastaan osallistuu tuomalla muuttaman munan pekonin rinnalle. Sika siis kieltäytyi kanan tarjouksesta. Käytännössä tämä heijastuu agile-maailmaan niin, että kun tiimi koontuu daily scrum -palaveriin, on muillakin kuin tiimin jäsenillä mahdollisuus osallistua, mutta kuitenkin niin, että ainoastaan tiimin varsinaisilla jäsenillä, eli sioilla on lupa puhua. Näin tiimin sisällä säilyy täydellinen autonomia sprintin aikana, eikä kenelläkään tiimin ulkopuolisella taholla ole lupaa ohjata tiimin tuotantoon liittyvää toimintaa tai sen valintoja. (Schwaber 2004: 5 - 6.)

Henkilötasolla tiimin jäsenet ovat yleisesti varsin erilaisia ja jokaisen ydinosaaminen poikkeaa toisesta. Vaikka jonkun rooli saattaisi ensisijaisesti ollakin ohjelmoija, voi hän yhtä hyvin ottaa sprint backlogilta tehtäväkseen testaukseen, suunnitteluun tai dokumentointiin liittyviä tehtäviä. Scrum-menetelmät kannustavat näin tekijöitä mukavuusalueensa äärirajoille laajentamaan omaa osaamistaan sekä ammatillista kehitystään. Lisäksi henkilötasolla on huomioitava jokaisen jäsenen senhetkinen henkilökohtainen tilanne. Muuttuvissa tiimeissä ja suhteellisen nopeatempoisessa sprint-työskentelyssä työlle eri tavoin omistautuminen saattaa luoda turhia jännitteitä tiimin jäsenten välille. Varsinkin jos jollakin tiimin jäsenistä on henkilökohtaisella tasolla normaalia raskaampaa tai muita vaikeuksia. Tämä voidaan nähdä ongelmaksi tilanteissa, joissa muut tiimin jäsenet kokevat tekevänsä toisenkin työt. Tämänkaltaiset henkilökohtaiset vaikeudet tulee huomioida toisen työskentelyssä niin, että hänen osallistumisestaan ja työpanostaan arvioidaan hänen senhetkistä parasta suoritusta vasten, eli voisiko hän edes pystyä kyseisessä tilanteessa parempaan. (Schwaber 2002: 36 - 38.)

Tiimin tärkein tehtävä on saavuttaa sprint planning -palaverissa sovitut päämäärät, eli suorittaa kaikki sprint backlogille asetetut tehtävät. Tiimillä on oltava sama käsitys siitä, milloin mikäkin tehtävä on aidosti valmis. Kun jokin tehtävä on merkitty valmiiksi, sen on tarkoitettava yksiselitteisesti sitä, että kyseisen ohjelmointitehtävän ohjelmointi, testaus ja dokumentointi on suoritettu. Pelkkä testiympäristössä kääntyvä ohjelmakoodi ei siis tarkoita että tehtävä olisi suoritettu. Kun sprint backlogille listatut toiminnallisuudet on demonstroitu ja toimitettu dokumentoituna tuotteenhaltijalle, sprintin tavoitteet on saavutettu. (Shore & Warden 2008: 156 - 157.)

2.2.2 Scrum Master

Tiimillä ei ole varsinaista johtajaa, vaikka yksi jäsen toimiikin scrum masterin roolissa. Hänen tehtävänsä ei kuitenkaan ole johtaa tiimiä vaan ensisijaisesti huolehtia siitä, että scrum itsessään onnistuu. Tämä tarkoittaa siis sitä, että scrum-käytäntöjä noudatetaan. Näitä käytäntöjä ovat mm. daily scrum -palaverit ja että ne järjestyvät time-boxingin rajoissa ja etenevät scrum-metodien edellyttämällä tavalla. Time-boxingista kerrotaan tarkemmin luvussa 2.2.4 ja daily scrum -palaverin metodeista seuraavassa luvussa. Lisäksi scrum master huolehtii siitä, että sprint backlogia ylläpidetään päivittäin edistymisen osalta. Scrum masterin tehtävänä on myös poistaa tiimin toiminnan esteitä ja mahdollistaa näin asetettujen päämäärien saavuttaminen.

Daily scrum -palavereissa scrum master tarkkailee miten tiimin jäsenet ovat edistyneet tehtävissään ja havaitessaan ongelmia hän pyrkii mahdollisuuksien mukaan poistamaan ongelman tai järjestämään apua muilla tavoin.

Scrum master vastaa myös tuotteenhallinnalle ja muille hallinnollisille tahoille tiiminsä edistymisestä. Näin ollen tiimin muut jäsenet pystyvät paremmin keskittymään suorittamaansa tehtävään hallinnollisten tehtävien pysyessä daily scrum -palaverien ja sprint backlog -päivitysten tasolla.

Tilanteissa, joissa useampi tiimi työskentelee saman product backlogin ympärillä scrum masterit järjestävät daily scrum -palaverin lisäksi oman päivittäisen scrum of scrums -palaverinsa, jossa he tasaavat tietonsa tiimiensä edistymisestä samaan tapaan kuin daily scrum -palaverissa. (Schwaber 2002: 31 - 32.)

2.2.3 Daily scrum -palaveri

Lyhyesti kuvailtuna daily scrum -palaveri on päivittäinen status-palaveri, jossa jokainen tiimin jäsen kertoo lyhyesti, mitä hän on tehnyt edellisen ja tämän palaverin välillä ja mitä aikoo tehdä tämän ja tulevan palaverin välillä. Lisäksi hän raportoi mahdollisista ongelmista tai esteistä ja puuttuvista taskeista sprint backlogilta jotka koskevat hänen suorittamaa tehtävää.

Daily scrum -palaveri on suositellusti aikarajattu, eli time-boxattu maksimissaan 15 minuutin mittaiseksi. Scrum master valvoo tämän toteutumista. Palaverissa ei keskustella muusta kuin kyseisen sprintin tehtävistä. Tehtävistä ei puhuta toteutus- tai suunnittelutasolla, vaan ainoastaan niiden edistymisen osalta.

Päivittäistä palaveria varten on hyvä olla oma kokoontumistila, josta löytyy kaikki tarvittava palaverin onnistumisen kannalta. Joissakin tiimeissä saattaa olla jäseniä, jotka työskentelevät osittain tai jopa kokonaan toisessa toimipisteessä. Tällöin on järjestettävä konferenssipuhelu jokaiseen palaveriin, jotta kaikki tiimin jäsenet pystyvät osallistumaan. Lisäksi tilassa on hyvä olla fläppitaulu tai taulu, johon voidaan tarvittaessa kirjoittaa tai kaavioida kyseisen sprintin tehtäviä koskevia yksityiskohtia. Tilassa on myös oltava mahdollista kokoontua ympärään ja mahdollisuus istua tai seistä pöydän ääressä. Seisomisen on todettu nopeuttavan palaverin etenemistä. (Schwaber 2002: 40 - 46.)

Palaveri etenee niin, että jokainen kertoo vuorollaan tiivistetysti sen, mitä hän on tehnyt edellisestä palaverista saakka ja mitä aikoo tehdä seuraavaan palaveriin mennessä. Kaikki muut kuuntelevat hiljaa. Palaverissa on tarkoitus antaa lyhyt ja ytimekäs kuvaus siitä, miten kyseisen henkilön suorittama työ etenee. Mikäli tehtävän suorittamisessa on ongelmia tai esteitä niistä raportoidaan palaverissa. Scrum master on ensisijaisesti vastuussa ongelmien ja esteiden poistamisesta. Mikäli hän ei onnistu poistamaan tiimin raportoimia ongelmia seuraavaan daily scrum -palaveriin mennessä scrum master raportoi tästä, jotta tiimi tie-

dostaa asian olevan käsittelyssä. Palaveriin voi myös osallistua ulkopuolisia tarkkailijoita, mutta he jäävät tiimin jäsenten muodostaman ympyrän ulkopuolelle. Heillä on oikeus tarkkailla tilannetta, muttei lupaa puhua tai millään muullakaan tavalla vaikuttaa palaveriin. Kaikki taustakeskustelu on siis kielletty, myös tiimin jäsenten kesken.

Ajoittain palaverissa ilmenee asioita, jotka vaativat tietyltä tiimin osalta jatkopalaverin, jossa asiaa käsitellään tarkemmin. Tällaisia asioita saattavat olla mm. alkuperäisen toteutussuunnitelman uudelleensuunnittelu teknisten ongelmien takia. Näin daily scrum -palaveri ei venny tuhlauksena tarpeettomasti koko ryhmän työaikaa. (Highsmith 2004: 192 - 194.)

2.2.4 Time-boxing

Scrum-prosessissa ajanhallintaan liittyvä ryhti saavutetaan time-boxingin avulla. Tämä tarkoittaa yksiselitteisesti sitä, että jokaisella palaverilla on tarkka maksimikesto. Scrum master valvoo, että kaikki ovat ajoissa paikalla ja ettei määritelty kesto ylitä. (Schwaber 2004: 8 - 9, 37, 51 - 52.) Hyvä käytäntö on, että myöhästymisistä ja luvattomista poissaoloista rangaistaan sovittu summa, esimerkiksi yksi euro ja kootut varat lahjoitetaan hyväntekeväisyyteen (Schwaber 2004: 43).

Esimerkkejä time-boxing rajoituksista taulukossa 1 mukailten Schwaberin antamia esimerkkejä (2004: 133 - 139).

Taulukko 1: Esimerkkejä time-boxing rajoituksista

Tapahtuma	Kesto
Sprint planning -palaveri	8 tuntia (yksi tauko)
Daily scrum -palaveri	15 minuuttia
Sprint	30 kalenteripäivää
Sprint review -palaveri	4 tuntia
Sprint retrospective -palaveri	3 tuntia

2.3 Sprint

Sprint on scrummin keskeisiä määritelmiä. Se on maksimissaan 30-päivän ajanjakso, jossa tiimi sitoutuu suorittamaan valitut toiminnallisuudet product backlogilta. Rakenteeltaan sprint koostuu: sprint planning, daily scrum sekä sprintin lopussa järjestettävistä sprint review ja sprint retrospective -palavereista.

Sprintin aikana tiimin jäsenten on huolehdittava päivittäin kahdesta asiasta: sprint backlogin päivittämisestä, sekä daily scrum -palavereihin osallistumisesta. Kaikki muu toiminta on vapaasti tiimin päätettävissä,

kunhan sen tarkoitus on saavuttaa sprint planning -palaverissa määritellyt päämäärät. (Schwaber 2002: 52 - 53.)

2.3.1 Sprint planning -palaveri ja product backlog

Scrum-prosessilla kehitettävän tuotteen vaatimukset scrum-tiimille tulevat tuotteenhallinnan ylläpitämältä product backlogilta (Kuvio 2). Tuotteenhallinta toimii siis ylemmällä arkkitehtitasolla määrittellen tuotteen uusia tai päivitettäviä toiminnallisuuksia. Nämä määrittelyt on myös priorisoitu niin, että kunkin sprintin aikana suoritetaan vain sen aikana valmistuvaksi arvioitu määrä priorisoituja toiminnallisuuksia. Kun nämä uudet toiminnallisuudet on demonstroitu, tuotteenhallinnalla on tarvittaessa aikaa tehdä muutoksia priorisointeihinsa seuraavan sprintin vaatimusten osalta. (Schwaber 2002: 48 - 49.)

Sprint planning -palaverissa tiimi kokoontuu pohtimaan tuotteenhallinnan asettamia vaatimuksia tulevalle sprintille. Tuotteenhallinnan priorisoimat vaatimukset product backlogilta pyritään pilkkomaan mahdollisimman pieniksi tehtäviksi, jotta ne voitaisiin listata järkevästi sprint backlogille. Listauksen jälkeen jokaiselle tehtävälle arvioidaan toteuttamiseen kuluva aika tuntitasolla, jonka tulisi olla maksimissaan 16 tuntia tehtävää kohden. Yhden tehtävän tuntiarvion ollessa suurempi, se pilkotaan pienempiin osiin. Tehtävien lisäksi tiimi määrittelee oman sprint-kohtaisen saatavuutensa, eli tiimin jäsenten yhteenlasketun tuntimääräisen henkilöresurssisaatavuutensa. Yhteenlasketun tehtävien tuntiarvion ja tiimin kyseisen sprintin saatavuuden pohjalta pystytään laskemaan, onko halutut toiminnallisuudet mahdollista toteuttaa halutussa ajassa vai ei. Tarvittaessa neuvotellaan tuotteenhallinnan kanssa toiminnallisuuksien vähentämisestä tai lisäämisestä, mikäli toteutusarvio ei vastaa saatavuutta. (Martin 2003: 19 - 22.)

Req ID	Business Priority	Architectural priority	Requirement Category (Feature)	Product Requirement i.e. Business Requirement
REQ_01	Important	Important	ISR Core	1.2 Data type conversions New engine shall support data type conversions between external interface and ISR internal data format. Conversion between new datatypes like int, word, char, ... shall be supported.
REQ_02	Important	Important	ISR Core	1.3 Complex data type conversions

Functionality	Functionality Area	Business Priority	Architectural priority	EE	Status (new, ongoing, complet)	Phase
Specification of supported conversions	New engine	Important	Important	75	Ongoing	FS
Implementation of the conversion routines	New engine	Important	Important	625	New	D&I
FT	New engine	Important	Important		New	FT
PET	New engine	Important	Important		New	PET
Specification of supported conversions	New engine	Important	Important	75	Ongoing	FS
Not needed						

Kuvio 2: Esimerkki product backlogin sisällöstä

2.3.2 Sprint backlog

Sprint backlogille on sprint planning -palaverissa listattu kaikki yksilölliset tehtävät, jotka tiimin on tarkoitus suorittaa sprintin aikana (Kuvio 3). Tehtävien lisäksi taulukosta löytyy sprintin yhteenlaskettu henkilöresurssisaatavuus tunteina, sekä burndown chart, eli diagrammi tiimin edistymisestä verrattuna saatavuuteen. Tehtäväkuvauksen viereltä löytyy kolme keskeistä tietoa kyseisestä tehtävästä: sen suorittaja, tila ja tunti-arvio jäljellä olevasta työstä. (Schwaber 2002: 71.)

Tasks	Status	WHO?	Hours of Effort Remaining ---->	Planning							
				14.2.08	15.2.08	16.2.08	17.2.08	18.2.08	19.2.08	20.2.08	
Hour bank(EH): 790				776	751	718	681	658	615	615	
Specified task list				795	765						
REQ1000 - Basic MMIO connection parameters	complete	mmio	24	24	24	18	8	8	8	8	
REQ1000 - MMIO -> Searching for changes in active connection status	complete	mmio	4	4	0	0	0	0	0	0	
REQ1000 - MMIO -> Informing bridge about connection status	complete	mmio	1	1	1	1	1	1	1	1	
REQ1000 - MMIO -> Sending notification message to MMIOv1 v1	Ongoing	mmio	6	6	6	6	5	5	5	5	
REQ1000 - MMIO -> Generating automatic update user	complete	mmio	2	2	2	1	1	1	1	1	
REQ1000 - MMIO -> Sending new active connection v1	Ongoing	mmio	36	36	36	36	36	32	32	32	
REQ1000 - MMIO -> Sending active MMIO to MMIOv1 v1	descoped	mmio	2	2	2	0	0	0	0	0	
REQ1000 - MMIO -> Sending active MMIO to MMIOv1 v1	Ongoing	mmio	6	6	6	6	6	3	3	3	
REQ1000 - MMIO -> Sending automatic update user v1	complete	mmio	1	1	1	1	1	1	1	1	
REQ1000 - MMIO -> Sending automatic update user v1	complete	mmio	6	6	3	3	3	0	0	0	
REQ1000 - MMIO -> Sending automatic update user v1	complete	mmio	6	6	3	3	3	0	0	0	

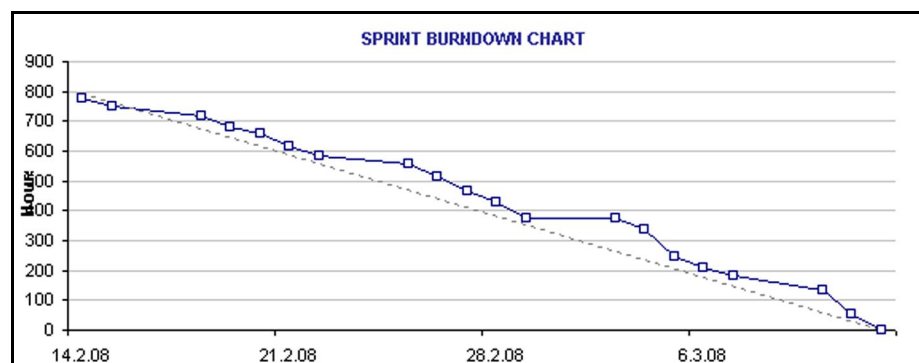
Kuvio 3: Esimerkki sprint backlogin sisällöstä

Suorittaja-kenttä kertoo kuka kyseistä tehtävää on suorittamassa ja ideaalitulanteessa yhdellä tiimin jäsenellä on kerrallaan vain yksi tehtävä työstettävänä. Usein käytännössä käy kuitenkin niin, että tehtävät muodostavat kokonaisuuksia, jotka yksi tiimin jäsen suorittaa.

Tila-kenttä on normaalitulanteessa määritelty jollekin seuraavista kolmesta tasosta: aloittamatta, kesken tai valmis. Nämä kertovat yksiselitteisesti missä tilassa kyseinen tehtävä on. Erikoistapauksissa tila voi olla myös: hylätty tai siirretty. Näitä käytetään tilanteissa, jolloin kyseistä tehtävää ei tarvitse suorittaa lainkaan tai jos sen suoritus syystä tai toisesta siirtyy seuraavaan sprinttiin.

Tuntiarvio-kenttiä on alkuperäisen tuntiarvion lisäksi yhtä monta, kuin kuluvasse sprintissä on työpäiviä. Tiimin jäsenten on päivittäin täydennettävä sprint backlogia oman tehtävänsä osalta niin, että sen päivän tuntiarvio vastaa aitoa jäljellä olevaa työtuntimäärää. Näin ollen tuntimäärä voi joskus jopa kasvaa yllätysten tai muiden vaikeuksien edessä. Tehtävän viivästymisestä on ehdottomasti raportoitava tiimille daily scrum -palaverissa. Sprint backlog on projektihallintatyökalu ja jotta sillä saadaan aito kuva tehtävän työn valmistumisesta, on tiimin oltava rehellinen pientenkin tehtävien tuntiraportoinnissa.

Burndown chart piirtää diagrammin tiimiin edistymisestä päiväkohtaisesti (Kuvio 4). Parhaimmassa tapauksessa aloitushetkellä saatavuus ja työmäärä aloittavat täsmälleen samasta kohdasta, mutta tämä on harvinaista. Saatavuus piirtyy diagrammiin tasaisena suorana viivana alkaen ylhäältä vasemmalta, laskeutuen tasaisesti oikeaan alakulmaan, eli sprintin päätökseen. Saatavuus jakautuu siis tasaisesti kaikille työpäiville. Työn edistymisestä piirtyy käyrä sen mukaan, miten jäljellä olevaa työmäärä on tehtäväkohtaisesti raportoitu. Päämääränä on siis saavuttaa samainen oikea alakulma sprintin päättyessä. Tällöin jokaisen tehtävän työmäärä on asetettu nolnaan ja tilaksi merkitty suoritettu, hylätty tai siirretty. (Schwaber 2004: 10 - 14.)



Kuvio 4: Esimerkki burndown chartista

Yhteenlaskettua sprint-kohtaista saatavuutta käytetään opinnäytetyönä toteutetun tilaus- ja resurssienhallintajärjestelmässä silloin, kun sen avulla pyritään hahmottamaan saatavuutta suhteutettuna nykyiseen ja tulevaan tilauskantaan.

2.3.3 Sprint review ja retrospective

Sprintin lopussa tiimi järjestää kaksi erillistä palaveria: sprint review -palaverin tuotteenhallinnalle ja sprint retrospective -palaverin tiimille itselleen.

Sprintin lopulla valmistuneet toiminnallisuudet esitellään tuotteenhallinnalle sprint review -palaverissa. Useamman tiimin työskennellessä saman product backlogin ympärillä, voidaan järjestää yksi yhteinen sprint review -palaveri kaikille tiimeille. Näin tuotteenhallinta näkee yhdessä palaverissa koko tuotteen kehityksen kyseisen sprintin aikana.

Käytännössä sprint review -palaveri järjestetään usein testilaboratoriossa tai testivastuullisen työpisteellä, jossa toiminnallisuuksien esittely on helppoa. Jokainen tiimi esittelee vuorollaan kyseisen sprintin tavoitteet ja lopputuloksensa, jolla nämä tavoitteet on saavutettu. Esittelyn jälkeen tuotteenhallinnalla on mahdollisuus kysellä ja keskustella tiimin kanssa toteutetusta toiminnallisuudesta. Esityksen ja saamiensa vastausten pohjalta tuotteenhallinta voi selkeämmin suunnitella tarvittavat muutokset ja uudet toiminnallisuudet seuraaville sprinteille. (Schwaber 2002: 54 - 56, 2004: 56 - 57, 137 - 138.)

Sprint retrospective -palaverissa käsitellään ja arvioidaan kuluneen sprintin kulku. Ensisijaisesti tarkoitus on käsitellä scrummin toimivuutta, mutta tässä on hyvä tilaisuus purkaa myös muita tunteita sprintin onnistumisen osalta. Asioista listataan yhteenveto kahden otsakkeen alle, mikä meni hyvin ja missä on parannettavaa. Seuraavassa sprint retrospective -palaverissa tiimi voi verrata edellisellä kerralla listattuja asioita nykytilanteeseensa ja arvioida kehitystään. (Schwaber 2004: 102, 108, 138 - 139.)

2.4 Tavoitellut edut

Agile-menetelmillä saavutettavat edut voidaan jakaa kolmeen kategoriaan: organisaation menestymiseen, tekniseen ja henkilökohtaiseen onnistumiseen. (Shore & Warden 2008: 5.)

Organisaation menestymisellä tarkoitetaan hyvän tuloksen lisäksi myös mm. kilpailukykyistä erottuvuutta, sekä onnistunutta brändi- ja asiakaskeisyyttä. Agile-menetelmillä taataan organisaation menestymisen keskittymällä toimitettavan tuotteen arvoon ja kulujen vähentämi-

seen, eli tuoton kasvattamiseen. Käytännössä tämä mahdollistuu aikaisessa vaiheessa havaittujen virheiden tai lopputuotteen mahdollisten ongelmien tiedostamisella. Tällöin ne voidaan kustannustehokkaasti korjata. Esimerkiksi joissakin tilanteissa voidaan kesken projektin havaita lopputuotteen olevan täysin käyttökelvoton alkuperäisen suunnitelman mukaan toteutettuna. Tällöin nopeiden korjausliikkeiden tekeminen on korvaamattoman tärkeää organisaatiotason onnistumisen takaamiseksi. (Shore & Warden 2008: 6.)

Tekninen onnistuminen mahdollistuu agile-menetelmien mukanaan tuomin hallintamenetelmin (Shore & Warden 2008: 6). Scrum-prosessin yhteydessä suurinta roolia näyttelevät sprint backlog ja daily scrum palaveri. Näiden avulla työn etenemistä voi seurata reaaliaikaisesti. Suoritettavat tehtävät ja niiden jäljellä oleva työtarve on helposti ja selkeästi havaittavissa ja hallinnoitavissa. (Schwaber 2004: 7 - 12.) Tämä tiivistynyt yhteistyö osaltaan edistää tuotteen osien yhteensopi- vuutta ja mahdollista siten teknisen onnistumisen.

Henkilökohtaisella onnistumisella tarkoitetaan henkilön roolista riippuen erilaisia asioita. Ohjelmistokehittäjän kannalta tämä tarkoittaa lähinnä lopputuloksen laadukasta teknistä toteutusta, onnistuneita työmäärä- arviointeja ja aikatauluja, sekä autonomista tiimityöskentelyä. (Shore & Warden 2008: 6.)

2.5 Agile-menetelmät TietoEnatorilla

Tampereen TietoEnator Telecom & Media Oy:n NRE-yksikössä siirryttiin käyttämään agile-menetelmiä vuosien 2007 ja 2008 aikana. Suurin syy muutokseen oli suurimman asiakkaan toive siirtyä käyttämään näitä menetelmiä. Asiasta tehtiin ensin esitutkimusta ja sen jälkeen asiakkaan mallista mukailtuja menetelmiä testattiin pilottiryhmissä vuoden 2007 aikana. Pilottiryhmät olivat varsin tyytyväisiä uusiin menetelmiin ja vuoden 2008 alussa menetelmät otettiin käyttöön koko yksikössä. Tämä muutos tapahtui työssäoloaikani, jolloin en vielä ollut tietoinen agile-menetelmistä. Sisäisten koulutusten kautta olen saanut tietooni miten agile-menetelmiä sovelletaan TietoEnatorilla. Vaikka agilen henkeen kuuluukin vahvasti ajatus tiimin omasta päätäntävällas- ta prosessin toteutuksen suhteen, ovat teoreettiset scrum-perusteet säilyneet lähes muuttumattomana jokaisessa tiimissä.

3 Tilaus- ja resurssienhallintajärjestelmä

3.1 Vaatimusmäärittely ja suunnittelu

Vuoden 2006 lopulla TietoEnator Telecom & Media Oy:n Registersyksikössä oli suoritettu esitutkimus uuden tietojärjestelmän tarpeellisuudesta tilausten- ja resurssienhallintaan. Tutkimus osoitti käytössä olleen erilaisia työkaluja ja tapoja, jotka haluttiin yhdistää ja yhdenmukaistaa yhden järjestelmän ja toimintatavan mukaisiksi. Tutkimuksen pohjalta kyseinen järjestelmä päätettiin toteuttaa (Pohjalainen 2002: 27). Järjestelmän lähtökohdista laadittiin esittelymateriaalia, jota alettiin käyttää tulevan toteuttajan etsinnässä. Haku oli päätetty kohdentaa TAMK:n tietojenkäsittelyn tradenomiopiskelijoihin tarjoten projektia suoritettavaksi työharjoitteluna tai opinnäytetyönä. Toteuttaja valittiin joulukuussa 2006 ja työ aloitettiin tammikuussa 2007.

Esitutkimuksen ollessa suoritettu, työ päästiin aloittamaan vaatimusmäärittelystä. Esitutkimuksen pohjalta listattuja asiakasvaatimuksia täsmänsi palaveripohjaisissa aivoriihissä kolme jaospäällikköä, tuotekoordinaattori sekä yksikön päällikkö. Vaatimukset kirjattiin Haikalan ja Märijärven (2004) ohjeiden mukaisesti tyyppin, päiväyksen ja prioriteetin kera taulukkoon (Taulukon 2). Vaatimusten antajaa ei katsottu tarpeelliseksi listatta erikseen, sillä järjestelmää tuotettiin yrityksen sisäiseen käyttöön. (Haikala & Märijärvi 2004: 78 - 81, 91 - 97.)

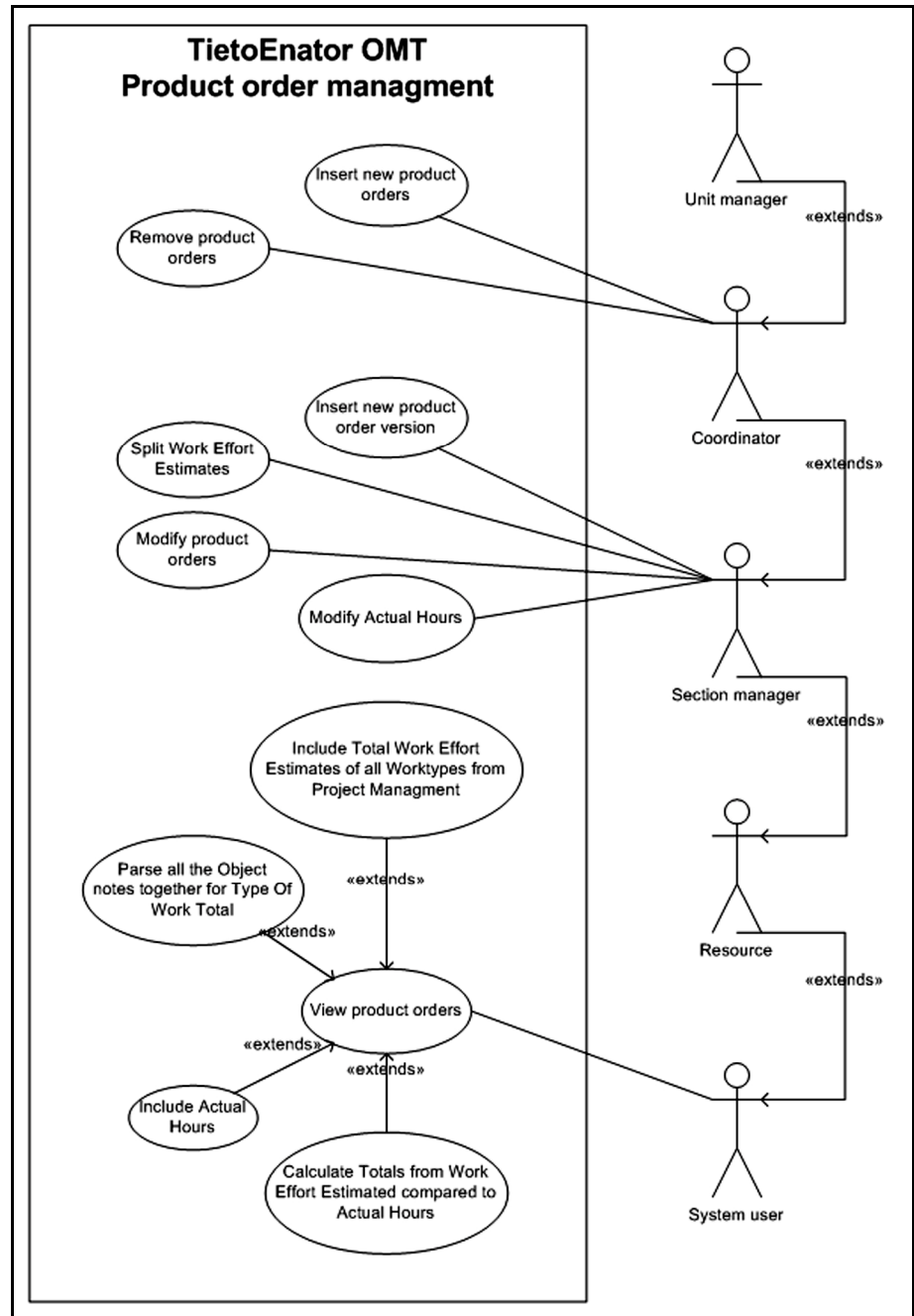
Taulukko 2: Ote tilaus- ja resurssienhallintajärjestelmä vaatimuksista

No	Type	Date	Description	Priority
TE OMT REQUIREMENTS				
1	Business	1.2.2007	Product Order management (Insert/Modify/Remove)	1
2	Business	1.2.2007	Work Effort Estimate management (Demand)	1
3	Business	1.2.2007	Resource management (Resources)	1
4	Business	1.2.2007	Get Resource / Demand reports	2
5	Business	1.2.2007	Get Product Order reports	2
6	Business	1.2.2007	Get ETC reports	2
7	Business	1.2.2007	Get Sold/Billed reports	2
8	Business	1.2.2007	Get Contract reports	2
9	Business	1.2.2007	Get Actual Hours & Missing Actual Hours reports	2
10	Functional	1.2.2007	Excel-export	3
11	Functional	1.2.2007	Excel-Import	4
12	Functional	1.2.2007	System log & history log from all changes	2
13	Functional	1.2.2007	Internal messaging system	4
...

Ideointipalavereja järjestettiin koko projektin aikana kahdeksan, joista jokainen täsmensi aiemmin määriteltyjä vaatimuksia, sekä toi mukanaan uusia. Kaiken kaikkiaan vaatimuksia listattiin 111 kappaletta ja ne asettuivat 26 eri yläotsikon alle.

Palaverit olivat vaatimusten ensisijainen lähde, mutta joitakin vaatimuksia täsmennettiin yksilötasolla tai pienemmissä ryhmissä. Vaatimusmäärittely on työnä hankala, eikä sen toteuttamiseen ole selkeää ja suoraviivaista mallia. Tämän vuoksi suoritettu vaatimusmäärittely olikin kokoelma erilaisia ratkaisuja, jotka yhdessä muodostivat toimivan kokonaisuuden kyseisen projektin vaatimusmäärittelylle (Pohjalainen 2002: 28).

Järjestelmän suunnittelutyö käynnistyi heti ensimmäisen palaverin jälkeen. Ohjelmistoa kehitettiin pieni osa kerrallaan päämäärittelyn suuntaviivoja noudatellen. Näin pienoisesiputousmallin tapaisesti määrittely, suunnittelu, toteuttaminen ja testaus olivat toistuvia prosesseja koko projektin läpi (Pohjalainen 2002: 29). Kaikkia vaatimuksia ei suunniteltu tai mallinnettu sen tarkemmin niiden yksiselitteisen luonteensa vuoksi. Laajemmat kokonaisuudet mallinnettiin UML:n käyttötapausten avulla kuvion 5 tapaan, niin että kyseinen tapaus ja siihen liittyvät toimijat kuvattiin keskinäisten suhteiden yhdistäminä.



Kuvio 5: Esimerkki järjestelmän UML-käyttötapauskaaviosta

Lisäksi käyttötapauksista kirjoitettiin sanallinen kuvaus niiden toiminnasta TietoEnatorin dokumenttipohjiin esimerkki kuvion 6 tyyliä käyttäen (Eriksson & Penker 2000: 8). Käyttötapausten kuvauksesta löytyvä aina vähintään seuraavat tiedot: toiminnon tarkoitus, miten tapaus käynnistyy, toimijat, viestien kulku toimijoiden ja muiden käyttötapausten välillä, vaihtoehdot toimintatavat sekä milloin käyttötapaus päättyy ja mitkä ovat palautetut tulokset (Eriksson & Penker 2000: 49 - 50).

TietoEnator 123

USE CASE: 1.3. View product order 26.02.2007 Version 0.001

Overview
Basic information of Product order may be shown to any user who has valid active permission

Actor	Frequency of usage (e.g. transactions / day)
Coordinator	40 / week
Link manager	20 / week
Section manager	30 / week
Resource	10 / week
System user	2 / week

Usability requirements

Preconditions
System has identified the user when he passed the system login. Still the system will check if the user still has an active permission to use the system or does he need login again.
Product order has already been created.

Post conditions
No changes.

Description of the use case	References
The system checks the active permission to be still valid.	UC: 25. Login & User Check E1
The system selects the wanted data of PRODUCTORDER and displays them to the user	UI: Product Order R1

Exceptions

No	The description of the exception	References
E1	User is not allowed to view product orders, because his active permission is no longer valid.	R2

Alternative choices

No	Alternative	Decision control	References

© TietoEnator Corporation - TE Object 50 | © Document and Software Management/2006/09/26
SRTO_ENATOR_LUMA/Käyttökäytännöt/TECOM/USE_CASE/1.3 View Product Order/001.doc

TietoEnator Use case: < name > 323

No **The results from the actors' point of view**

R1	Product Order has been shown successfully.
R2	System was unable to show Product order. System opens user main page.

Additional information

Open issues

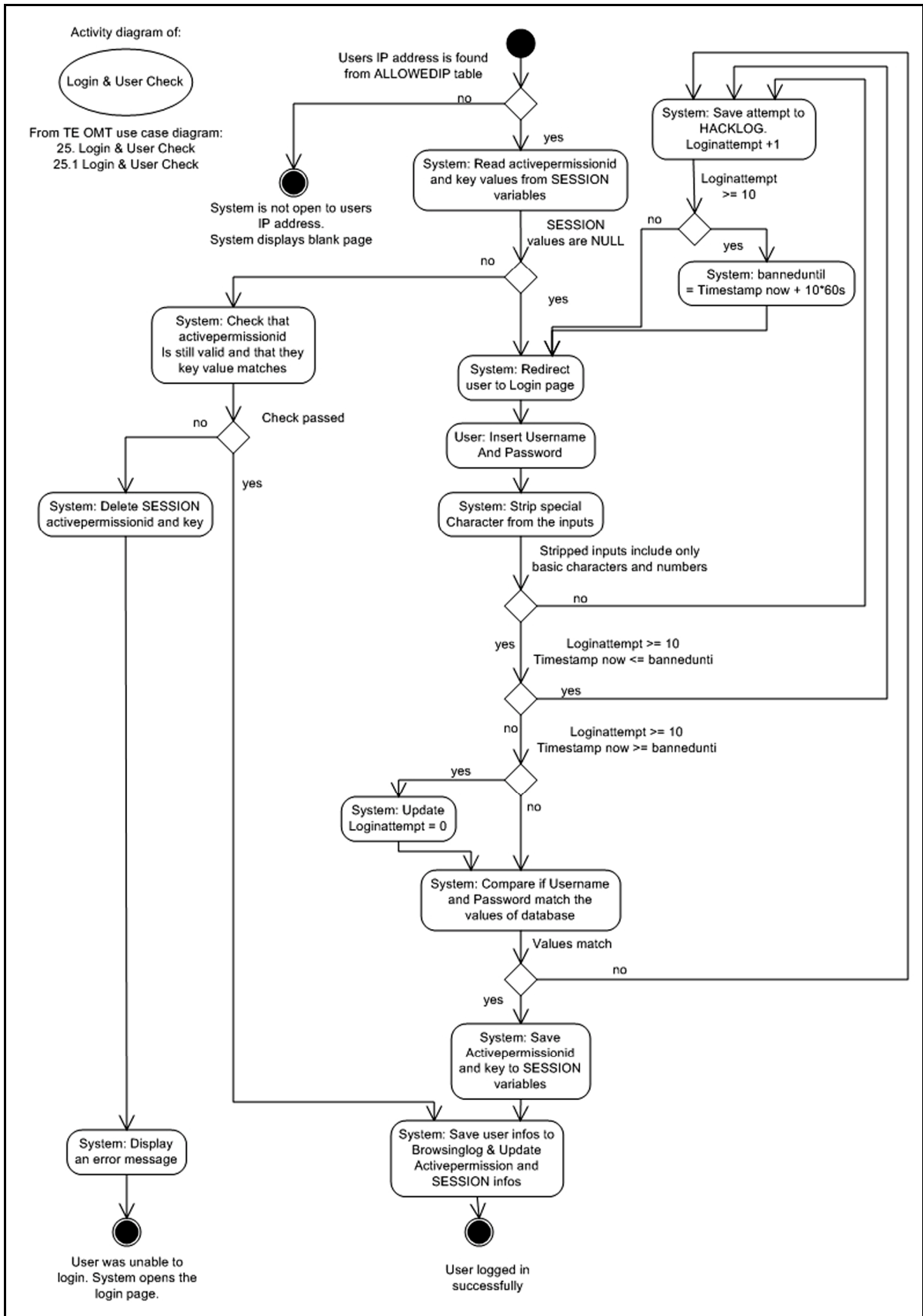
Version history

Version	Date	Author	Change description
0.001	26.2.2007	Mikko Kuosmanen	Document Created

© TietoEnator Corporation - TE Object 51 | © Document and Software Management/2006/09/26
SRTO_ENATOR_LUMA/Käyttökäytännöt/TECOM/USE_CASE/1.3 View Product Order/001.doc

Kuvio 6: Esimerkki käyttötapauksen sanallisen kuvauksen dokumentoinnista

Järjestelmää ei toteutettu oliopohjaiseksi, joten luokkakaavioita ei näin ollen tehty lainkaan. Viestiyhteykskaaviot jouduttiin myös jättämään pois, sillä järjestelmään ei luotu selkeitä alijärjestelmiä, joilla luokkien ilmentymät olisi voitu viestiyhteykskaavioissa korvata. Toiminnallisuuksien tarkempaa kuvailua varten käytettiin toimintokaavioita kuvion 7 tavalla. Toimintakaavioilla haluttiin selventää monimutkaisia käyttötapauksia ja niiden toiminnan seurauksia (Eriksson & Penker 2000: 128).



Kuvio 7: Esimerkki järjestelmän UML-toimintokaavio

3.2 Valitut tekniikat

Esitutkimuksessa ehdotetut keskeiset toteutustekniikat, PHP-skriptikieli ja MySQL-relaatiotietokanta varmistuivat määrittelyvaiheessa käytettäviksi tekniikoiksi. Palvelimeksi valittiin Windows Server 2003, jolle TietoEnatorin paikallinen tukiyksikkö tarjosi perusasennuksen ja ylläpidon. Windows palvelimen, Apache Web -serverin, XHTML- tai CSS-tekniikoiden käyttöä ei ole valinnan osalta tarpeen selventää maintaa enempää.

PHP

PHP, eli PHP: HyperText Preprocessor on palvelinpuolen skriptikieli, joka suorittaa ohjelmoidun toiminnallisuuden palvelimella ja palauttaa käyttäjälle ainoastaan lopputuloksen (Zandstra 2001: 20). PHP valittiin mm. seuraavista syistä:

- Ohjelmakoodi on suojattuna palvelimella, eli asiakas ei näe tai pysty muokkaamaan sitä millään tavoin.
- Mahdollistaa tietokantayhteyden hakuja, lisäyksiä, muutoksia ja poistoja varten.
- Takaa tehokkaan ja monipuolisen tietojenkäsittely tietokannasta haetulle datalle.
- Aktiivinen kehitystyö ja sen takaama tietoturva.
- Laaja levinneisyys, jolloin mahdollisia jatkokehittäjiä on helppo löytää.
- Ohjelmakoodia on helppo muokata ja päivittää.
- Maksuttomuus.

Kilpaileviin tekniikoihin mm. Sun Micro Systemsin JavaServer Pages (JSP) ja Microsoftin Active Server Pages (ASP) ei toiminnallisuuden osalta ole merkittäviä eroja. Yllä olevat ominaisuudet yhdistyvät kuitenkin parhaiten PHP:ssä, joten tämä valittiin toteutuskieleksi.

MySQL

MySQL valittiin käytettäväksi tiedontalletusvarastoksi. MySQL-relaatiotietokanta on nimensä mukaisesti Structured Query Language, eli SQL-pohjainen tietokanta, jonka muokkaamiseen käytetään SQL-standardin mukaisia komentoja (Lahtonen 2002: 38). MySQL valittiin mm. seuraavista syistä:

- Mahdollisuus käyttää tietokantarelaatioita, perus- ja viiteavaimia, sekä indeksejä.
- Tietokantaa on helppo käyttää PHP-koodin avulla.
- Suorituskyvynäkökulmasta tietokanta on kevyt ja tehokas käyttää.
- Sen kehitys on jatkuva ja vakaalla pohjalla.
- Tietokannan käyttöön on saatavilla laaja valikoima hyviä ja laadukkaita tukityökaluja.
- Maksuttomuus.

4 Järjestelmän toteutus

4.3 Työvaiheet ja valitut toteutustavat

Kappaleessa kuvataan järjestelmän ydinprosessi. Teknisestä toteutuksesta esitellään lyhyesti millaisia toteutustapoja ja malleja hyödynnettiin itse ohjelmakoodissa, tietokannan rakenteessa, kyselyistä sekä tyylien hallinnassa ja käyttöliittymien optimoinnissa.

4.3.1 Ydinprosessi

Järjestelmä toteutettiin kuvion 8 kaltaisen ydinprosessin mukaiseksi. Prosessi itsessään rakentuu tuotetilauksen, eli *product orderin* elinkaaren ympärille. Ydinprosessin ympärille rakentuu neljä käyttäjäroolia, joiden käytetyimmät toiminnot pystytään kyseisestä kuvioista myös helposti havaitsemaan.

Ensimmäisessä vaiheessa tuotekoordinaattori neuvottelee asiakkaan kanssa uudesta tilauksesta. Oman esikäsittelynsä jälkeen tuotekoordinaattori tallentaa tilauksen perustiedot ja mahdollisten toteutusvariaatioiden tiedot järjestelmään.

Toisessa vaiheessa tuotekoordinaattori täsmentää järjestelmään jokaisen toteutusvariaation tietoja mahdollisimman yksityiskohtaiselle työvaihetasolle siitä, mitä tuoteversiota ja ohjelmalohkoa kyseinen tuotetilaus koskee.

Kolmannessa vaiheessa tuotekoordinaattori esiarvioi itse tai pyytää kyseisen osa-alueen specialistilta toteutettavan työvaiheen työmääräarvion ja syöttää sen järjestelmään.

Neljännessä vaiheessa jaospäällikkö voi merkitä tuotetilauksen avoimena olevan työvaiheen budjetilliseksi varaukseksi. Budjetillisella varauksella tarkoitetaan tuleviin mahdollisiin tilauksiin ennakoivaa budjetillistä resurssivarausta, jota käsitellään varsinaisen tilauksen tapaan.

Viidennessä vaiheessa jaospäällikkö voi täydentää budjetillistä varausta ajankohdalla jolloin kyseinen työvaihe voitaisiin mahdollisesti suorittaa. Hän osoittaa hallinnoimansa tiimin alustavasti suorittamaan työvaiheen määriteltynä toteutusajankohtana. Näiden budjetillisten ja tilauksiin perustuvien resurssivarausten pohjalta pystytään luomaan raportteja tiimien reaaliaikaisista saatavuuksista.

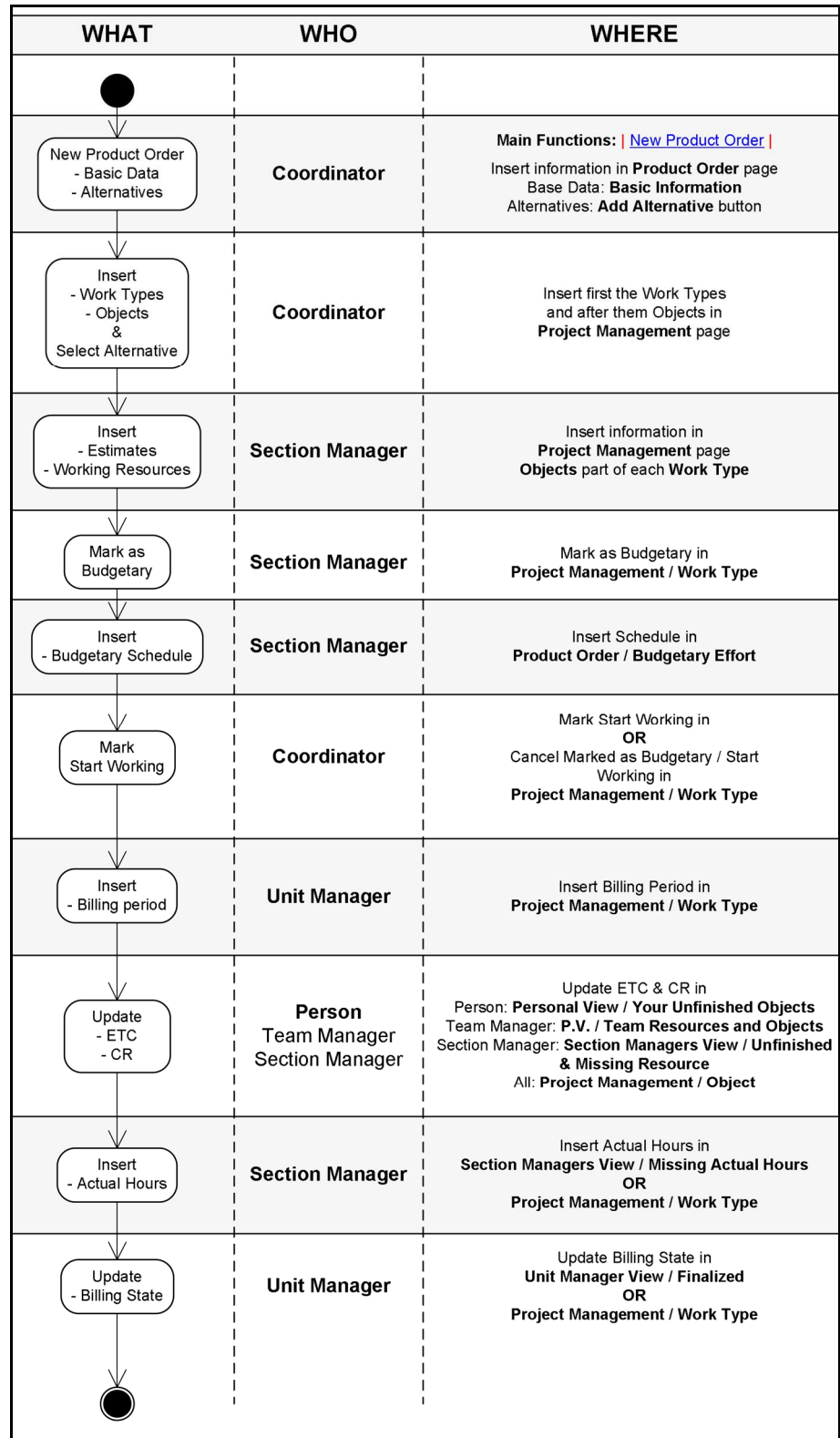
Kuudennessa vaiheessa tuotekoordinaattori on tarjonnut asiakkaalle tämän pyytämää tarjoutta työmääräarvioituna ja alustavasti aikataulutettuna. Saadessaan asiakkaan tilauksen tai tarjouksen hylkäämisen, tuotekoordinaattori päivittää tiedon järjestelmään. Asiakkaan tilatessa tuotekoordinaattori merkitsee tuotetilauksen tilatuksi, jolloin jokainen tuotetilaukseen liittyvä työvaihe aktivoituu suoritettavaksi määritellyn aikatauluvarauksen mukaisesti. Asiakkaan hylätessä tuotekoordinaattori peruuttaa tuotetilauksen, jolloin työvaiheiden budjetilliset resurssivarausten vapautuvat ja resurssit ovat jälleen käytettävissä.

Seitsemännessä vaiheessa yksikön päällikkö täydentää tuotetilausta yksityiskohtaisemmilla laskutustiedoilla.

Kahdeksannessa vaiheessa tiimin jäsen, tiimin päällikkö tai jaospäällikkö päivittää vastuulleen osoitettujen työvaiheiden edistymisen. Työvaiheen edistymisestä kirjataan päivittäin jäljellä oleva työtuntimäärä, joka työn suorittamiseen kuluu. Lisäksi edistymisestä kirjataan prosentuaalinen arvio siitä, miten paljon kyseisen työvaiheen toteutuksesta on suoritettu. Kun työvaiheen jäljellä oleva työtuntimäärä saavuttaa nollan ja työstä on suoritettu 100 %, työvaihe katsotaan järjestelmätasolla suoritetuksi. Tällöin järjestelmä lukitsee työvaiheen ja merkitsee sen suoritetuksi.

Yhdeksännessä vaiheessa jaospäällikkö täydentää suoritettujen työvaiheiden tietoja niiden toteutuneilla työmäärillä. Alkuperäistä arviota ja toteutunutta tuntimäärää vertaillaan pystytään arvioimaan työmääräarvioijan arviointitarkkuutta.

Kymmenennessä vaiheessa yksikön päällikkö viimeistelee tuotetilaukset toteutuneilla laskutustiedoilla, jolloin tuotetilauksia katsotaan valmiiksi.



Kuvio 8: TE OMT-järjestelmän ydinprosessi

4.3.2 Ohjelmakoodi

Järjestelmän käyttämä PHP5 tukee olio-ohjelmointia, mutta toteutus päätettiin kuitenkin toteuttaa funktiopohjaisena ratkaisuna. Järjestelmää suunniteltaessa haluttiin nähdä nopeita tuloksia, tehdä nopeita muutoksia ja erilaisia versioita. Käytännössä tämä oli helpointa ja nopeinta toteuttaa luomalla yksittäisiä funktioita suorittamaan haluttu toiminto. PHP:n olio-ohjelmointi on kuitenkin usein liian raskas tapa toteuttaa yksinkertaisia toiminnallisuuksia (Schlossnagle 2004: 37). Myöhemässä vaiheessa funktiota ryhmiteltiin käyttötarkoituksensa mukaan erillisiin tiedostoihin. Tästä esimerkkinä mm. *TE-OMT-SQL-add-modify-functions.php* tiedosto, joka sisältää ainoastaan MySQL-tietokannan datanhallintaan liittyviä tiedon lisäykseen ja muokkaukseen liittyviä funktioita.

Yksinkertaisetkin funktiot kommentoitiin ja toteutettiin hyvän ohjelmointitavan mukaan rivittäen ja sisentäen ohjelmakoodia selvyuden parantamiseksi. Pitkät rivit jaettiin useammalle riville, ehtolauseet sisennettiin ja muuttujien alustaminen asemoitiin yhtä suuruusmerkkien kanssa tasan. (Schlossnagle 2004: 10 - 13.) Tästä esimerkki kuviossa 9, jossa esitellään funktiota *updaterowchanged*. Kyseistä funktiota kutsutaan, kun tietokannan taulun riviä muokataan. Funktio kasvattaa kyseisen rivin *rowchanged* arvoa yhdellä. Näin pystytään tarkkailemaan kuinka monesti kutakin riviä on muokattu ja kohdentamaan mahdollinen raskas kuormitus muokkausten osalta.

```

### Function "updaterowchanged" updates the rowchanged number of
### selected tables "rowchanged" value. No return value.
function updaterowchanged($table, $identifiername, $identifierid) {

    ### Include the WRITE database access variables:
    ### $host, $user, $password, $database
    include('TE-OMT-dbwrite.php');

    ### Connect to the database
    mysql_connect($host, $user, $password)
    OR DIE ("There is no access to database. Try again later");

    ### Select the wanted database
    mysql_select_db($database);

    ### Form the SQL query
    $sqlcommand = "UPDATE ";
    $sqltable   = " ".$table." ";
    $sqlset     = " SET rowchanged = rowchanged+1";
    $sqlwhere   = " WHERE ".$identifiername.
                 " = ".$identifierid." LIMIT 1 ";

    $sqlorder  = "";
    $sql       = $sqlcommand.$sqltable.$sqlset
                 .$sqlwhere.$sqlorder;

    ### Perform the SQL update query
    mysql_query($sql);

    ### Close the database connection
    mysql_close(mysql_connect($host, $user, $password));

    ### Nothing to return, end of function.
}

```

Kuvio 9: Esimerkki PHP-funktion rakenteesta

Funktio toteutettiin dynaamiseksi kohdetaulun ja rivitunnisteen osalta, jolloin se on helposti uudelleenkäytettävissä. Järjestelmän muutkin funktiot toteutettiin mahdollisimman dynaamisiksi, jolloin monimutkaisia toiminnallisuuksia sisältävät funktiot pystyttiin hyödyntämään useampaan kertaan.

4.3.3 Tietokanta ja kyselyt

Tietokantakyselyissä noudatettiin samoja hyviä tapoja, kuin PHP-ohjelmakoodissa. Kyselylauseissa tämä toteutettiin kirjoittamalla kaikki avainsanat isoin kirjaimin, aloittamalla avainsanat uudelta riviltä ja sientäen lauseita selkeyden lisäämiseksi (Schlossnagle 2004: 14). Tauluille ei kuitenkaan annettu erillisiä lyhenteitä, sillä useassa kyselyssä tarvittiin yli kymmentä taulua. Näissä tilanteissa selkeyden säilyttämiseksi oli parempi kirjoittaa taulujen nimet kokonaisuudessaan. Tätä käytäntöä sovellettiin kaikissa kyselyissä.

Esimerkkinä on esittely tietokantakysely, jossa halutaan listata tiimien työkuorma halutulta kuukaudelta ja joissa haluttu työntekijän työskentelee kuviossa 10.

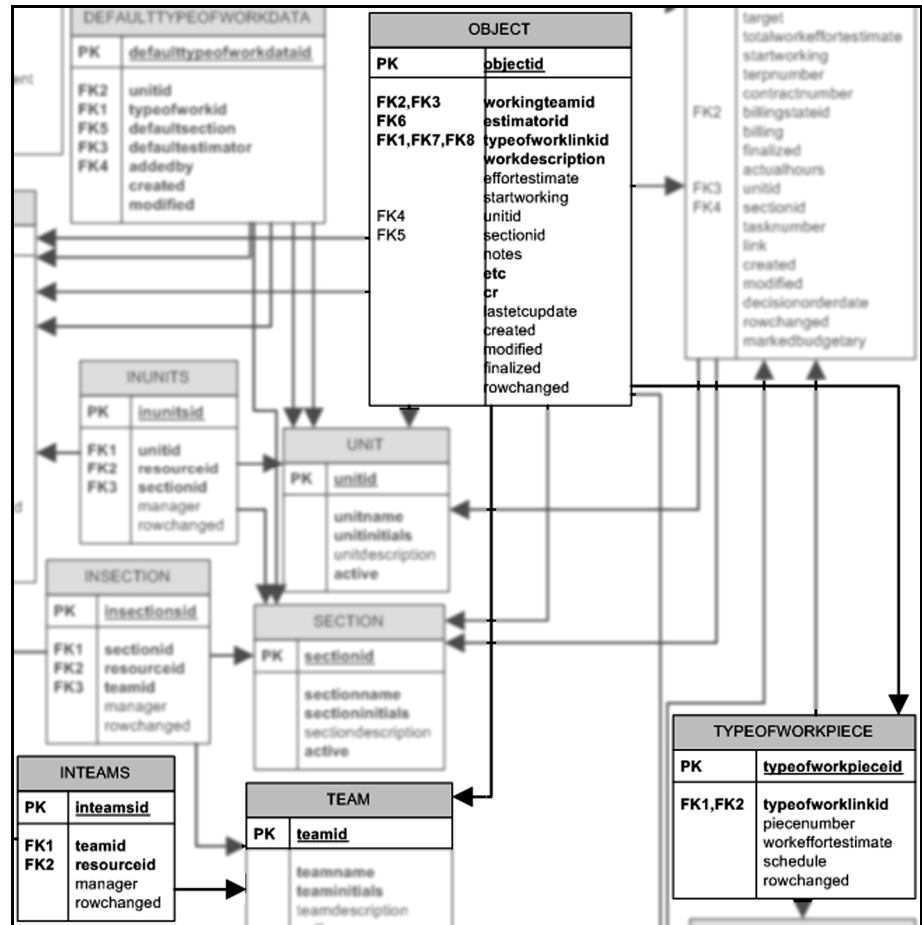
```

### Selecting all the effort estimates from wanted month of the
### Teams that the selected Resources is a member of.
$sql = '
SELECT DISTINCT
    TYPEOFWORKPIECE.typeofworkpieceid
        AS "TypeOfWorkPieceID"
    , OBJECT.effortestimate
        AS "Estimate"
FROM
    TYPEOFWORKPIECE LEFT JOIN OBJECT
    ON (TYPEOFWORKPIECE.typeofworklinkid
        = OBJECT.typeofworklinkid)
WHERE
    OBJECT.workingteamid = ANY
    (
        SELECT INTEAMS.teamid
        FROM INTEAMS
        WHERE INTEAMS.resourceid= ".$resourceid."
    )
AND TYPEOFWORKPIECE.schedule >= ".$currentstartstamp."
AND TYPEOFWORKPIECE.schedule <= ".$currentendstamp."
';

```

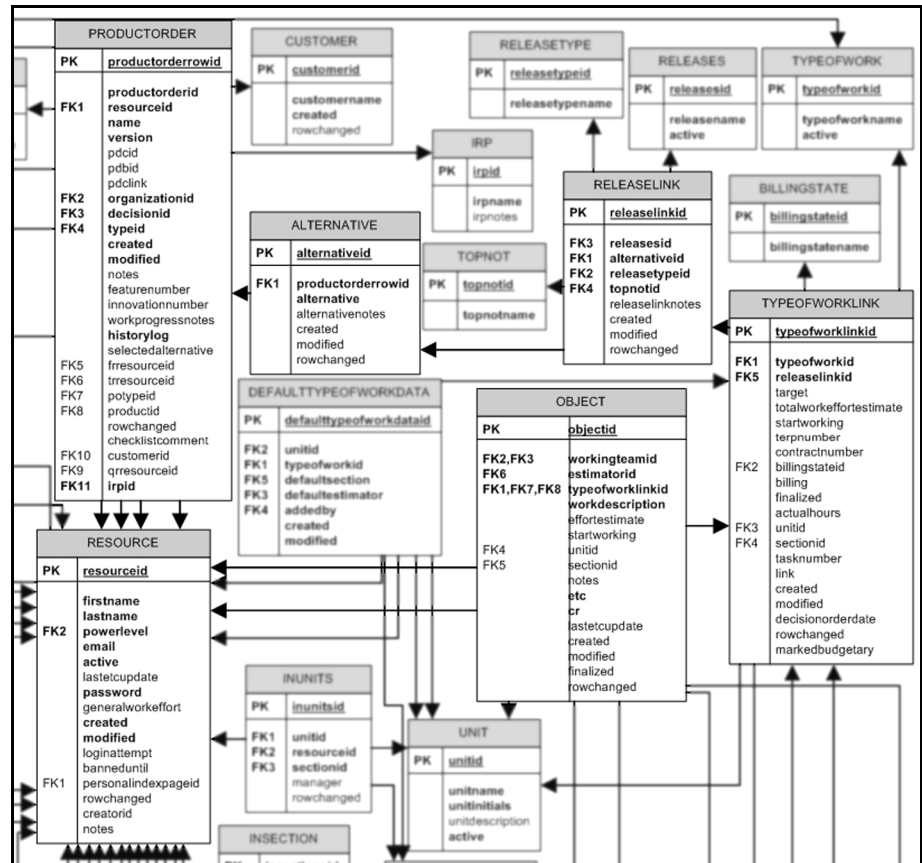
Kuvio 10: Esimerkki MySQL-kyselystä

Edellisen esimerkkikyselyn osalta kuvio 11 esittelee tietokannan rakennetta. Tietokantakuvaus toteutettiin Microsoft Visio 2007 Pro:lla ja vaikka kuvio ei piirrä yhteyksiä viivaa tarkemmin, on ne kuitenkin määritelty yhteyskohtaisesti. Näitä yhteyksiä pystytään siis tarkastelemaan ja muokkaamaan kyseisen ohjelmiston avulla.



Kuvio 11: Tietokantarakenne edellisen esimerkkikyselyn osalta

Lopullinen tietokanta sisälsi kaiken kaikkiaan 58 taulua, jotka muodostivat lumihutalemallisen rakenteen. Lumihutalemallia hyödyntämällä pystyttiin normalisoimaan tähtimallisen rakenteen dimensiot (Hovi 2005: 139 - 140). Vaikka tietokannassa olevien taulujen määrä on suuri, tarvitaan näistä tauluista vain 6 järjestelmän keskeisten tietojen kirjaamiseen ja toiminnan ylläpitämiseen. Nämä kuviossa 12 esitellyt kuusi faktataulua sisältävät keskeisimmät tiedot tilauksista, tarkennetuista työvaiheista, sekä tiedot työvaiheen toteuttajasta. Lähes kaikki loput taulut toimivat dimensiotaulujen roolissa parantaen suorituskykyä staattisen tiedon varastoina.



Kuvio 12: Tietokannan keskeiset taulut

Järjestelmän keskeisimmät taulut ovat *PRODUCTORDER*- sekä *OBJECT*-taulu, jotka sisältävät ydintiedot kustakin tilauksesta ja tarkennetusta työtehtävästä. *RESOURCE*-taulu toimii faktatauluna järjestelmään sisään kirjauduttaessa. Jokainen järjestelmään luotu henkilöresurssi on siis samalla myös järjestelmän käyttäjä.

4.3.4 Tyylien hallinta

Järjestelmän ulkoasu luotiin dynaamiseksi CSS-tyylimäärittelyn sekä erillisen värimuuttujatiedoston avulla, jolloin ulkoasu on helppo muokata. Värimuuttujatiedostoon tallennettiin TietoEnatorin oman tyylin määrittelemät värikoodit, joita hyödynnettiin järjestelmässä TietoEnatorin sisäisen ohjeistuksen mukaisesti. Värimuuttujatiedoston värikoodeja hyödynnettiin niissä ohjelmakoodin osissa, joissa CSS-tyylejä ei voitu hyödyntää. CSS-tyyleillä määriteltiin lomakkeiden, tekstien ja kappaleiden yleisluotoisia asetuksia pyrkien välttämään yhteensopimattomuudet eri selainten välillä (Budd & Collison & Moll 2007: XVI). Kuviossa 13 esitellään ote CSS-tiedostosta, jossa määritellään otsikkotason tyyli tekstin ja tausta-asetusten osalta.

```

.headingwithbackground1 {
  font-family:    Arial, Veranda, Sans Serif, Tahoma;
  font-weight:    bold;
  text-decoration: none;
  color:          #000000;
  font-size:      16px;
  align:          center;
  background-image: url(pic/TE-fade-h-rl-900-
                  lightgreywhite.gif);
  background-repeat: repeat-y;
  background-color: #FFFFFF;
}

```

Kuvio 13: Esimerkki CSS-tiedostosta

4.3.5 Optimoidut käyttöliittymät

Jokainen käyttöliittymä sisältää yleiset toiminnallisuudet ja listaukset tilauksista ja niihin liittyvistä perustiedoista. Perusnäkömöt poikkeavat vain hieman toisistaan. Osa tiedoista on alemman tason käyttäjien näkymistä poistettua ja mitä ylemmäs hierarkiassa siirrytään, sitä enemmän näkyvyyttä ja muokkausmahdollisuuksia sallitaan. Näistä esimerkkinä tilaukseen liittyvät laskutuspäivä ja laskutustila. Kumpainkaan tieto ei näy normaalille työresurssille, mutta esimerkiksi jaospäällikkö-tason käyttäjä näkee päiväyksen sekä tilan, mutta yksikön päällikkö -tason käyttäjä pystyy tämän lisäksi vielä muokkaamaan näitä tietoja. Teknisesti tämä on toteutettu niin, että järjestelmän tulostamat tietokentät on toteutettu funktion avulla, jolle annetaan parametrina tiedon näkemiseen ja muokkaamiseen tarvittavat tehotasot. Jokaisella järjestelmän käyttäjällä on määritelty henkilökohtainen tehotaso. Näin jokainen järjestelmässä esitettävä tieto pystytään yksilöllisesti osoittamaan tietyn käyttäjätason nähtäväksi ja muokattavaksi.

Käyttäjärooleille luodut optimoidut käyttöliittymät sisältävät perusnäkömysten lisäksi jokaiselle roolille ominaiset työkalut ja raportit. Kuviossa 14 esitelty jaospäällikön näkymä listaa jaospäällikön hallinnoimat tiimit. Tiimeistä listataan halutun kuukauden työresurssien saatavuusarvio tunteina sekä yleinen saatavuusarvio, jota käytetään raporteissa täsmennetyt kuukausiarvion puuttuessa. Näitä kahta arvoja voidaan tämän näkymän kautta myös muokata. Lisäksi näkymässä esitetään raportti tiimin työkuormasta suhteutettuna saatavuuteen. Järjestelmä laskee työkuorman nykyhetkestä käyttäjän valitseman kuukauden loppuun saakka. Käytännössä laskukaava ottaa huomioon vain valittuun kuukauteen ja siitä taaksepäin ajoitetut keskeneräiset työvaiheet. Näiden työvaihteiden jäljellä olevat tuntimäärät lasketaan tiimin työkuormaksi. Saatavuus lasketaan valitusta kuukaudesta taaksepäin aina nykyhetkeen saakka niin, että kokonaisten kuukausien osalta käytetään kyseisen kuukauden saatavuustuntimäärää ja mikäli tätä ei ole kyseiselle kuukaudelle asetettu käytetään yleistä saatavuusarviota. Keskeneräisen kuukauden kohdalla lasketaan kuukausiarviosta osa joka vastaa jäljellä olevan kuukauden osuutta koko kuukauden saatavuudesta.

Logout Marko Kuusniemi Sun 21.09.2008 05:09:44 | Search | Personal view | Coordinator view | Section MGR view | Unit MGR view | Unit MGR checklist | New Product Order | Admin View | Hide All | Show All

SECTION MANAGER VIEW:

| Unfinished & Missing Resource | Missing Actual Hours | Active Resources | Active Team Resources | Capacity Forecast | Resources Basic Information | My Messages |

ACTIVE TEAM RESOURCES: -

Selected month and year: 1 2009 open

Team:	for 1-2009	General	Workload:		Save	View
Contact Point	280.00	311.00	- 19 / 1524 h	1 %	save	View
development team 1	350.00	372.00	- 830 / 1838 h	45 %	save	View
development team 2	400.00	383.00	- 1188 / 1932 h	61 %	save	View
development team 3	375.00	394.00	- 900 / 1951 h	46 %	save	View
TOTALS: 4			2937 / 7245 h	41 %		

Kuvio 14: Esimerkki jaospäällikön näkymästä

4.4 Tuki scrum-timien resurssienhallintaan

Saatavuuden osalta järjestelmään kirjataan tiimikohtaisesti oletusarvio tiimin keskimääräisestä kuukausisaatavuudesta sekä täsmälliset kuukausiarviot niiden ollessa saatavilla. Oletusarviota käytetään silloin, kun kyseisen kuukauden tarkkaa saatavuusarviota ei ole vielä saatavilla. Täsmällinen kuukausisaatavuus on suoraan verrannollinen scrummenetelmissä käytettävän sprintin arvioituun tiimin tuntisaatavuuteen. Tätä arvoa voidaan siis suoraan käyttää tietojärjestelmässä tiimin kyseisen kuukauden saatavuutta kirjattaessa. Tämän arvon avulla pystytään paremmin hallinnoimaan tulevaa tilauskantaan saatavuuden puitteissa. Kuukausittaiset yli- tai alimyyntit pystytään näin pitämään minimissä ja toimimaan täyttää tuotantokapasiteettia hipoen. Kun täsmällinen kuukausisaatavuus kullekin kuukaudelle on tiedossa, se kirjataan järjestelmään. Tällöin saatavuuden ja tilauskannan suhde tarkentuu entisestään.

Kuviossa 14 esitelty jaospäällikön näkymän kautta pystytään helposti syöttämään ja muokkaamaan tarkkoja kuukausisaatavuuksia, sekä yleisiä saatavuustuntimääriä. Tämän lisäksi näkymästä pystytään yhdellä silmäyksellä havaitsemaan kunkin tiimin työkuorma nykyhetkestä valitun kuukauden loppuun saakka.

Resurssienhallinnan osalta toteutettu tietojärjestelmä tarjoaa mahdollisuuden hallita saatavilla olevaa resurssimäärää tiimeittäin ja mahdollisuuden muodostaa raporteja yli- tai alijäämistä, saatavuutta verrattaessa kuukausikohtaiseen tilauskantaan. Eräs keskeisimmistä raporteista on juuri kuukausitason saatavuus, jonka toteutusta voimme tarkastella kuvioista 15.

SECTION CAPACITY FORECAST: -										
Selected month and year: 10 2008 open										
development -										
CAPACITY FORECAST: Partly based on general effort Completely based on general effort										
Year:	2008			2008		2008	2009			
Month:	7	8	9	10	11	12	1	2	3	4
Resources:	3736	3736	3736	3430	1660	3525	3630	3736	3736	3736
Demand:	600	0	0	0	600	1600	0	0	0	0
Budgetary:				2734	866	2732	1866			
Total:	+3336	+3736	+3736	+696	+194	-807	+1764	+3736	+3736	+3736

Kuvio 15: Raportti kuukausisaatavuudesta

4.5 Toteutetun järjestelmän arviointi ja jatkokehitys

Toteutettu tietojärjestelmä onnistui vastaamaan priorisoituja tarpeita hyvin jo ensimmäisen puolenvuoden kehityskaaren jälkeen. Ensisijaisen tärkeänä pidetty resurssien reaaliaikaisen saatavuuden ja työkuorman hahmottaminen ja hallinnointi onnistui odotettua paremmin, toteutettu raportoinnin ollessa nopea ja selkeä. Lisäksi tilausten ja suoritettavien työvaiheiden muokkaaminen onnistuu helposti, nopeasti ja kätevästi käyttäjäroolikohtaisesti optimoitujen käyttöliittymien avulla.

Toiminnallisten vaatimusten osalta järjestelmä tuottaa ongelmatilanteissa selkeät virheilmoitukset. Tämän lisäksi järjestelmän käyttäjien kirjautuminen on toteutettu turvallisesti ja tarvittaessa lukittavaksi mekanismiksi.

Teknisten vaatimusten osalta järjestelmä kehitettiin helposti laajennettavaksi SQL-pohjaiseksi tietojärjestelmäksi. Lisäksi tietokannan rakenne ja siihen kohdistuvat kyselyt suunniteltiin ja toteutettiin optimoidusti niin, että järjestelmä pystyy kevyesti palvelemaan useaa samanaikaista käyttäjää.

Jatkokehitys alkoi heti ensimmäisen puolenvuoden kehityskaaren jälkeen, jolloin keskeisimmät osiot oli toteutettu. Uudet toiminnallisuudet määriteltiin ja priorisoitiin senhetkisen tarpeen mukaan ja näiden toteutus aloitettiin toiminnallisuus kerrallaan. Järjestelmän ylläpito ja pienet muutostyöt ovat olleet olennainen osa koko järjestelmän kehityskaarta, sen alusta saakka.

Uusina vaatimuksina mainittakoon muutamina mm. lomalistalaajennus, vuosittaisten ylläpitosopimusten hallinta, järjestelmäkäyttäjän sijaisuuksien hallinta, muutosten historiatiedon hallinnointi sekä uusien tehtäväkohtaisten hallintanäkymien ja raporttien luominen.

Tiimien muuttuessa dynaamisemmiksi, järjestelmä on muutettava tiimipohjaisesta toteutuksesta yksittäisten henkilöresurssien hallintajärjestelmäksi. Tämän muutoksen toteuttaminen aiheuttaisi isoimman työn raporttien uusimisessa, jotka pohjautuvat perusajatukseen jossa tiimit suorittavat työvaiheita. Muutoksen jälkeen raporttien tulisi pohjautua perusajatukseen, jossa yksittäiset henkilöresurssit suorittavat työtehtäviä. Yksilöresurssien ollessa jo entuudestaan tietokannassa kirjautumisen ja perustietojen ansiosta, muutos ei vaadi lisätauluja tietokantaan vaan ainoastaan muutoksen työvaiheen ja sen suorittajan väliseen linkitykseen.

Mahdollisten teknisten muutosten osalta voitaisiin ajatella ohjelmakoodin muuttamista oliopohjaiseksi, joka vaatisi samalla myös arkkitehtuurillisia muutoksia. Lisäksi tietokantaan voitaisiin toteuttaa kyselyitä stored procedure -tekniikkaa hyödyntämällä. Näin tietokantaan suunnatun liikenteen määrä vähenisi, kun laajoja kyselyitä pystyttäisiin suorittamaan lyhyellä proseduurikutsulla. Tarvittaessa nämä muutokset mahdollistavat myös muiden tekniikoiden vaivattoman käyttöönoton, jos jostain syystä haluttaisiin kääntyä selainpohjaisesta PHP-toteutuksesta johonkin muuhun. Valmiin luokkakaavioinnin pohjalta luokat ja niiden metoditoteutukset olisi helppo muuntaa mille tahansa ohjelmointikielelle. Lisäksi tietokantaproseduureja voitaisiin käyttää suoraan esimerkiksi reaaliaikaisten raporttien hakemiseen myös järjestelmän ulkopuolisista sovelluksista.

5 Johtopäätökset

Opinnäytetyön tuloksena syntyneen järjestelmän ohjelmistoprojekti oli suorituksen osalta eräänlainen yhdistelmä protoilusta, jatkuvasta ideointipalaverien ketjusta ja agile-menetelmistä. Koska agile-menetelmiä ei ollut vielä projektin alkaessa otettu TietoEnatorin sisällä käyttöön, sen tarjoamia tekniikoita ei osattu opinnäytetyön yhteydessä hyödyntää. Oma työskentelyni tämän projektin osalta mukaili varsin pitkälle agile-menetelmiä. Tästä syystä varsinaisen muutoksen saapuessa TietoEnatorilla tunsin oloni varsin mukavaksi.

Työn ainoana tekijänä, minulla oli vapaus määritellä, suunnitella ja toteuttaa järjestelmä hyvin pitkälle haluamallani tavalla. Vapauden mukana tuli tietysti myös vastuu projektin kokonaisvaltaisesta onnistumisesta. Projekti eteni hyvin pitkälle aikaisempien kokemusteni mukaisesti, eli järjestelmän tilaajilla ja loppukäyttäjillä ei koskaan ole tarpeeksi aikaa paneutua ohjelmiston suunnitteluvaiheessa. Jatkuvat ideointipalaverit ja välitulosten esittämiset toimivat kuitenkin tässäkin projektissa hyvin. Väärinkäsitykset ja virhearvioinnit saatiin näin ollen korjattua helposti ja nopeasti tuotantovaiheen edetessä. Suunnittelun ja selkeän arkkitehtuurin tarpeellisuuden alleviivaamista en voi silti vähätellä edes agile-menetelmien maailmassa.

Vastuu työssäni ulottui myös ohjelmistotuotannon rajojen ulkopuolelle. Alun kick-off -palaveria lukuun ottamatta järjestin projektin kaikki palaverit, varaten tilat ja kutsuen tarvittavat henkilöt kaikille sopivaan aikaan paikalle. Kokousjärjestelyjen lisäksi vastasin laitehankinnoista ja ohjelmistoasennuksista järjestelmän tarvitseman palvelimen osalta. Järjestelmän käyttöönoton jälkeen myös ylläpito palvelimesta, sen ohjelmistoista ja tietysti toteutetusta järjestelmästä osoitettiin minulle.

Annetusta vastuusta ja luottamuksesta voin henkilökohtaisesti olla kiitollinen. Sain suunnitella ja toteuttaa laajan tietojärjestelmän täysin tyhjältä pöydältä, täyteen käyttövalmiuteen ja ylläpitoon saakka. Tekninen osaamiseni on projektin aikana karttunut huomattavasti. Suunnittelun ja arkkitehtuurin ymmärrys kasvoi myös. Kaikesta oppimastani on jo nyt ollut hyötyä uusissa haasteissani.

Lähdeluettelo

- Highsmith, Jim 2001. History: The Agile Manifesto. [online] [viitattu 16.4.2008].
<http://agilemanifesto.org/history.html>
- Agile Alliance, The 2001. The Agile Manifesto. [online] [viitattu 16.4.2008].
<http://agilemanifesto.org/>
- Highsmith, Jim 2004. Agile Project Management Creating Innovative Products. United States of America: Addison-Wesley.
- Schwaber, Ken 2004. Agile Project Management with Scrum. Redmond, Washington: Microsoft Press.
- Schwaber, Ken 2002. Agile Software Development with Scrum. United States of America: Prentice Hall.
- Martin, Robert C. 2003. Agile software development: principles, patterns, and practices. Upper Saddle River, N.J: Prentice Hall PTR
- Shore, James & Warden, Shane 2008. The Art of Agile Development. United States of America: O'Reilly Media, Inc.
- Budd, Andy & Collison, Simon & Moll Cameron 2007. CSS tehokas hallinta. Helsinki: Readme.fi.
- Eriksson, Hans-Erik & Penker, Magnus 2000. UML. 3. painos 2002. Jyväskylä: IT Press.
- Haikala, Ilkka & Märijärvi, Jukka 2004. Ohjelmistotuotanto. 6. uudistettu painos 2004. Helsinki: Talentum.
- Hovi, Ari 2005. Tietokantojen suunnittelu & indeksointi. Jyväskylä: Docendo.
- Koskimies, Kai 2005. Ohjelmistoarkkitehtuurit. Helsinki: Talentum.
- Lahtonen, Tommi 2002. SQL. 1. painos. Jyväskylä: Docendo Finland Oy.
- Linjamaa, Tero 2001. XHTML. Jyväskylä: Docendo.
- Pohjolainen, Risto 2002. Tietojärjestelmien kehittäminen. Jyväskylä: Docendo.
- Schlossnagle, George 2004. Advanced PHP Programming. Indianapolis, Indiana: Developer's Library.
- Zandstra, Matt 2001. PHP Trainer Kit. Helsinki: Oy Edita Ab.