Opinnäytetyö (AMK)

Degree Programme in Business Information Technology

Business Information Systems Management

2010

Joonas Rivinoja

OHJELMISTOTUOTTEEN DOKUMENTOINTI

– kohdeyrityksenä Agenteq Solutions Oy

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Degree Programme in Business Information Technology | Business Information Systems Management

Joulukuu 2010 | Sivumäärä: 34 + 2 liitettä

Ohjaaja: Minna-Kristiina Paakki

Joonas Rivinoja

OHJELMISTOTUOTTEEN DOKUMENTOINTI

- kohdeyrityksenä Agenteq Solutions Oy

Agenteq Solutions Oy on kasvava ohjelmistoyritys, joka toimii Salossa ja Helsingissä. Yritys on Suomen markkinajohtaja kiinteistöalan ohjelmistojen tarjoajana sekä lukuisten muiden web-palvelujen kehittäjä. Yrityksen päätuotteita ovat kiinteistötietojärjestelmä Tampuuri sekä julkaisujärjestelmä Latomo.

Tämän opinnäytetyön tarkoituksena on tutkia ja arvioida yrityksen tämänhetkisiä ohjelmistotuotteiden dokumentointikäytäntöjä ja luoda toimivia ratkaisuja, jolla käytäntöjä voidaan kehittää entistä paremmaksi. Dokumentoinnin kehittämisellä haetaan helpompaa ja hallitumpaa tuotteiden ja tietojen ylläpitoa sekä tehokkaampaa tuotekehitystä. Ilman kunnollista dokumentointia ja käytäntöjä kommunikointi yrityksen sisällä ja ulospäin on selkeästi puutteellista.

Käytäntöjen yhtenäistämiseksi eri dokumenttityyppien kesken on luotava niille dokumenttipohjat sekä määrättävä niille toimenpideroolit. Yrityksen dokumentit voidaan jakaa kahteen ryhmään: tuotedokumentit ja projektidokumentit. Yhtenä tärkeänä kehittämisen osa-alueena on myös koodidokumentaatio ja sen automatisointi. Koodidokumentaatiolla pyritään helpottamaan kehitystiimien välistä yhteistoimintaa ja henkilöstön vaihtumista projektien kannalta.

Työ tehtiin ensin tutkimalla yrityksen nykyistä dokumentointikäytäntöjen tilannetta ja suunnittelemalla tulevia tarpeita varten uusia ratkaisuja. Käytössä oleville ja uusille tarvittaville dokumenteille luotiin pohjat, koodidokumentaatio otettiin osaksi kehitystyötä ja dokumenteille luotiin RACI-toimenpidelistat. Lopulta näitä testattiin käytännössä yhdellä projektilla.

Yrityksen dokumentointikäytännön kehitystyön aikana huomattiin selkeitä puutteita. Osa niistä saatiin korjattua edellä mainituilla toimenpiteillä, mutta osa tuloksista tulee esille vasta testausvaiheen jälkeen, kun järjestelmää ja käytäntöjä hiotaan ja henkilöstöä koulutetaan niiden käyttämiseen.

ASIASANAT:

dokumentointi, ohjelmisto, ohjelmointi, laadunvalvonta, tuotehallinta

BACHELOR´S THESIS | ABSTRACT

UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology | Business Information Systems Management

December 2010 | Total number of pages: 34 + 2 appendices

Instructor: Minna-Kristiina Paakki

Joonas Rivinoja

DOCUMENTATION OF SOFTWARE PRODUCT

- case company Agenteq Solution Oy

Agenteq Solutions Oy is a growing software company located in Salo and Helsinki. The company is the market leader in estate management software distribution in Finland and developer of many other web-based services. The main products of the company are estate information system Tampuuri and content management system Latomo.

The aim of this thesis is to study and evaluate the current state of documentation practices in software products and create working solutions that may enhance the practices even better. Development of documentation is done to make easier and more manageable product and information administering and more efficient product development. Without good documentation and practices the communication inside the company and outwards is clearly deficient.

To standardize the practices between different document types there has to be document templates and select responsibility roles for them. The documents of the company can be split in two: product documents and project documents. One important aspect of development is also code documentation and its automation. The purpose of code documentation is to make co-operation easier between the development teams and personnel changes for projects.

The thesis was done by first researching the current state of the documentation practices in the company and planning new solutions for upcoming requirements. Template documents for documents currently in use and new required ones were made, code documentation was taken as a part of development work and RACI responsibility matrix was created for used documents. In the end these were tested in practice with one project.

During the development of documentation standards for the company certain defects were noticed. Some of them have been corrected with the aforementioned measures but some of the results will become clear only after the testing phase when the system and standards are being polished and staff being trained.

KEYWORDS:

documentation, software, programming, quality control, quality assurance, product management

SISÄLTÖ

1 JOHDANTO 5
1.1 Yrityksen taustaa 5
1.2 Syyt aiheen valintaan 6
1.3 Työn tavoitteet 6
2 TEORIA 7
2.1 Dokumentoinnin hallinta yleisesti 7
2.2 Ohjelmistotuotteen dokumentointi yleisesti 9
2.3 Yleisesti dokumentoinnista 9
3 AGENTEQIN NYKYINEN TOIMINTATAPA 12
4 AGENTEQIN DOKUMENTOINTIKÄYTÄNNÖN KEHITYS 14
4.1 Ohjelmistotuotteen dokumentoinnin vaiheet 16
4.2 Koodidokumentointi 18
5 DOKUMENTOINNIN HALLINNOINTI AGENTEQISSÄ 22
6 TESTAUS 28
7 YHTEENVETO 30
8 LÄHTEET 32

TAULUKOT

1 JOHDANTO

Dokumentoinnilla pyritään pitämään yllä, seuraamaan ja helpottamaan tuotteiden kehitystä. Ohjelmistoalalla sovelluksia kehitetään jatkuvaa tahtia, joten on tärkeää, että niiden tiedot ovat ajan tasalla. Eri kehittäjien tehdessä töitä keskenään ja henkilöstön vaihtuvuuden kannalta dokumentaatio helpottaa tiedon saamista niin vanhoille kuin uusille kehittäjille. Ohjelmistosuunnittelussa dokumentoinnista on erilaisia standardeja ja suosituksia, mutta organisaatioissa on harvoin täydellistä ja yhdenmukaista käytäntöä dokumentaatioista. Usein käytännössä kaikki tehdään yritysten omia tarpeita vastaaviksi ja tapauskohtaisesti.

1.1 Yrityksen taustaa

Agenteq Solutions Oy on kasvava ohjelmistoyritys Salossa, jolla on Salon lisäksi toimipiste Helsingissä sekä Savonlinnassa. Kirjoitushetkellä vuonna 2010 henkilöstöä on 47, josta suurin osa sijaitsee Salossa. Agenteq on Suomen markkinajohtaja kiinteistöalan ohjelmistojen tarjoajana sekä lukuisten muiden web-palvelujen kehittäjä. Yritys toimii pääasiassa ohjelmistokehityksen parissa ja päätuotteita ovat kiinteistötietojärjestelmä Tampuuri sekä julkaisujärjestelmä Latomo. Näiden lisäksi yritys toteuttaa myös asiakaskohtaisia palveluratkaisuja ja verkkosivustoja. (http://www.agenteq.fi)

Henkilöstö koostuu pääosin ohjelmistosuunnittelijoista, jotka suunnittelevat, ohjelmoivat ja hallinnoivat ohjelmistojen eri osa-alueita. Eri tuotteiden parissa työskentelevät kehittäjät muodostavat tiimejä, jotka itsessään voivat tehdä useampiakin projekteja eri kokoonpanoilla. Jokaisella tiimillä on myös tiimivetäjä, jotka useimmiten toimivat samalla projektipäälliköinä.

Ohjelmistoja kehitetään jatkuvassa tahdissa joko asiakkaiden tilauksista tai yrityksen sisäisestä tarpeesta. Suurimmista kokonaisuuksista luodaan omat

projektinsa ja näille nimitetään projektipäällikkö sekä tarvittava määrä kehittäjiä tekemään omat osuutensa. Myös tiimit keskenään joutuvat silloin tällöin jakamaan töitään keskenään johtuen sovellusten monimuotoisuudesta, keskinäisistä riippuvuuksista ja käytössä olevien resurssien kannalta.

Ohjelmistojen kehityksessä käytetään avuksi versionhallintaa. Versionhallinta toimii keskitettynä paikkana yrityksen ohjelmistojen lähdekoodien säilytyksessä, jakamisessa ja muutoshistorian pitämisessä. Versiohallinnan avulla pystytään ohjelmistoon yhtäaikaisesti kehittämään eri ominaisuuksia eri henkilöiden toimesta pitämällä kehitysversiot erillään.

1.2 Syyt aiheen valintaan

Agenteq on projektien ja ohjelmistokokonaisuuksien monimuotoisuuden lisääntyessä huomannut tuotteiden dokumentoinnin tarpeen kasvaneen. Ohjelmistojen kehityksessä ja ylläpitämisessä dokumentoinnin puute haittaa varsinkin laajemmissa, useamman henkilön projekteissa, joissa tietoa jaetaan muiden kesken ja tuotteella on asiakkaan laatimat laatukriteerit ja määrittelyt. Määrittelyjä, ohjeita ja koodidokumentaatiota voidaa myäs osaltaan käyttää hyväksi laadunvarmistuksessa. Dokumentointiprosessin kehityksen käynnistäminen katsottiin aiheelliseksi ja sain yritykseltä toimeksiannon opinnäytetyön muodossa.

[Ei julkaistavaa tietoa]

1.3 Työn tavoitteet

Opinnäytetyön pääasiallisena tarkoituksena hakea eri ratkaisuja, joilla ohjelmistojen dokumentoinnin määrää ja laatua voidaan nostaa kuitenkaan haittaamatta yrityksen muuta normaalia prosessia. Ratkaisujen tutkiminen rajoitetaan tiettyjen tarpeiden täyttämiseen resurssien puitteissa, ja ne olisi voitava toteuttaa nykyisin yrityksessä käytettävillä tekniikoilla ja mahdollisimman joustavasti, jotta niitä voidaan jatkokehittää ja vaihtaa toisiin tarpeen vaatiessa.

Tavoitteet on katsottu saavutetuiksi, kun tarpeet on kartoitettu ja kehityksen suunta on selvä. Koko prosessin valmistuminen aikarajan puitteissa ei ole realistista, mutta tiettyjä asioita on tarkoitus saada myös testausvaiheeseen.

Aihe on rajattu yrityksen dokumentointikäytäntöihin, dokumenttityyppien sisältöihin, dokumentoinnin hallintaan yrityksen tietojärjestelmissä sekä koodidokumentaatioon ja sen automatisointiin. Työssä ei käsitellä dokumenttien tyylejä tai rakennetta, ohjelmistojen teknisten piirteiden dokumentointia, kuten ER-kaavioita, eikä dokumenttien jakamiseen tai tuhoamiseen liittyviä prosesseja.

2 TEORIA

2.1 Dokumentoinnin hallinta yleisesti

Tuotteista syntyneillä dokumenteilla on oltava jokin säilytyspaikka. Nykyään yrityksissä dokumentit säilytetään suurelta osin sähköisessä muodossa, kuten dokumenttitiedostoissa (doc, pdf), taulukoissa (excel) jne. jossain keskeisessä tietovarastossa. Tietovaraston valintaan vaikuttaa henkilöstön koko, tekninen osaaminen, tietomäärät, resurssit sekä joissain tapauksissa myös ulkopuoliset vaatimukset. Dokumentoinnin hallinnan tarkoituksena on pitää organisaatioiden materiaalit helposti saatavilla, turvassa ja noudattaa organisaation asettamaa tietoturvapolitiikkaa. Helpolla saatavuudella sallitaan ihmisten pääsy heidän tarvitsemiinsa materiaaleihin helposti, esimerkiksi keskitetyllä palvelimella ja helpolla käyttöliittymällä. Materiaalien turvallisuus taataan säännöllisillä ja automaattisilla varmuuskopioilla. Tietoturvapolitiikalla tarkoitetaan muun muassa materiaalien lukuoikeuksien rajaamista tietyille henkilöille, tietylle verkolle ja ulkopuolisten pääsyn estämistä sisäisiin dokumentteihin.

Nykyään tietovarastoja pidetään useimmiten niille tarkoitetuilla keskitetyillä palvelinjärjestelmillä. Yksi yleinen esimerkki on Windows Server ja Sharepoint Server -yhdistelmä, joka voidaan yhdistää Windowsin Active Directoryyn ja näin

hallita henkilöstön pääsyä eri järjestelmiin keskitetysti. Sharepoint Serverin päälle voidaan räätälöidä tarpeiden mukaan sivustoja, esimerkiksi yrityksen kotisivut, intranet yrityksen sisäisten tietojen hallintaan ja muita dynaamisia sivustoja. (sharepoint.microsoft.com)

2.2 Ohjelmistotuotteen dokumentointi yleisesti

Yleisesti ottaen dokumentointi on yksi tärkeimmistä osista ohjelmistotuotteen elinkaarta, mutta toisaalta usein myös suuresti laiminlyöty. Dokumentoinnista käy ilmi, mikä on ohjelmiston tarkoitus, toiminnat sekä mahdollisesti antaa kehittäjille olennaista tietoa kehitystä varten. Ulkopuolisille tahoille se tarkoittaa useimmiten medialle markkinointimateriaalia ja loppukäyttäjälle käyttöohjeet.

Ennen kuin tuote varsinaisesti tilataan asiakkaan toimesta, asiakas listaa tietyt tärkeimmät vaatimukset sekä business casen. Näiden pohjalta asiakas voi kysyä tarjousta useammalta toimittajalta ja valitsee tiettyjen kriteerien pohjalta yhden toimittajan. Sen jälkeen tuotteen tilaaja ja toimittaja luovat yhdessä toiminnalliset määrittelyt, joiden pohjalta voidaan luoda halutunlainen tuote. Tuotteistuksen ohella syntyy muita erilaisia dokumentteja, kuten teknisiä määrittelyjä ja muita valmistajan sisäisiä materiaaleja, joita tarvitaan niin jatkokehityksessä kuin muiden materiaalien täsmentämisissä.

Ohjelmiston kehityksessä on olemassa myös koodidokumentaatio. Koodidokumentaatiolla tarkoitetaan ohjelmiston koodin tekemisestä luettavaan muotoon – usein sähköisesti – jossa käy ilmi ohjelman toiminta sisäisesti. Ymmärrettävyyden selventämiseksi koodiin lisätään kommentteja, joissa selvitetään koodin osien toimintaa. Koodidokumentaation luomiseksi on olemassa monia käytäntöjä ja ohjelmia. Automaattinen koodidokumentaatio on taas käytäntö, jossa koodidokumentaatio luodaan automaattisesti – esimerkiksi ajastetusti – mahdollisimman vähäisellä lisätyöllä, kun käytäntö on saatu kehitettyä ja toimintaan.

2.3 Yleisesti dokumentoinnista

Dokumentoinnin tekemisellä on myös tarkoitus. Osaltaan sillä annetaan asiakkaalle tarvittavaa tietoa, toisaalta valmistajan organisaatio ja siihen liittyvät osapuolet pystyvät luotujen materiaalien avulla pitämään ajantasaista tietoa tuotteiden ylläpitoa ja kehitystä varten. Dokumentointi on siksi osaltaan myös

yksi viestinnän muoto. Osapuolia ovat muun muassa yrityksen johto, tuotekehittäjät ja markkinointihenkilöstö, jotka oman roolinsa mukaan tuottavat ja käyttävät erilaisia dokumentteja.

Dokumenttien luonnissa ei ole tiettyä yhtenäistä määrättyä tapaa, mutta useita eri tahoja – kuten IEEE- ja ISO -organisaatiot – suosittelevat tiettyjä metodeja. Dokumentit syntyvät joko omasta tai ulkoisesta tarpeesta ja toteutetaan resurssien ja osaamisen puitteissa. Kun asiakas tilaa tuotteen tai palvelun, valmistava yritys ja asiakas pyrkivät yhdessä määrittämään mahdollisimman tarkoin, mitä tuotteelta halutaan. Tätä kutsutaan toiminnalliseksi määrittelyksi. Mitä tarkempi määritys, sitä varmemmin kehittäjä pystyy arvioimaan tarpeet ja asiakas saa tilaamansa. Useimmiten määrittely kuitenkin muuttuu tuotteistuksen elinkaaren myötä, kun toimintoja lisätään ja hiotaan. Liian tarkan määrittelyn huonona puolena on, että tarvittavat muutokset vaativat enemmän resursseja, kun taas liian avoin saattaa poiketa asiakkaan alkuperäisestä tarpeesta paljonkin.

Tuotteen kehityksen aikana syntyy yritykselle myös omia sisäisiä materiaaleja, kuten teknisiä määrittelyjä, toimintakaavioita ja kehittäjien tuottamaa materiaalia. Näille on yleensä määrätty tiettyjä kohderyhmiä ja ylläpitäjiä. Loppukäyttäjiä varten luodaan ohjeistuksia, joiden avulla käyttäjät osaavat käyttää tuotetta. Ohjeet kirjoitetaan kohdeyleisö huomioon ottaen, eli jos käyttäjien oletetaan olevan teknisesti osaavia, ei kirjoiteta liian yksinkertaistetusti ja toisinpäin. Näiden lisäksi on olemassa markkinointimateriaalia, tuotteiden kehityssuunnitelmia ja sopimuksia. On yrityksen oma asia, kuinka vastuut jaetaan ja miten niiden suorittamisista huolehtii, mutta ilman järjestelmällistä perehtymistä asiaan ja käytäntöjen luomista koko organisaation laajuisesti asiat jäävät helposti taka-alalle ja tekemättä.

Esimerkki: Asiakkaalle tulee tarve tietylle ohjelmistolle ja siitä asiakas luo siitä vaatimuusmäärittelyn tai muun vastaavan dokumentin kysyäkseen tarjouksia yrityksiltä, jotka voivat mahdollisesti tarjota ohjelmistoa asiakkaalle. Kun asiakas on päätynyt tiettyyn toimittajaan, tilataan tuote yritykseltä. Ennen kehityksen aloittamista luodaan kokousten ja muun tiedonvaihdon pohjalta tuotteelle määrittelyt. Tilauksen jälkeen tehdään tuotantosopimus ja mahdollisesti myös ylläpitosopimus, joiden pohjalta molemmat osapuolet sitoutuvat tiettyihin asioihin tuotteen osalta. Kehityksen aikana päivitetään toiminnallista määrittelyä, koodidokumentaatiota sekä muita projektiin liittyviä dokumentteja. Riippuen tuotteen monimutkaisuudesta tai asiakkaan halusta luodaan myös ohje käyttäjälle, joka voi olla asiakkaan tahtomassa muodossa, kuten pdf-tiedosto tai web-sivusto.

3 AGENTEQIN NYKYINEN TOIMINTATAPA

Agenteqin ohjelmistokehityksessä käytetään suurelta osin nopean kehityksen mallia (rapid application development) ja ketterän kehitykseen (agile software development) kuuluvaa Extreme Programming -menetelmiä, sekä joissain tapauksissa myös Scrum-menetelmiä, riippuen projektin luonteesta ja asiakkaan vaatimuksista. Näillä menetelmillä ohjelmistoa kehitetään ja testataan samalla, ja kehityskaari täydentyy sen aikana.

[Ei julkaistavaa tietoa]

Dokumentointi näillä kehitysmalleilla on hieman erilainen verrattuna perinteisiin. Ohjelmistolle luodaan alussa alustava määrittely, ja ohjelmiston elinkaaren aikana se saattaa muuttua paljonkin. Riippuen dokumentaation tärkeydestä projektissa saattaa se projektin kehityksen aikana jäädä jälkeen varsinaisesta toteutuksesta, jolloin päivitys voi hankaloitua pidemmän ajan kuluttua kehittäjien siirtyessä muihin projekteihin.

Taulukko 1. Dokumentaatioprosessin kypsyysmallin yhteenveto (vapaasti käännetty, Marcello Visconti)

	Taso 1 Tarpeen mukaan	Taso 2 Epäyhtenäinen	Taso 3 Määritelty	Taso 4 Hallittu
Avainsanat	Kaaos, muuttuvuus	Standardit, tarkistuslistat, epäyhtenäisyys	Tuotteen arviointi Kehityksen määritys	Kehityksen arviointi Mittaus Hallinta Palaute Edistäminen
Tiivistetty kuvaus	Dokumentaatiolla ei korkeaa tärkeysastetta	Dokumentaatio tiedostettu tärkeäksi ja täytyy tehdä	Dokumentaatio tiedostettu tärkeäksi ja täytyy tehdä hyvin	Dokumentaatio tiedostettu tärkeäksi ja täytyy tehdä hyvin ja johdonmukaisesti
Avainkäytännöt	Tarpeen mukaan Ei tärkeää	Epäyhtenäiset käytännöt standardeissa	Dokumentaatiotason arviointi Dokumentaation hyödyllisyyden varmistuminen Kehityksen määritys	Kehityksen laadun arviointi ja mittaus
Avainmittarit	Dokumentaatio puuttuu tai vanhentunut	Standardit luotu ja tarkistuslistan käyttö	Software quality assurance (SQA)	Data-analyysi ja edistämistavat
Avainhaasteet	Luoda dokumentaatio-standardit	Laatukontrollin käyttö sisällölle Dokumentaation hyödyllisyyden arvioiminen Kehityksen määrittely	Prosessimittauksen luominen Hallinnan yhdistäminen kehitykseen	Tiedon keräämisen automatisointi ja analysointi Jatkuva optimoinnin tavoittelu

[Ei julkaistavaa tietoa]

4 AGENTEQIN DOKUMENTOINTIKÄYTÄNNÖN KEHITYS

Nykyinen dokumentointikäytäntö on lähinnä tilanteesta riippuvaa eikä sitä harjoiteta läheskään tarvittavalla tasolla, vaikka selkeä tarve sille on jo huomattu. Tätä varten olisi tärkeää luoda helposti ja luotettavasti käyttöönotettava dokumentaatiostardardi, joka toimisi tärkeänä osana varsinaista ohjelmistokehitystä. Agenteqissä on jo jonkin verran käytössä ohjelmistosuunnittelussa usein käytetyimmät dokumenttityypit, kuten toiminnalliset määrittelyt, ohjeet ylläpitäjille sekä käyttöohjeet loppukäyttäjille. Tarkoituksena on saada jatkossa kaikille tuotteille ja projekteille yhtenäiset dokumentointipohjat ja -käytännöt. Tämä tarkoittaa olemassa olevien dokumenttien tutkimista, tarpeiden kartoittamista, esimerkkien luomista ja testaamista.

Dokumenttityyppien roolien ja tehtävien jaotteluun sopii RACI-toimenpidelista. RACI on kirjainyhdistelmä sanoista *Responsible*, *Accountable*, *Consulted*, *Informed* ja on yksi tapa listata tehtäviä ja niiden tekemiseen liittyvien henkilöiden rooleja. Dokumenttia voidaan pitää tehtävänä ja sen tekemisestä vastuussa oleva henkilö merkitään *A*:lla ja niitä voi yhdellä dokumentilla olla vain yksi. Dokumentin tekeminen määrätään *R*:llä ja niitä voi olla useampia per dokumentti. Jos *R* tarvitsee dokumentin tekemisessä jotain apua, merkitään tämä kohde *C*:llä. Dokumentin tietoja tarvitsevat henkilöt merkitään *I*:llä.

Taulukko 2. Toimenpidelista kehityslista-dokumentille

Dokumentti	Johto	Tuote-päällikkö	Kehittäjä	QA	Tuki	Asiakas	Markki-nointi	Ylläpito
Kehityslista		A / R	R				I	

Dokumentit on jaettu kahdeen ryhmään; tuotedokumentit ja projektidokumentit. Tuotedokumentit ovat kohdistettuja tuotteeseen yleisesti (esimerkiksi Tampuuri). Projektidokumentit ovat eri projekteja varten luotuja dokumentteja, jotka elävät koko projektin elinkaaren ajan. Projekti voi olla esimerkiksi asiakaskohtainen ohjelmisto. Nämä kaksi ryhmää koostuvat itse erinäisistä dokumenteista. Tuotedokumenttien sisältö tulee suurelta osin projektidokumenteista, jotka yhdessä antavat tuotteelle ominaisuuksia ja samalla luovat koko tuotekokonaisuuden. Niihin kuuluvat muun muassa roadmap (tuotteen etenemissuunnitelma), kehityslista ja tuotekuvaus. Projektidokumentteja ovat esimerkiksi projektisuunnitelma, sopimukset, määrittelyt ja käyttöohjeet. Jokaiselle dokumenttityypille on olemassa määrätyt vastuut eri kohderooleille. Näistä on voitu luoda erilliset listat, jotka soveltuvat myös tarkistuslistoiksi tuotteiden ja projektien dokumentoinnin tilan tarkistamisessa. Opinnäytetyön liitteenä on molempien ryhmien dokumenttien kuvaukset ja toimenpidelistat. (Taulukko 2).

Käyttöohjeita voi olla eri versioita eri käyttäjäryhmille. Esimerkiksi pääkäyttäjän roolissa oleva käyttäjä oletettavasti tarvitsee koko tuotteen kattavat ohjeet, mutta tietyn moduulin tiedonsyöttäjän roolissa toimiva ei välttämättä tarvitse kuin moduulia koskevat ohjeistukset. Perinteisten sähköisten ohjetiedostojen ja paperiversioiden ylläpito on koitunut aikaavieväksi ja niiden sijaan pyritään jatkossa keskittämään kaikki yhteen helposti ylläpidettävään paikkaan. Agenteqillä on oma ohjesivustonsa Tampuurille (ohjeet.tampuuri.fi), jonne on tarkoitus lisätä kaikki Tampuuria ja sen moduuleita koskevat ohjeet. Sivusto on toteutettu Agenteqin omalla sisällönhallintaohjelmistolla Latomolla ja sivuston käyttäminen onnistuu selaimella. Jokaiselle Tampuuri-asiakkaalle annetaan sivustolle tunnukset, joilla voivat kirjautua sisälle.

Dokumentaatioprosessin kehitystä varten täytyy ottaa esiin uusi laatuvastaavan rooli (QA). Sen tehtävänä on luotujen dokumenttien avulla käydä läpi tuotteiden ja projektien laatua sekä tiedottaa siitä eri osapuolille. Kyseessä ei välttämättä ole vain yksi tietty henkilö, vaan se voidaan nimetä tarpeen mukaan, jopa yksittäisille projekteille. Lisäksi laatuvastaavan roolissa voi toimia useampi

henkilö, joista osa voi esimerkiksi keskittyä koodin laatuun ja osa ohjeiden laatuun. Laatuvastaavan rooliin kehittäjätiimistä erillisen henkilön nimeämisen vahvuutena on, että hänellä on projektiin ulkopuolisen näkökulma, joka jo sinällään pakottaa kehittäjien tekemään dokumentaation jo alusta alkaen huolellisemmin. Samalla tiedonvälitys selkiytyy myös projektipäälliköille ja yrityksen johdolle, kun laadusta kommunikointi tapahtuu heidän välillään.

4.1 Ohjelmistotuotteen dokumentoinnin vaiheet

Ohjelmistotuotteen dokumentoinnin vaiheet vaihtelevat tarpeen mukaan. Asiakkaille suunnatut määrittelyt ja käyttöohjeet ovat tärkeimmällä sijalla ja niitä pyritään pitämään aina ajan tasalla.

Asiakkaan tilatessa sovelluksen tai uuden, suuremman ominaisuuden olemassa olevaan ohjelmistoon, luodaan tästä asiakkaan kanssa toiminnallinen määrittely. Määrittelyssä asiakas kertoo, mitä haluaa ohjelmiston tekevän, ja sen pohjalta voidaan tehdä kehityssuunnitelma ja arvio työn kestosta. Projektipäällikkö laajuuden mukaan voi jakaa kehityksen pienempiin palasiin ja eri kehittäjille.

Toiminnalliset määrittelyt syntyvät ennen ohjelmiston kehityksen aloittamista ja sen pohjalta aloitetaan kehitys. Määrittely muuttuu lähes aina kehityksen aikana, kun toimintoja kehitetään ja asiakas haluaa muutoksia. Tämä tapahtuu kehittäjän, projektipäällikön ja asiakkaan vuorovaikutuksessa. Projektin loppuvaiheilla toiminnallisesta määrittelystä muotoutuu toiminnallinen kuvaus, joka kuvaa projektin ja sen ominaisuuksien toimintaa yleisemmällä tasolla.

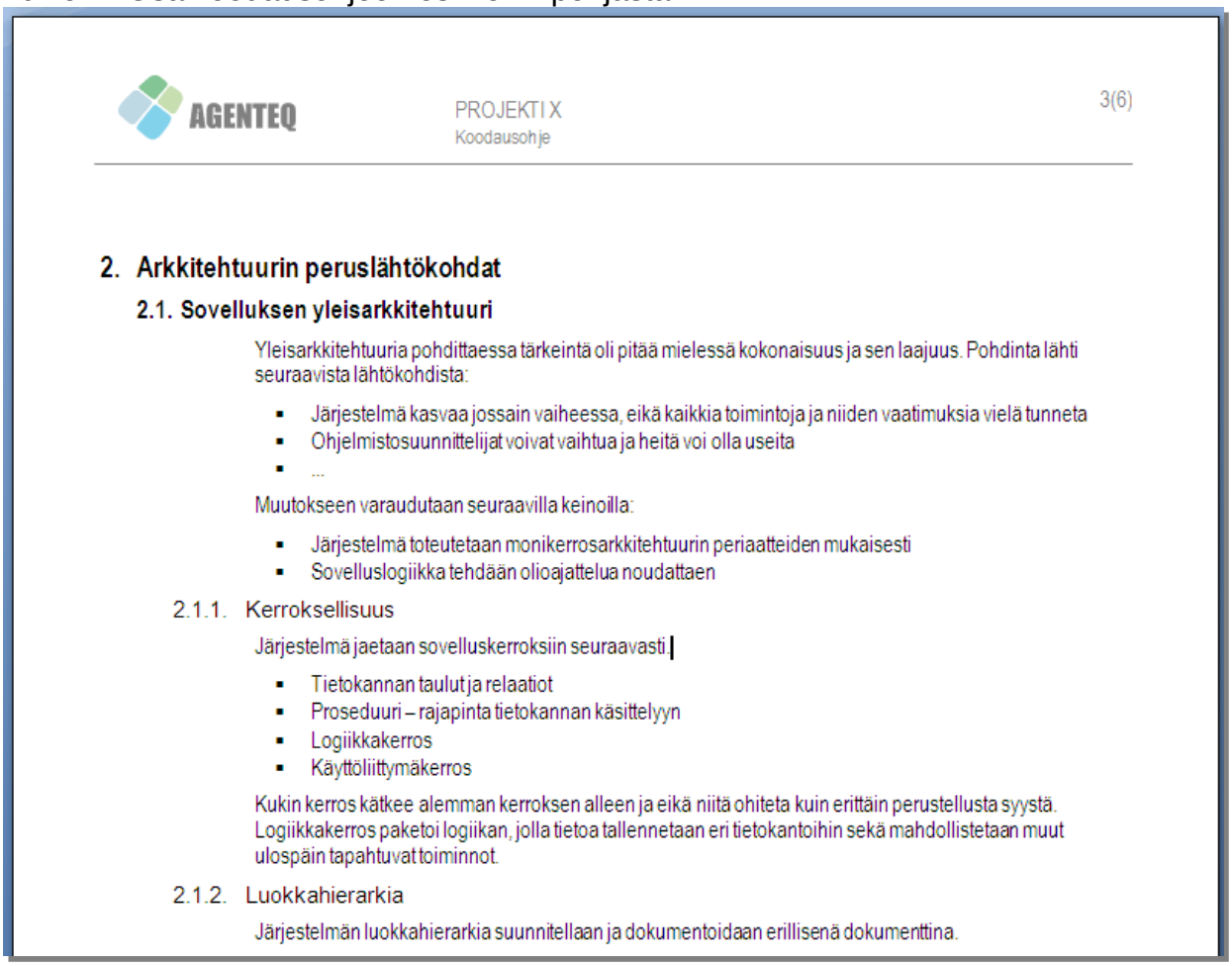
Tuotteen ja projektin käyttöohjeet luodaan kehityksen loppuvaiheessa, kun testauksen jälkeen kehittäjien ja asiakkaan projektivastaavien lisäksi tulee muita käyttäjiä. Ohjeet pyritään tekemään mahdollisimman selkeiksi ja käyttää kuvia tarpeen vaatiessa. Perinteisistä paperiohjeista ja sähköisistä dokumenteista poiketen web-sivustolla voidaan myös linkittää toisia sivuja yhteen, joka

helpottaa ohjeiden selaamista. Kuten Tampuuri, ohjesivusto on myös avattavissa kaikkialta, kunhan on tietokone, Internet-yhteys ja selain.

Kun tuotetta tai projektia on tarkoitus testata, luodaan sitä varten erikseen testaussuunnitelma. Suunnitelma listaa tuotteen ominaisuudet sekä mitä niiden toimintojen käytöstä olisi tarkoitus tapahtua. Toimintojen vieressä on yksinkertaisesti ”OK/EI”-sarake, johon voidaan merkitä, oliko testauksen tulos hyväksyttävä vai ei. Lisäksi testattavalle toiminnoille voidaan muita tietoja, kuten testauksen huomiot, onko korjattu, sekä testausta koskevat lisätiedot. Testaussuunnitelman voi täyttää suoraan Excelillä, jolloin sen voi lähettää esimerkiksi sähköpostilla. Testauksen suorittaa yleensä aina asiakas. Agenteqin sisältä laatuvastaava, kehittäjä tai projektipäällikkö käy läpi testauksen tulokset ja raportoi eteenpäin kehittäjille. (Liite 3)

Kehittäjiä varten projektille luodaan koodausohje. Sen tarkoituksena on kertoa uudelle kehittäjälle ohjelmiston arkkitehtuurista sekä sovelluskehittäjän toimintaohjeet. Arkkitehtuurista kerrotaan yleisellä tasolla, miten siihen päädyttiin ja kuinka se toteutetaan, sovelluksen kerroksellisuudesta sekä muuta niihin liittyvää. Sovelluskehittäjän toimintaohjeet kertovat kehittäjälle, miten projektin eri osa-alueet tulisi huomioida, käytettävät ohjelmointikielet, koodausohjeet ja nimeämiskäytännöt. Siinä kerrotaan myös kuinka kehitysympäristö määritellään ja pystytetään omalle koneelle, miten versio päivitetään asiakkaalle ja versionhallinnan käytännöistä. Näillä tiedoilla projektiin tullut uusi henkilö pääsee helposti sisälle projektin käytäntöihin ja saa kehitysympäristön nopeasti pystytetyksi.

Kuvio X. Osa koodausohjeen esimerkkipohjasta



AGENTEQ PROJEKTI X Koodausohje 3(6)

2. Arkkitehtuurin peruslähtökohdat

2.1. Sovelluksen yleisarkkitehtuuri

Yleisarkkitehtuuria pohdittaessa tärkeintä oli pitää mielessä kokonaisuus ja sen laajuus. Pohdinta lähti seuraavista lähtökohdista:

- Järjestelmä kasvaa jossain vaiheessa, eikä kaikkia toimintoja ja niiden vaatimuksia vielä tunneta
- Ohjelmistosuunnittelijat voivat vaihtua ja heitä voi olla useita
- ...

Muutokseen varaudutaan seuraavilla keinoilla:

- Järjestelmä toteutetaan monikerrosarkkitehtuurin periaatteiden mukaisesti
- Sovelluslogiikka tehdään olioajattelua noudattaen

2.1.1. Kerroksellisuus

Järjestelmä jaetaan sovelluskerroksiin seuraavasti.

- Tietokannan taulut ja relaatiot
- Proseduuri – rajapinta tietokannan käsittelyyn
- Logiikkakerros
- Käyttöliittymäkerros

Kukin kerros kätkee alemman kerroksen alleen ja eikä niitä ohiteta kuin erittäin perustellusta syystä. Logiikkakerros paketoi logiikan, jolla tietoa tallennetaan eri tietokantoihin sekä mahdollistetaan muut ulospäin tapahtuvat toiminnot.

2.1.2. Luokkahierarkia

Järjestelmän luokkahierarkia suunnitellaan ja dokumentoidaan erillisenä dokumenttina.

Kun uusi projekti luodaan ja sille aletaan kirjoittamaan koodausohjetta esimerkin pohjalta, tulee kehittäjälle tarve miettiä etukäteen sovelluksen arkkitehtuuria ja muita kehityksen osa-alueita ennen varsinaista kehityksen aloittamista. Arkkitehtuurin ja koodauskäytäntöjen muuttamisen vaikeus kasvaa kehityksen aloittamisen jälkeen ajan mittaan, joten on erittäin tärkeää aloittaa hyvältä pohjalta ja selkein tavoittein.

4.2 Koodidokumentointi

Yksi tärkeimmistä ohjelmistotuotteen dokumentoinnin vaiheista on koodidokumentointi. Koodidokumentaatiolla tarkoitetaan ohjelmistokoodin joukkoon lisättyjä kommentteja, jotka selventävät ohjelmiston toimintaa ja antavat käyttöesimerkkejä toisille kehittäjille. Tietyllä tavalla muotoillut

kommentit sallivat myös ohjelmien käsitellä niitä. Esimerkiksi Microsoftin Visual Studio -kehitysympäristö luo niistä ohjelmointia helpottavia Intellisense-avustustuksia, joka näyttää koodia kirjoittaessa oleellisia tietoja käytetyistä luokista ja funktioista. Ilman mitään dokumentaatiota ohjelmoijan täytyy päätellä luokkien ja funktioiden nimistä niiden toiminnat, joka laajemmissa projekteissa on lähes mahdotonta. Käytännössä tämä hyödyttää koko ohjelmiston kehitystä. Toiset kehittäjät pääsevät nopeammin perille koodin toiminnasta, laadunvalvonta helpottuu ja tuotteen elinkaari voi pidentyä.

Suurella osalla ohjelmointikielistä on olemassa jokin standardi tapa koodin kommentointiin. C#-kielen syntaksi pohjautuu suurelta osin Javaan, joten siinä on käytössä samantyylinen kommentointitapa. Yksittäinen kommenttirivi alkaa kahdella vinoviivalla (//) ja kommenttialue vinoviivalla ja tähdellä (/*). XML-kommentteja varten on kolmen vinoviivan käyttäminen (///). XML-kommentteja käytetään dokumentaation luonnissa ja siinä on olemassa tiettyjä merkintöjä, jotka voivat vaikuttaa luodun dokumentin ulkonäköön ja toimintaan. Esimerkiksi example-tagien sisälle voidaan antaa esimerkki koodin käytöstä tai param-tagilla voidaan kertoa vastaanottavan parametrin tyypistä. Visual Basic-kielessä kommentointi onnistuu samalla tavalla, mutta vinoviivan sijaan käytetään heittomerkkejä (’). (Microsoft)

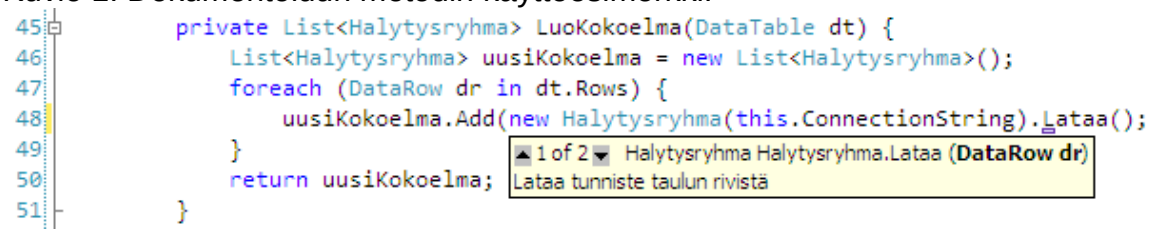
Kuvio 1. Dokumentoitua luokan koodia

```
namespace Agenteq.Tampuuri.Rakennusautomaatio {

    /// <summary>
    /// RAU-kohteen hälytysryhmä
    /// </summary>
    [Serializable]
    public class Halytysryhma : CommonConnection {
        private int? _halytysryhmaID;
        private int _kohteetID;
        private int _rooliID;
        private int _yritysID;
        private string _yritysNimi;
        private string _rooliNimi;
        private string _rooliYhteysNimet;
        private string _rooliYhteysOsapuoletID;
        private string _nimi;
        private IEnumerable<Aikataulu> _aikataulut;
        private IEnumerable<Vastaanottaja> _vastaanottajat;

        /// <summary>
        /// Alusta hälytysryhmä.
        /// </summary>
        public Halytysryhma() { ;}

        /// <summary>
        /// Alusta hälytysryhmä yhteydellä.
        /// </summary>
        public Halytysryhma(string connectionString) : base(connectionString) { ;}

        /// <summary>
        /// Alusta hälytysryhmä käyttäjän yhteydellä.
        /// </summary>
        public Halytysryhma(User user) : base(user.connectionstring) { ;}

        /// <summary>
        /// Lataa hälytysryhmä ID:n perusteella.
        /// </summary>
        /// <param name="halytysryhmaID">Hälytysryhmän ID</param>
        /// <returns>Hälytysryhmä</returns>
        public Halytysryhma Lataa(int halytysryhmaID) {
            Halytysryhma uusiKohde = new Halytysryhma(this.ConnectionString);
            uusiKohde.LataaKannasta(halytysryhmaID);
            return uusiKohde;
        }

```

Microsoftin Visual Studio -ohjelmassa kolmen vinoviivan lisääminen koodiin esimerkiksi metodin yläpuolelle luo sille automaattisesti XML-dokumentointipohjan, jonka voi täydentää vapaasti tekstillä. Olennaisinta on, että selittää metodin tehtävän ja monimutkaisimmissa tapauksissa täydentää muita tarpeelliseksi nähtäviä tietoja.

Hyvänä kommentointikäytäntönä voidaan pitää sitä, että koodia tulee jossain vaiheessa käyttämään joku toinen, jonka täytyy ymmärtää koodin toiminta mahdollisimman hyvin mahdollisimman nopeasti. Lisäksi ohjelmistokomponenttien koodauksessa kannattaa dokumentoida käyttötapauksia ja antaa esimerkkejä toimintojen käytöstä. Tämä säästää muiden ohjelmistokehittäjien aikaa ja samalla koko yrityksen resursseja.

Kuvio 1. Dokumentoidun metodin käyttöesimerkki.



Koodidokumentaatiolla ei sinällää vielä tarkoiteta sen saatavaksi asettamista, vaan siihen on olemassa monia tapoja. Esimerkiksi Visual Studio luo dokumentaatiosta erillisen xml-dokumentin, joka pitää olla komponenttikirjaston mukana, jos sitä halutaan hyödyntää. Koodidokumentaatioista voidaan luoda myös erillisiä dokumentteja, kuten pdf-tiedostoja tai html-sivustoja. Tätä varten on olemassa eri tapoja ja ohjelmistoja, johon valitaan sopivat ratkaisut tilanteen mukaan. Erillisten dokumenttien hyvänä puolena voidaan pitää, että ne voidaan laittaa saataville kehittäjien lisäksi myös muille henkilöille.

5 DOKUMENTOINNIN HALLINNOINTI AGENTEQISSÄ

Agenteq siirtyi keväällä 2009 käyttämään pääasiallisena dokumenttipankkina Windows Server 2003 -käyttöjärjestelmän päällä pyörivää Sharepoint-järjestelmää. Yritys ei nähnyt tarpeellisena dokumenttien versiohistorian ylläpitämiseen muutoin kuin organisaation sisäisissä dokumenteissa sekä joissain määrittelyissä. Tämän takia Sharepoint on tarpeen täyttävä keskitetty dokumenttipankki, jossa toteutuu tiedostojen tallennus ja poisto, käyttöoikeuksien hallinta sekä virtuaaliset hakemistot.

Kuvio 3: Sharepointin dokumenttien hakemistolistaus

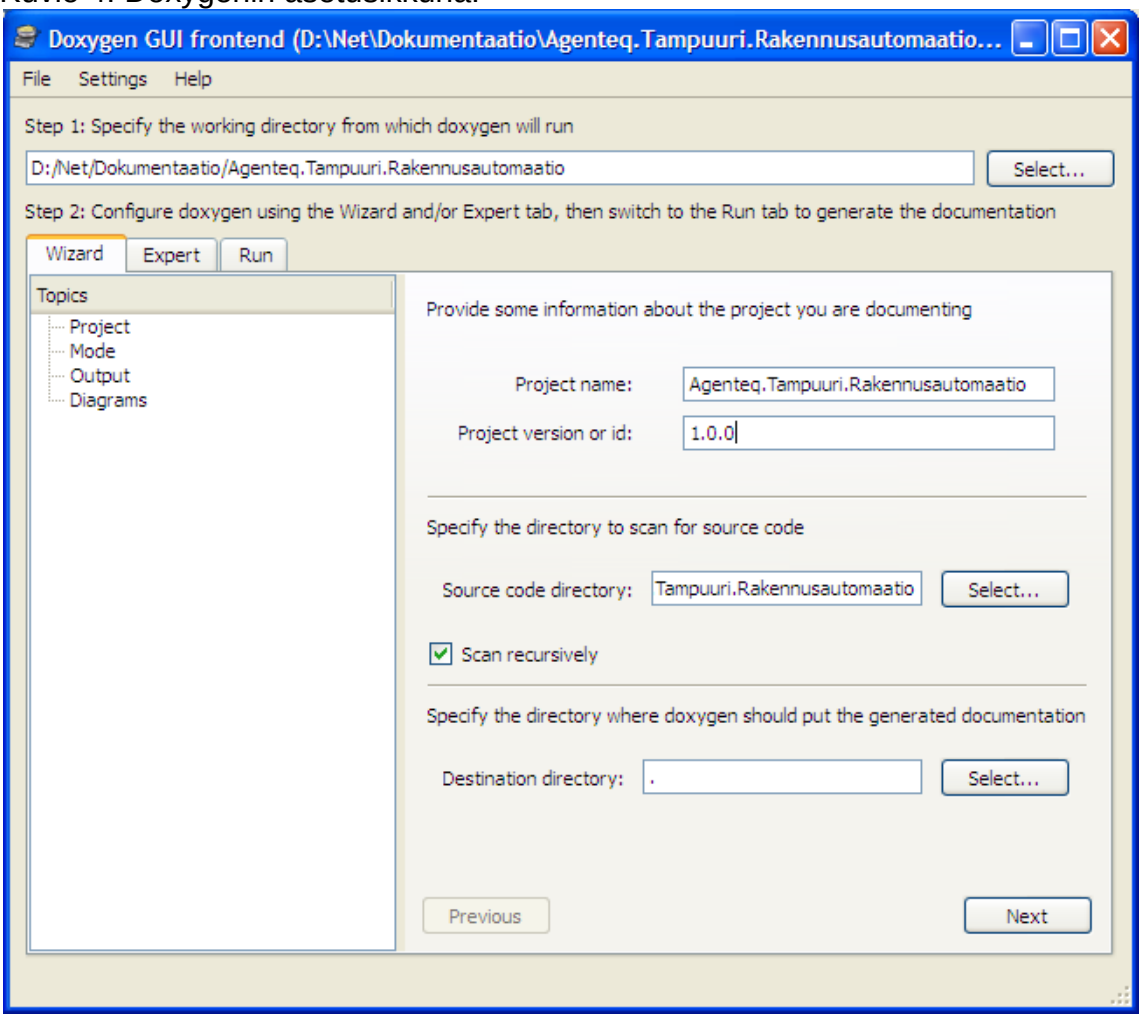
Sharepointia käytetään pääasiassa selainkäyttöliittymän kautta. Käyttäjän tunnistuksessa käytetään Windowsin Active Directorya, jonka avulla voidaan rajata pääsy suoraan pelkästään yrityksen henkilöstölle ja antaa käyttäjille halutut oikeudet eri toimintoihin.

Ohjelmistojen kehityksessä on käytössä Subversion-versionhallintajärjestelmä, joka mahdollistaa keskitetyn ylläpidon ohjelmistokoodille, versioinnille sekä varmuuskopioinnille. Versionhallinnassa voidaan pitää projektien kanssa myös kehittäjille tarkoitettuja dokumentteja, mutta muuten yleisiä dokumentteja sinne ei laiteta. Tällä tavalla kehittäjät voivat päivittää esimerkiksi projektin koodausohjetta tarpeen mukaan ja se välittyy automaattisesti muille kehittäjille.

Koodidokumentointi voidaan myös jossain määrin automatisoida. Kun ohjelmiston koodin kommentit ovat tietyssä muodossa, pystytään niistä tietyillä ohjelmilla luomaan erikseen koodidokumentaatiokokonaisuuksia eri muodoissa, esimerkiksi web-sivustona. Koodidokumentaation säilytys voidaan siten jatkossa keskittää esimerkiksi Intranetiin web-sivustoksi, johon kaikilla ohjelmistokehittäjillä olisi pääsy suoraan selaimella. Tämä voi helpottaa kehitystyön lisäksi mm. ohjelmistovirheiden korjausta ja laadunseurantaa, kun koodi on helposti kaikille saatavilla. Sivuston hyvänä puolena on myös se, että tiettyjä kohtia koodista voidaan esimerkiksi linkittää suoraan muihin järjestelmiin, kuten Sharepointin tiketteihin ja kehityslistoihin.

Agenteqin pääasiallisina ohjelmointikieliä ovat Visual Basic .NET ja C#, joten näiden tuki on tarpeellinen valittaessa käytettävää koodidokumentaationluontiohjelmaa. Tällaisia ohjelmia on muun muassa vapaan lähdekoodin Doxygen sekä Robodoc. Myös Microsoft on julkaissut oman ohjelmansa koodidokumentaatioon nimeltään Sandcastle.

Kuvio 4: Doxygenin asetusikkuna.



Doxygen on GPL-lisenssillä julkaistu avoimen lähdekoodin ohjelma, jolla on hyvä tuki C#:lle ja monelle muulle ohjelmointikielelle. Se on yksi suosituimmista – ellei jopa suosituin – koodidokumentaatio-ohjelmista, joten sen tukeminen tulee luultavasti jatkumaan pitkään. Asennus ja käyttöönotto tuntui helpolta, samoin sen luomat dokumentaatiot asiallisilta. (Kuvio 4, http://www.doxygen.org)

Visual Basic .NET -koodin dokumentaatiota varten on olemassa huomattavasti vähemmän ratkaisuja osaltaan sen erikoisen syntaksin, toisaalta taas Microsoftin tekniikoihin lukittautumisessa. Tässä tapauksessa Doxygenissä voidaan käyttää eräänlaista suodatusmahdollisuutta, jossa ohjelmakoodi kierrätetään toisen ohjelman kautta muuttaen Visual Basic -syntaksin

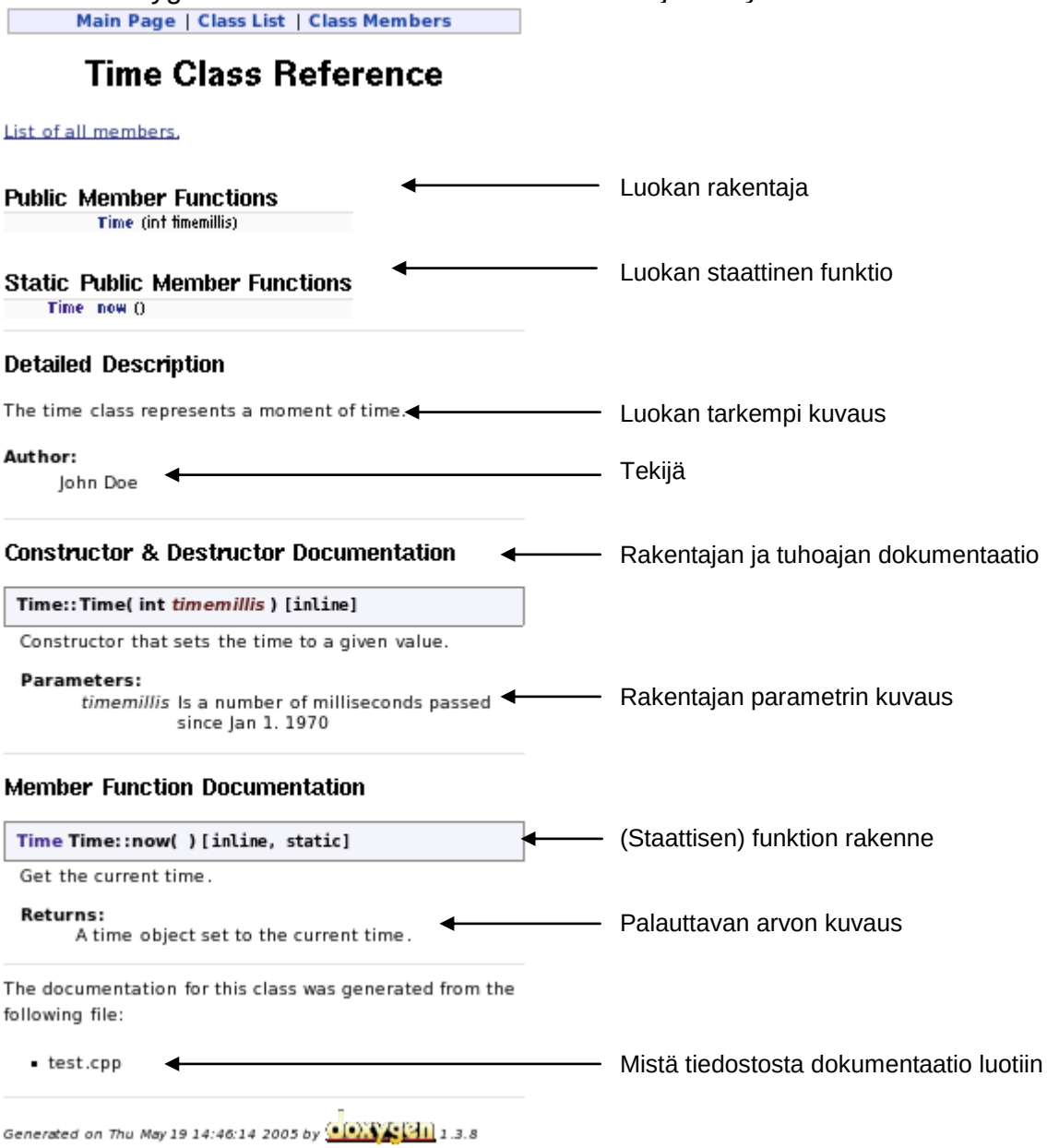
Doxygenin ymmärtämään muotoon. On olemassa muun muassa Unix-ohjelmia, kuten *gawk*, joiden avulla tämä on mahdollista. Tällä hetkellä saatavilla on perussyntaksin kääntävä suodatustiedosto (.awk), joka palauttaa Visual Basic -koodista C#-kielen vastaavaa koodia, josta Doxygen voi luoda dokumentaation. (http://trac.sevo.org/projects/doxyvb)

Robodoc on myös julkaistu GPL-lisenssillä, mutta se on täysin komentorivipohjainen ja sen päivitystahti vaikuttaa olevan hitaampi kuin Doxygenillä, joten uusien tarvittavien ominaisuuksien saaminen ei ole niin varmaa. Lisäksi käyttöönotto ei ollut niin helppoa verrattuna Doxygeniin. (http://www.xs4all.nl/~rfsber/Robo/robodoc.html)

Microsoftin Sandcastle taasen on julkaistu lisenssillä Microsoft Public License (Ms-PL) ja se on kokoelma eri komentoriviohjelmia. Sen käyttöönotto on hankalampaa kuin edellä mainittujen ohjelmien. Lisäksi dokumentaatio itse ohjelmasta on lähes olematonta ja vaikea ymmärtää. (http://sandcastle.codeplex.com)

Valitsin Doxygenin koodidokumentaation luontiin, koska sen käyttöönotto vaikutti näistä vaihtoehdoista helpoimmalta ja sen elinkaari varmimmalta. Lisäksi sen ollessa saataville jokaiselle käyttöjärjestelmälle on lähes kaikki siitä saadut kokemukset sovellettavissa käyttöjärjestelmästä riippumatta. Koska kohdeyleisö on organisaation sisäinen henkilöstö ja koodidokumentaatio on suurelta osin staattista web-sivustoa, muidenkin ohjelmien käyttöönoton tutkiminen tulevaisuudessa ei ole täysin poissuljettua. Automaation ja siihen liittyvien ohjelmien käyttöä tullaan hiomaan jatkossakin, jotta lopputulos olisi mahdollisimman hyödyllinen.

Kuvio 5: Doxygenin luoma esimerkkidokumentaatio ja selitykset.



Itse automatisointi voidaan toteuttaa esimerkiksi ajastamalla koodidokumentaationluontiohjelmat versionhallinnan yhteyteen, josta luotu dokumentaatio voidaan laittaa saatavaksi web-sivustona. Tällä tavalla ohjelmistokehittäjille ei tule ylimääräisiä töitä, vaan dokumentaatio perustuu kehittäjän omaan, normaalin työn ohella luotuun tuotokseen. Tähän ratkaisuun päädyin lopulta käyttämällä yrityksen omaa kehityspalvelinta, jonne haetaan versionhallinnasta koodit ja luodaan dokumentoinneille sivustot automaattisesti kerran viikossa.

Näin alkuvaiheessa koodidokumentaatiosivuston projektislistaus on staattista html:ää ja uusi projekti täytyy lisätä käsin, mutta tulevaisuudessa tämäkin muuttuu automaattiseksi. Sivusto listaa eri tuotteet omiksi kokonaisuuksiksi, joiden alle listataan niihin kuuluvien projektien omat koodisivustot. Sivustoille pääsee käsiksi yrityksen sisäverkosta normaalilla selaimella. (Kuvio 6)

Kuvio 6: Koodidokumentaatiosivuston projektilistaus

Koodidokumentaatiosivuston tarkoituksena on auttaa kehittäjiä ja helpottaa laadunvalvontaa, jonka takia projektien lisätietojen – kuten tekijä, viimeinen muokkaus, jne – tuominen sivustolle on tulevaisuudessa hyvin mahdollista.

6 TESTAUS

Testatakseni koko dokumentointiprosessin toimivuutta valitsin yhden oman projektin dokumentoitavaksi. Projektin nimi on Rakennusautomaatio ja on yksi Tampuurin moduuli. Ajan puutteen vuoksi kaikkia projektin dokumentteja ei tuotettu, mutta tärkeimmät ja varsinkin dokumentointikäytännön prosessin kehityksen kannalta oleellisimmat dokumentit otettiin mukaan. Lisäksi osa dokumenteista sisältää arkaluontoista asiaa, eikä niitä voi kokonaisuudessaan sisältää opinnäytetyöhön.

Projektin testaussuunnitelma luotiin sovelluksen ensimmäisen version valmistuttua ja se lähetettiin asiakkaille täytettäväksi. Testaussuunnitelma jaoteltiin sovelluksen osien ja toiminnallisuuksien mukaan listan selkeyttämiseksi ja kokonaisuuksien hahmottamiseksi. Asiakkaan antamien tulosten mukaan ohjelmasta korjattiin virheitä tai muutettiin toiminnallisuutta. (Kuvio 7)

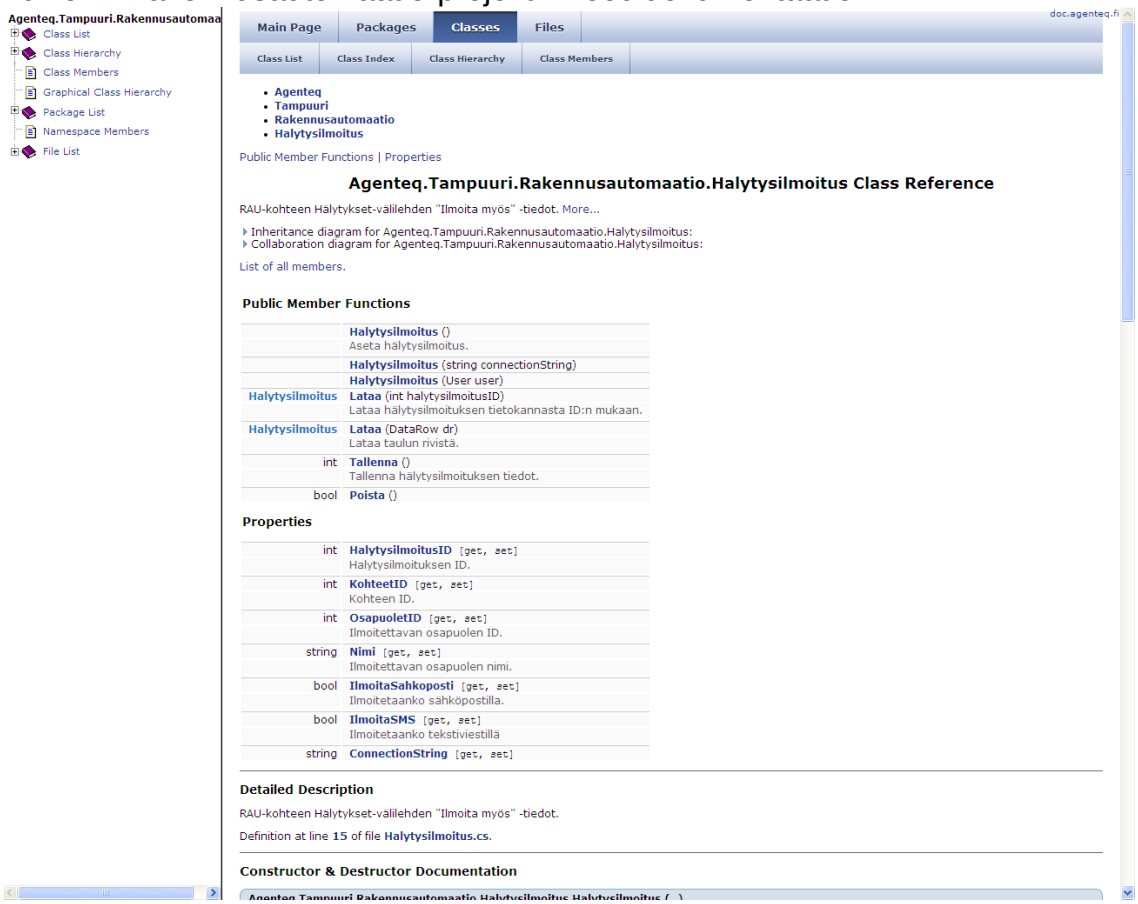
Kuvio 7. Rakennusautomaatio-projektin täytetty testaussuunnitelma

	A	B	C	D	E	F	G	H
1	**Tampuuri / Rakennusautomaatio 1.0**							
2	***Testaussuunnitelma***							
3			**Ominaisuus**	**Toiminto**	**Tulos (OK/EI)**	**Testauksen huomiot**	**Korjattu / huomioitavaa**	**Testausta koskevat lisätiedot**
4	***Navigaation linkit / Sivu***							
5		**Asetukset**						
6								
7			Tallentaminen					
8				Muuta arvoja ja paina Tallenna-nappia	ok			
9			Lataaminen					
10				Sivu lataa tallennetut arvot oikein	ok			
11								
12		**Kohderekisteri / Puu**						
13								
14			Puun toiminta					
15				Puu listaa vain RAU:n kohteet, sekä tarvittaessa polun kohteet RAU-kohteeseen asti	ok			Polun listausta testattaessa lu kuin kustannuspaikkatasolle
16				RAU-kohdetta klikatessa avautuu RAU-lomake tietojen muuttamiseen	ok			
17								
18		**Kohderekisteri / Kohdelistaus**						
19								
20			Kohdelistaus					
21				Listaa RAU-kohteet vain kustannuspaikkatasolta	ok			
22				Näyttää rivillä kohteen nimen, kunnan, IP-osoitteen ja merkin	ei	kunnan kohdalla on postinumero		
23				Kohteen verkonvalvonnan ja hälytyksen käytössä olot näkyvät rukseina	ok			
24				Nimeä klikattaessa avautuu kohteen RAU-lomake	ok			

Rakennusautomaation koodidokumentaation luonti osoittautui hankalammaksi. Projektissa osa on Visual Basic.NET -koodia, jonka dokumentoimisesta Doxygen ei suoriudu niin helposti. VB.NET-koodin suodattavaa lisäosaa

käyttäen siitä saa jokseenkin hyödynnettävää dokumentointia. Tärkeimpien komponenttien ollessa C#-kieltä koodidokumentaatiosta tuli laadunseurannassa hyödynnettävä osa ja muut kehittäjät pääsevät tutkimaan sovelluksen rakennetta ja koodia suoraan selaimella.

Kuvio 7. Rakennusautomaatio-projektin koodidokumentaatio



Muut projektidokumentit, kuten määrittelyt, koodausohje ja käyttöohje perustuivat jo aikaisempiin tuotoksiin ennen dokumentointiprosessia, mutta niiden tekeminen projektille antavat sille paremman pohjan jatkokehitykselle ja asiakkaiden tukemiseen. Testauksen avulla tuli ilmi koodidokumentaation ongelmat, mutta muuten se oli onnistunut. Testaussuunnitelman hyödyt tulivat heti esille, kun sovelluksen toiminnallisuudet voitiin jakaa tiettyihin testattaviin osiin ja asiakas pystyi näin löytämään virheitä suunnitelmallisesti. Muiden projektidokumenttien avulla taas voidaan luoda yhtenäiset pohjat muille projekteille.

7 YHTEENVETO

Aloitin Agenteissä ohjelmistosuunnittelijana kaksi vuotta ennen opinnäytetyön valmistumista syksyllä vuonna 2008. Varsinkin alussa dokumentoinnin puute oli häiritsevää ja hidasti uusien työntekijöiden pääsemistä sisälle projekteihin, minut mukaan lukien. Alusta alkaen Agenteqissä tiedostettiin dokumentoinnin puute, mutta asiaa ei viety kunnolla eteenpäin kuin vasta tämän opinnäytetyön tekemisen johdosta. Ollessani kehittäjänä monessa eri projektissa tiedostin itsekin tämän puutteen moneen kertaan. Käytännön puuttuessa dokumentointia oli tehty vain tarpeen vaatiessa ja usein ilman mitään laadittuja sääntöjä, vaikkakin organisaatiossa käytettiin joitakin vanhoja dokumenttipohjia. [Ei julkaistavaa tietoa]

Aiheen valinnan jälkeen piti miettiä, mistä aloittaa. Ohjelmistojen dokumentoinnista löytyy kirjallisuutta ja joitakin aihetta lähellä olevia opinnäytetöitä. Kyseessä oli kuitenkin tietylle yritykselle tehtävästä toimeksiannosta ja prosessien kehittämisestä, joten jätin kirjallisuuden lukemisen vähemmälle. Selvitin, mitä dokumentteja yrityksessä on nyt käytössä, mitä tuotteita ja projekteja on kehityksessä ja keitä ovat organisaation osapuolet ja ulkoiset roolit. Koska ohjelmointi on Agenteqissä suurin työllistäjä ja usein tulee vastaan tapauksia, joissa koodin dokumentointi olisi säästänyt monta työpäivää, päätin tehdä koodidokumentaatiosta yhden opinnäytetyön tärkeän osa-alueen. Idea laadunvalvonnasta ja sen liittämiseksi osaksi prosessia tuli sen jälkeen, kun koodidokumentaatiosivusto oli testattavana.

Opinnäytetyön tekemisen aikana on tullut esiin paljon hyviä tapoja parantaa käytäntöjä. RACI-toimenpidelistaukset voivat jo sellaisenaan antaa organisaatiolle hyvän keinon tarkistaa dokumenttien olemassaolon ja vastuuhenkilöt jokaiselle tuotteelle ja projektille. Tiettyjä vaikeuksia löytyi koodin dokumentoinnissa. Doxygen ei oletuksena tue Visual Basic .NET -koodin dokumentointia, jonka takia täytyy käyttää enemmän aikaa sen selvittämiseen. Myös koodidokumentoinnin automatisointia en saanut täysin automaattiseksi ajan puutteen vuoksi. Näistä huolimatta koodidokumentaatio on idealtaan ja

jatkon kannalta mielestäni onnistunut. Jatkokehityksen jälkeen siitä voi tulla yksi tärkeimmistä prosesseista yrityksessä. Organisaatiossa tullaan käyttämään dokumenttipohjia ensin pienemmässä määrin ja tarpeiden mukaan päivittäen, jonka jälkeen henkilöstöä koulutetaan käyttämään niitä mahdollisimman usein osana kehitysprosessia. RACI-toimenpidelistausten avulla voidaan helposti määrätä tuotteille ja projekteille vastuuhenkilöt dokumenttityyppien luomiseen ja ylläpitoon.

Yleisesti ottaen ohjelmistotuotteen dokumentoinnissa on monissa yrityksissä puutteita. Tässä opinnäytetyössä puutteet pyrittiin ja suurelta osin onnistuttiin korjaamaan kohdeyrityksessä. Tekemäni ratkaisut voivat sopia lähes sellaisenaan myös muihin ohjelmistoyrityksiin, ja toimenpidelistaa voidaan soveltaa muidenkin alojen yrityksissä. Osa prosesseista on jo käytössä ja käyttöä lisätään sen mukaan, kun nähdään tarpeelliseksi ja resurssit riittävät. Tässä opinnäytetyössä mainittujen ratkaisujen myötä dokumentoinnin tekeminen on saatu paremmalle pohjalle, mutta vasta tämän jälkeen alkaa itse koulutus ja käytäntöjen tekeminen rutiineiksi.

Työn viedessä suuren osan päivistäni oli opinnäytetyön kirjoitus hidasta. Osa viivästyksistä johtuivat itsestäni, kun taas jotkut kuukaudet olivat täynnä työkiireitä. Opiskeluajan lopun lähestyessä päätin puristaa opinnäytetyön loppuun. Opinnäytetyön ohjaajani Minna-Kristiina Paakki auttoi paljon työn etenemisessä, aiheessa pysymisessä ja kirjoituksen rakenteen muodostamisessa oikeanlaiseksi.

Yhteenvetona voidaan sanoa dokumentaatioprosessin kehityksen olleen onnistunut. Aihe sopi hyvin omaan tradenomitutkintoon ja yrityksessä ohjelmistosuunnittelijana työskentelevänä sitä oli jatkuvasti mietittävä muutenkin. [Ei julkaistavaa tietoa]

8 LÄHTEET

Agenteq Oy. Viitattu 11.10.2010. http://www.agenteq.fi

Briand, Lionel C. 2003. Software Documentation: How Much is Enough? Ottawa, Canada.

Doxygen. http://www.doxygen.org

Doxygen Visual Basic filter. Viitattu 18.10.2010. http://trac.sevo.org/projects/doxyvb

Microsoft Sharepoint. http://sharepoint.microsoft.com

Robodoc. http://www.xs4all.nl/~rfsber/Robo/robodoc.html

Sandcastle. Viitattu 23.8.2010. http://sandcastle.codeplex.com

Selic, Bran. 2009. Agile Documentation, Anyone? University of British Columbia.

Wikipedia. Sandcastle (software). Viitattu 23.8.2010.
http://en.wikipedia.org/wiki/Sandcastle_(software)

Wikipedia. C Sharp (programming language). Viitattu 18.10.2010.
http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Wikipedia. Responsibility assignment matrix. Viitattu 23.11.2010.
http://en.wikipedia.org/wiki/Responsibility_assignment_matrix.

Wikipedia. Comparison of documentation generators. Viitattu 16.8.2010.
http://en.wikipedia.org/wiki/Comparison_of_documentation_generators

XML Documentation Comments (C# Programming Guide). Viitattu 18.10.2010
http://msdn.microsoft.com/en-us/library/b2s063f7.aspx

Tuotedokumenttien kuvaukset

Dokumentti	Kuvaus
Roadmap	Tuotteen yleiseen kehitykseen liittyvät suunnitelmat.
Toiminnallinen kuvaus	Tuotteen toiminnallisuuden kuvaus.
Moduulien rajapintakuvaus	Moduulien tarjoamat rajapinnat, joiden avulla voidaan yhdistää muita moduuleita/palveluita toisiinsa.
Kehityslista	Tuotteen ja sen yksittäisten ominaisuuksien kehityslista.
Tuotekuvaus	Tuotteen ja sen ominaisuuksien kuvaus yleisellä tasolla, mm. markkinointia varten.
Testaussuunnitelma	Tuotteen testausta varten luotu suunnitelma.
Käyttöohje	Tuotteen käyttöön liittyvät ohjeet.
Koodausohje	Kehittäjille tarkoitettu ohje ohjelmien koodaukseen, jonka avulla voidaan pitää yllä tiettyjä käytäntöjä.
Asennusohje	Ylläpitäjälle luotu ohje tuotteen asennukseen ja säätämiseen.
Koodidokumentaatio	Ohjelmiston koodista luotu dokumentaatio.

Tuotedokumenttien RACI-roolit

Dokumentti	Johto	Tuote-päällikkö	Kehittäjä	QA	Tuki	Asiakas	Markki-nointi	Ylläpito
Roadmap	I	A / R	C			I	I	I
Toiminnallinen kuvaus		I	R	A	I	I		
Moduulien rajapintakuvaus		R	R	A				I
Kehityslista		A / R	R				I	
Tuotekuvaus	I	R	C		I	I	A / R	
Testaussuunnitelma		A / R	C	R				
Käyttöohje		A	R	C	R	I		
Koodausohje			R	A				
Asennusohje			R	A				C
Koodidokumentaatio			R	A				

Projektidokumenttien kuvaukset

Dokumentti	Kuvaus
Projektisuunnitelma	Asiakkaan kanssa luotu suunnitelma projektista, jonka arvioihin sopimusten ja muiden dokumenttien sisällöt voivat pohjautua.
Tuotantosopimus	Asiakkaan kanssa luotu sopimus projektin tuotannosta.
Ylläpitosopimus	Asiakkaan kanssa luotu sopimus projektin ylläpidosta.
Toiminnallinen määrittely / Toiminnallinen kuvaus	Asiakkaan kanssa projektin alussa luotu dokumentti. Projektin loppuvaiheilla toiminnallisesta määrittelystä muotoutuu toiminnallinen kuvaus, joka määrittelyjen sijaan kuvaa projektin ja sen ominaisuuksien toimintaa yleisemmällä tasolla.
Testaussuunnitelma	Projektin ja sen ominaisuuksien testausta varten luotu suunnitelma.
Käyttöohje?	Tapauskohtaisesti projektista voidaan luoda myös käyttöohje.
Koodausohje	Kehittäjille tarkoitettu ohje ohjelmien koodaukseen, jonka avulla voidaan pitää yllä tiettyjä käytäntöjä.

Projektidokumenttien RACI-roolit

Dokumentti	Johto	Projekti-päällikkö	Kehittäjä	QA	Tuki	Asiakas	Markki-nointi	Ylläpito
Projektisuunnitelma	I	A / R	C	I		R		
Tuotantosopimus	A / R					R	I	
Ylläpitosopimus	A / R				I	R	I	
Toiminnallinen määrittely / Toiminnallinen kuvaus		A / R	R	C		C		
Testaussuunnitelma		A / R	C	R		I		
Käyttöohje?		A / R	R	C		I		
Koodausohje			R	A				

Järjestelmä X

Testaussuunnitelma

Ominaisuus	Toiminto	Tulos (OK/EI)	Testauksen huomiot	Korjattu / huomioitavaa	Testausta koskevat lisätiedot
Navigaation linkit / Sivu					
Kirjautumissivu					
Kirjautuminen	Hyväksyy vain olemassa olevan tunnus- ja salasanayhdistelmän				
Unohditko salasanasi? -linkki	Linkin alta avautuu kenttä, johon tunnukseen liitetyn sähköpostiosoitteen annettua lähetetään uusi salasana				Väärästä sähköpostiosoitteesta ei anneta virheilmoitusta
Etusivu					
Yläpalkki	Näyttää käyttäjän tiedot ja uloskirjautumisnapin				
Navigaatio	Näyttää käyttäjän oikeuksien mukaisen navigaation				
Alisivu 1					
...	...				