Lauri Limnell

Laatutekijöiden merkitys ohjelmistotyössä

Tietojenkäsittelyn koulutusohjelma

2012

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

LAATUTEKIJÖIDEN MERKITYS OHJELMISTOTYÖSSÄ

Limnell Lauri
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Kesäkuu 2012
Ohjaaja: Stenfors, Juha
Sivumäärä: 32
Liitteitä:

Asiasanat: laatu, käytettävyys, projekti, ominaisuudet, ohjelmisto

---

Työn keskeisin tarkoitus on mallintaa laatutekijöitä, niiden mittaamista ja soveltamista ohjelmistomaailmassa. Työ on kaksiosainen. Se alkaa laadun määrittelemisellä ja mallintaa käytettävien standardien perustoiminnallisuutta ja laadun elinkaarta edeten laadun vaatimusten kehykseen, ominaisuuksiin ja dokumentointiin. Työn toinen osa on opitun tiedon hyödyntämistä projektin omaisena tuotantokäytössä olevassa tietojärjestelmässä käytettävyyden näkökulmasta. Työn loppuosa tehtiin raporttina.

THE SIGNIFICATION OF QUALITY FACTORS IN SOFTWARE ENGINEERING

Limnell Lauri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Management
June 2012
Supervisor: Stenfors Juha
Number of pages: 32
Appendices:

Keywords: quality, usability, project, properties, software

The purpose of this thesis was demonstrating quality factors. Measuring and using them in software engineering. The thesis is two-parted. It begins by defining quality, its life cycle and demonstrating the basic functionality of the used standards advancing to the frame of quality requirements, quality properties and documentation. The second part of the thesis is utilizing the learned knowledge as a report from the perspective of usability on an information system that is already in production use.

SISÄLLYSLUETTELO

1 JOHDANTO

Ihmisillä on valmiita käsityksiä puhuttaessa laadusta. Päällimmäisinä ovat tuotteisiin ja palveluihin kohdistuvat ennakko-odotukset. Jos käyttäjä on tyytyväinen ja toteaa saaneensa vastineeksi sen, mitä hän on odottanutkin, laatuvaatimukset ovat täyttyneet. Ohjelmistomaailmassa pyritään aina mahdollisimman tasaiseen lopputulokseen, jotta järjestelmä tai tuote olisi objektiivisesti laadukas kaikilta ominaisuuksiltaan ja se olisi vaatimusten mukainen sekä täyttäisi sille asetetut spesifikaatiot. Laatutasolle tulee määritellä selkeät kriteerit, jotta kaikki ymmärtävät ne samalla tavalla. Vähimmäisvaatimuksena on tuotteen toiminnallisuus.

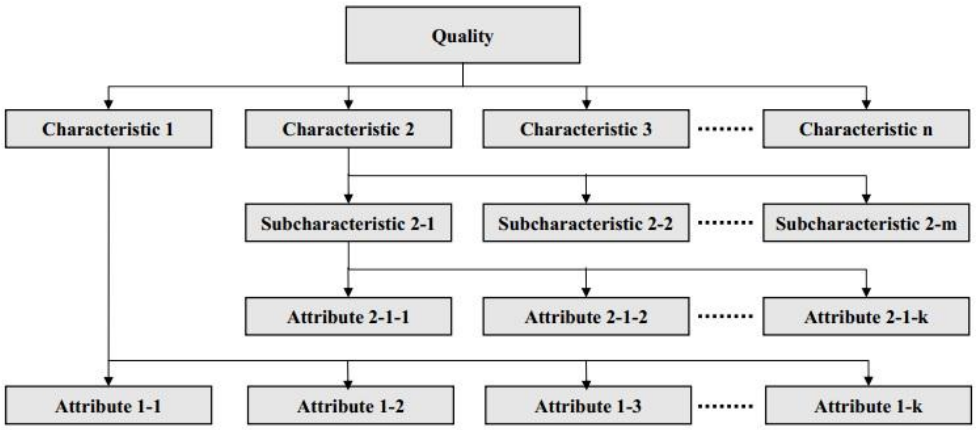
Kuinka laatua sitten mitataan? Laadun mittaaminen tapahtuu määrittelemällä tuotteen tai järjestelmän ominaispiirteet. Tämän tutkielman ensimmäinen osio mallintaa, kuinka jokainen ominaispiirre voidaan jakaa omiksi aliominaispiirteikseen aina attribuutteihin asti. Nämä attribuutit luonnehtivat ja rajaavat ominaispiirteitä. Tässä tutkielmassa demonstroidaan myös, kuinka laatua ja sen elinkaarta mitataan, millaisia vaatimuksia laatutekijöille asetetaan, mitkä ovat laadun ominaisuudet ja mikä merkitys dokumentoinnilla on ohjelmistotyöskentelyssä. Tutkielman teossa käytetty materiaali on kansainvälisesti tunnettu ISO/IEC 25000 – standardisarja, joka kuvastaa järjestelmien laatua, sen vaatimuksia ja määrittelyjä.

Tutkielman toisessa osiossa (projektiosuus) on käytetty top-down – menetelmää (tunnetaan myös nimellä vaiheittain suunnittelu), joka tarkoittaa järjestelmän purkamista pienempiin osiin, jotta saadaan selkeä yleiskuva järjestelmän sisältämistä osajärjestelmistä ja peruselementeistä. Projektiosuus on kiteytetty analysointi tietojärjestelmien käytettävyystutkimusta, jonka painopisteenä tässä tutkielmassa on käytettävyys, sen asettamat vaatimukset, attribuutit ja heijastuminen käyttäjän näkökulmasta. Projektissa käytetyn tietojärjestelmän jokaista laatuominaisuutta tullaan tulevaisuudessa mallintamaan samalla tavalla, kuin tämän tutkielman käytettävyysanalyysiakin. Tämä laaja prosessi on tälläkin hetkellä ”ongoing”, joka tarkoittaa, että se on jatkuvaa. Projektiosuus päättyy osaltamme kesällä 2012, joka tarkoittaa sitä, että muiden laatuominaisuuksien analysointi jää toisien tutkielmien tehtäväksi.

2 ISO/IEC 25000

2.1 Historia

Tietokoneita käytetään laajoissa ja yhä kasvavissa moninaisissa sovellusympäristöissä ja niiden oikea toimintatapa on usein kriittinen liiketoiminnan onnistumiselle ja/tai ihmisten turvallisuuden kannalta. Korkealaatuisten sovellusten kehittäminen tai valitseminen on siis hyvin tärkeää. Ohjelmistojen laajat vaatimukset ja niiden laatutekijät ovat avainasemassa, kun halutaan varmistaa asianmukainen laatu. Tämä voidaan saavuttaa määrittelemällä sopivat tunnusomaiset laatutekijät, samalla ottaen huomioon ohjelmistotuotteen tarkoitettu käyttö. On tärkeää, että jokainen olennainen ohjelmistotuotteen laatutekijä on tarkennettu ja arvioitu käyttäen validoituja tai laajasti hyväksyttyjä keinoja. (Azuma, Scalet 2004, 7.)



Kuva Laatu (Nevalainen 2009, 3.)

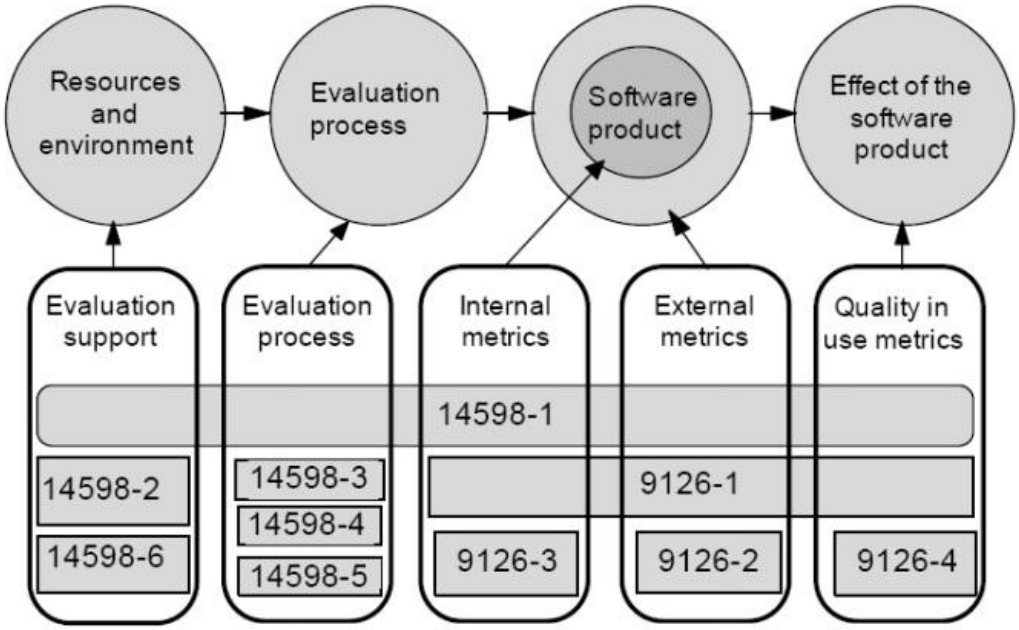
Kuva 1 määrittelee laadun ominaispiirteisiin, jotka on jaettu vieläkin pienempiin osiin. Laatu pystytään siis kuvailemaan tarkasti, jotta toiminnan tai ohjelmiston ominaisuudet varmistavat parhaimman mahdollisen tuloksen. Jokainen ominaispiirre sisältää attribuutteja, jotka luonnehtivat ja rajaavat ominaispiirteitä.

ISO (International Organization for Standardization) ja IEC (International Electrotechnical Commission) muodostavat erikoistuneen elimen maailmanlaajuiselle standardoinnille. Kansalliset elimet, jotka ovat osa ISO:a tai IEC:tä, osallistuvat kansainvälisten standardien kehittämiseen keskinäisten organisaatioiden muodostamissa tek-

nisissä komiteoissa, jotka erikoistuvat tiettyihin teknisen toiminnan aloihin. ISO- ja IEC-komiteat tekevät yhteistyötä yhteisten kiinnostuskohteiden pohjalta. Toiset kansainväliset organisaatiot, hallinnolliset ja ei-hallinnolliset, jotka ovat yhteistyössä ISO ja IEC standardien kanssa, ottavat myös osaa työhön. Informaatioteknologian alalla, ISO ja IEC ovat muodostaneet yhteisen teknillisen komitean, ISO/IEC JTC 1:n. (ISO/IEC, 5.)

2.2 SQuaRE

Laadulle tunnusomaiset piirteet ja niihin liittyvät mittausjärjestelmät voivat olla hyödyllisiä sekä ohjelmistotuotteen arvioinnissa että laatutekijöiden määrittelyssä. SQuaRen edeltäjä ISO/IEC 9126:1991 on korvattu kahdella siihen liittyvällä moniosaiseilla standardilla: ISO/IEC 9126 (Software product quality) ja ISO/IEC 14598 (Software product evaluation). Käytännöllisistä syistä molempien versioiden hyvät puolet antoivat impulssin luoda uuden SQuaRe - standardisarjan. (Azuma, Scalet 2004, 7.)

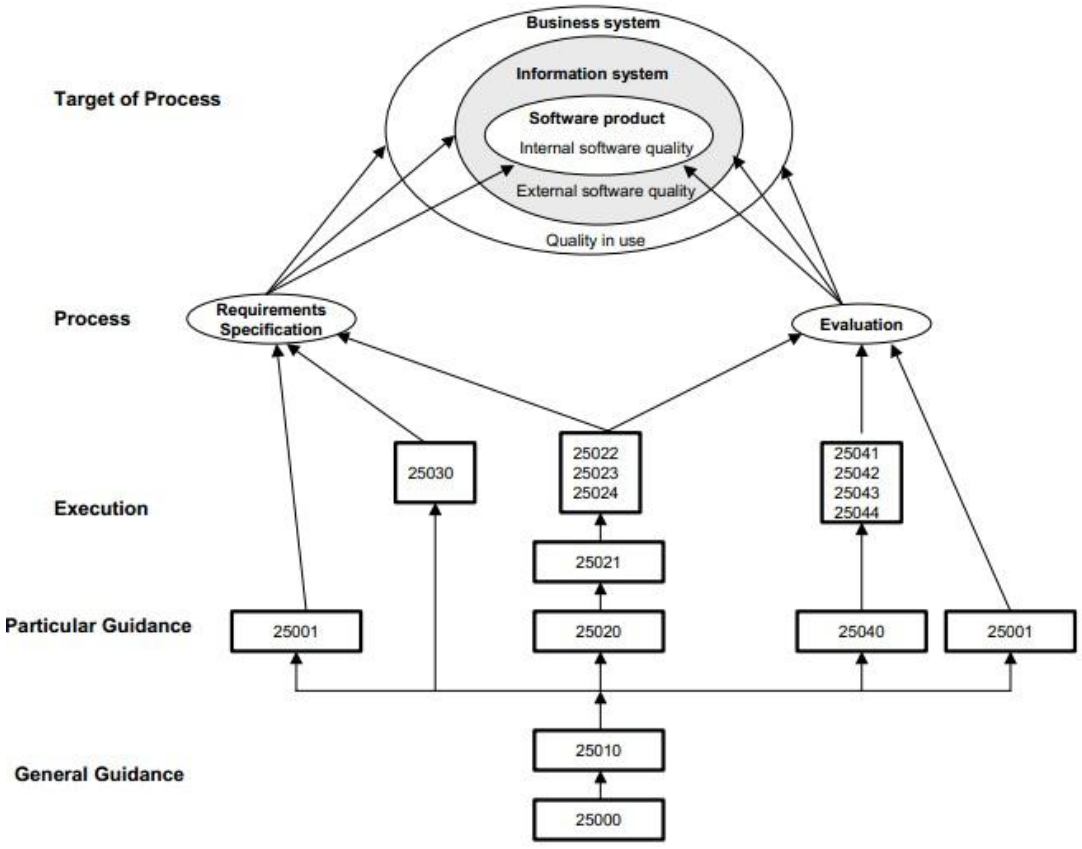


Kuva ISO/IEC 9128 ja 14598 sarjat (Nevalainen 2009, 3.)

Kuva 2 antaa tarkan selvityksen ISO/IEC – sarjojen 9126 ja 14598 toiminnasta. Sarjat on jaettu alakategorioihin, jotka sisältävät erilaisia laadun mittareita. Näitä ovat

esimerkiksi sisäiset, ulkoiset ja käyttölaadun mittarit. Alakategoriat ovat osa suurempia laadun mittaamisen osia, joita ovat resurssi- ja ympäristötekijät, arviointiprosessi, itse ohjelmistotuote ja sen vaikutukset.

Tärkein tarkoitus SQuaRe:n luomiselle on saada aikaiseksi loogisesti organisoitunut, rikastettu ja yhdistetty sarja, joka pitää sisällään kolme täydentävää prosessia: vaatimusten määrittely, mittaus ja arviointi. Standardin tarkoituksena on edesauttaa ohjelmistojen kehitystä ja hankintaa määrittelemällä ja arvioimalla laatuvaatimuksia. Se sisältää kaksiosaisen laatumallinnuksen asiakkaan laatumäärittelyiden linjaukselle koskien kehitysprosessin ominaisuuksia. Lisäksi standardisarja tarjoaa suositellut mittaustavat tuotteen laatutekijöihin, joita voivat käyttää kehittäjät, hankkijat ja arvioijat. (Azuma, Scalet 2004 7.)



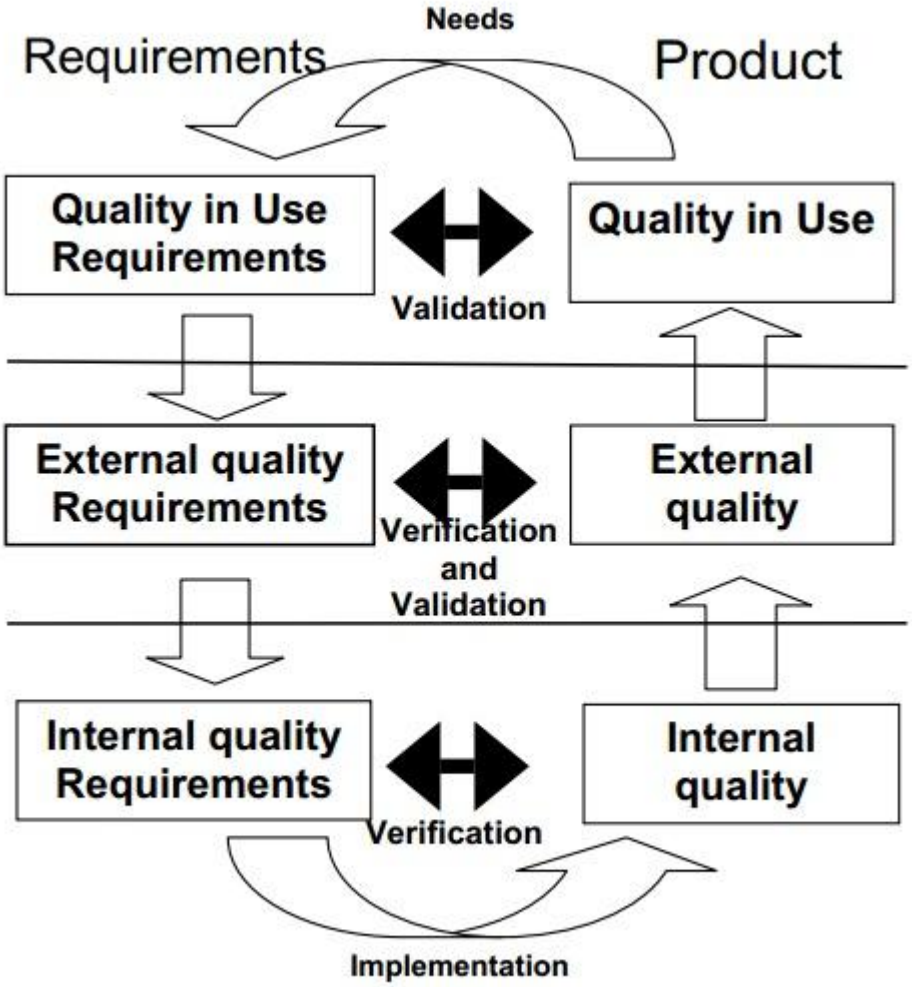
Kuva SQuaRE-standardit (Azuma, Scalet, 14.)

Kuva 3 esittää SQuaRE – mallinnuksen suositellut dokumentit ja standardit eri käyttäjille, jotta he voivat vaivatta löytää sopivat standardit yksilölliseen käyttöön, joka riippuu käyttäjän roolista ja informaatiosta, jota hän tarvitsee (esim. vaatimusmäärit-

tely, arviointi). On kuitenkin suositeltavaa, että jokainen käyttäjä tutustuu omien tarvittavien standardien lisäksi myös 25000 - standardiin, joka antaa johdatuksen SQuaRE:n maailmaan ja tarkentaa mitkä standardit kuvaavat mitäkin prosessia.

3 LAADUN ELINKAARI

Ohjelmistotuotteen laadun elinkaari kuvaa laadun rakentuvan kolmessa vaiheessa: tuotteen ollessa vielä kehitysvaiheessa, tuotteen ollessa toiminnallinen ja tuotteen ollessa käytössä. (Azuma, Scalet 2004, 14.)

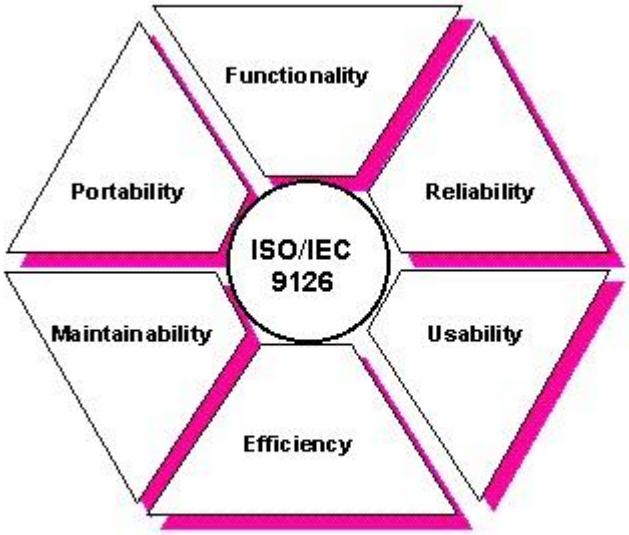


Kuva Elinkaarimallinnus (Azuma, Scalet 2004, 15.)

Ohjelmistotuotteen elinkaarimallia toteuttavan kehitysprosessin on otettava eri vaiheissaan huomioon kaikki kolme kuvassa 4 esitettyä näkökulmaa laatuun. (Azuma, Scalet 2004, 15.)

Laatumalli (kuvassa 4) määrittelee kolme erilaista näkökulmaa laatuun: sisäinen laatu, ulkoinen laatu ja käyttölaatu. Sisäinen laatu tarjoaa ”valkoinen laatikko” – näkökulman ohjelmistoon ja ottaa kantaa ohjelmiston ominaisuuksiin, joihin pystyy vaikuttamaan ohjelmiston kehityksen aikana. Sisäinen laatu yhdistetään lähinnä ohjelmiston staattisiin ominaisuuksiin. Ulkoinen laatu tarjoaa ”musta laatikko” – näkökulman ohjelmistoon ja ottaa kantaa ohjelmiston toteuttamiseen. Käyttölaatu – näkökulma yhdistetään ohjelmiston soveltamiseen käyttöympäristössä eli tiettyjen tehtävien suorittamiseen tiettyjen käyttäjien toimesta. Sisäisellä laadulla on vaikutusta ulkoiseen laatuun, joka puolestaan vaikuttaa laatuun käytössä (kts. kuva 4). Laatumalli palvelee kehyksenä, joka takaa sen, että kaikki laadun näkökulmat on otettu huomioon sisäisen, ulkoisen ja käytön näkökulmista. (Azuma, Boegh 2005, 4.)

Ohjelmiston laatu on sen kyky tarjota määritellyn tasoista palvelua. Palvelun vaadittu taso on määritelty laatumallin mukaisesti. Ohjelmistotuotteen laatumalli on määritelty standardissa ISO/IEC 9126-1 [ISO/IEC 25010]. Standardi (kuva 5) määrittelee kuusi laadun ominaisuutta: toiminnallisuus (ovatko vaadittavat toiminnot käytössä), luotettavuus (kuinka luotettava ohjelmisto on), käytettävyys (onko ohjelmistoa helppo käyttää), ylläpidettävyys (kuinka helppoa ohjelmistoa on ylläpitää ja muokata), siirrettävyys (kuinka helposti ohjelmisto on siirrettävissä toiseen ympäristöön) ja tehokkuus (kuinka tehokasta ohjelmaa on käyttää). (Azuma, Boegh 2005, 4.)



Kuva Laatumallin ominaisuudet (http://www.cse.dcu.ie)

Lisäksi laatumalli määrittelee laadun käyttäen neljää ominaisuutta: vaikuttavuus, tuottavuus, turvallisuus ja asiakastyytyväisyys. Laatuominaisuuksilla on kuusi ali-ominaisuutta ja standardi sallii käyttäjien määritellä alialiominaisuuksia hierarkiseen rakenteeseen. Määritellyt laatuominaisuudet käsittävät kaikki oleelliset laadun näkökulmat suurimmalle osalle ohjelmistotuotteita ja niitä voidaan sellaisenaan käyttää tarkistuslistana laadun täydellisen kattavuuden varmistamiseksi. (Azuma, Boegh 2005, 4.)

4 LAATUMALLIT

Tällä hetkellä on olemassa kolme laatumallia, jotka kuuluvat SQuaRe - sarjaan: käyttölaatu, tuotelaatu ja tiedon laatu. Laatumallit muodostavat yhdessä kehyksen, joka varmistaa, että kaikki laadun tunnuksenomaiset piirteet otetaan huomioon. Nämä mallit tarjoavat laajan kokoelman laatutekijöitä sidosryhmille, kuten: ohjelman kehittäjät, järjestelmäintegroijat, hankkijat, omistajat, ylläpitäjät, sopimusosapuolet, laadunvarmistuksen ja -hallinnan ammattilaiset sekä käyttäjät. (ISO/IEC 2010, 2.)

Kaikki laadun piirteet näissä malleissa eivät tietenkään ole olennaisia kaikille sidosryhmille. Kuitenkin jokaiselle sidosryhmälle tulisi esittää täydellinen raportti koskien laatutekijöitä jokaisessa mallinnusvaiheessa ennen kuin käytettävät piirteet lyödään lukkoon, esimerkiksi tuotteen ja järjestelmän käyttövaatimukset tai arviointikriteerit. (ISO/IEC 2010, 3.)

4.1 Käyttölaatu (25010)

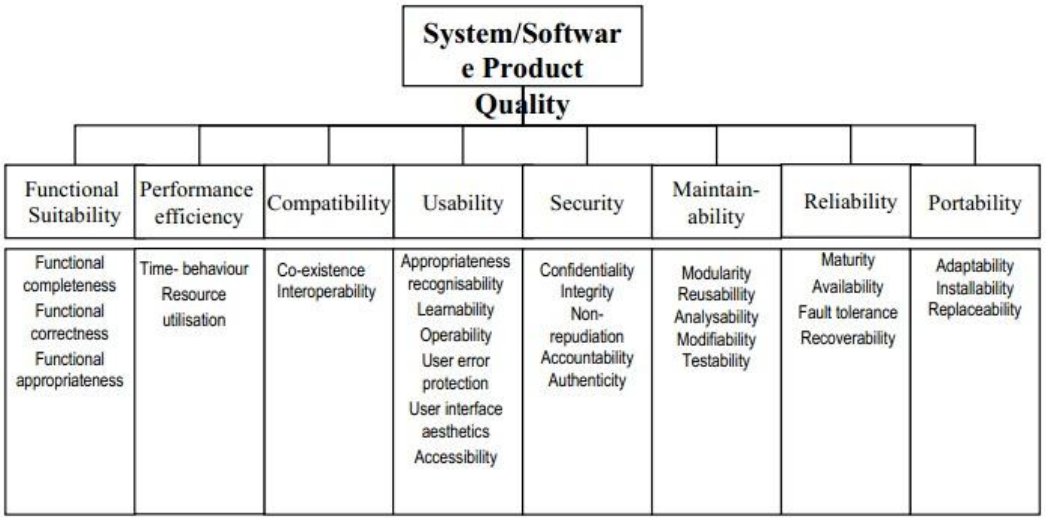
Käyttölaadun malli (kuvassa 6) määrittelee viisi tunnusomaista piirrettä, jotka liittyvät järjestelmän vuorovaikutukseen: tehokkuus, suorituskyky, tyytyväisyys, riskinhallinta ja kattavuus. (ISO/IEC 2010, 3.)

Effectiveness
Efficiency
Satisfaction
Usefulness
Trust
Pleasure
Comfort
Freedom from risk
Economic risk mitigation
Health and safety risk mitigation
Environmental risk mitigation
Context coverage
Context completeness
Flexibility

Kuva Käyttölaatumalli (ISO/IEC 2010, 3.)

Järjestelmän käyttölaatu kuvaa, millainen vaikutus tuotteella (järjestelmä- tai ohjelmistotuote) on sidosryhmiin. Vaikutuksen määrittelee tuotteen laatu, laitteisto ja toimintaympäristö sekä käyttäjien luonteenpiirteet, tehtävät ja sosiaalinen ympäristö. Kaikki nämä osatekijät osallistuvat ohjelmiston laadun varmistamiseen. (ISO/IEC 2010, 3.)

4.2 Tuotelaatu (25012)



Kuva Tuotelaatumalli (Nevalainen 2009, 8.)

Tuotelaadun malli (kuvassa 7) luokittelee järjestelmän/ohjelmiston laatuominaisuudet kahdeksaan omaan luokkaan: toiminnallinen soveltuvuus (functional suitability), suorituskyky (performance efficiency), yhteensopivuus (compatibility), käytettävyys (usability), luotettavuus (reliability), turvallisuus (security), ylläpidettävyys (maintainability) ja siirrettävyys (portability). Jokainen laatuominaisuus muodostuu joukosta alipiirteitä. (ISO/IEC 2010, 3.)

4.3 Tiedon laatu (25012)

Tiedon laatu määrittää informaation laadun. Jokainen tunnusomainen piirre/alapiirre tiedon laatu -mallissa vaihtelee tärkeydeltään ja niillä kaikilla on erilaiset mittarit riippuen järjestelmän sovellusten/ohjelmistojen vaatimuksista. (ISO/IEC 25012, 9.)

Tämä käytettävä standardi ottaa huomioon kaikki tiedon tyypit (esim. merkkijonot, päiväykset, numerot, kuvat, äänet, jne.), määrätyt tiedon arvot ja käytännöt, jotka vaikuttavat tiedon rakenteeseen ja tiedon välisiin suhteisiin (esim. johdonmukaisuuteen tiedon välillä, sen ollessa yhtenäisyydessä muun tiedon kanssa tai itsenäisenä kokonaisuutena). (ISO/IEC 25012, 9.)

Kuva Tiedon laadun malli (ISO/IEC 25012, 14.)

Kuva 8 tarkentaa tiedon laadun mittaamisen. Tiedon ulkoinen ja sisäinen laatu on jaettu kuuteen osaan ja niiden määrittelyihin: toiminnallisuus (functionality): konsistenssi (consistency), esilläolo (currency), täydellisyys (completeness), tarkkuus (precision), virheettömyys (accuracy), yhteensopivuus (interoperability), turvallisuus (security) ja toiminnallinen vaatimustenmukaisuus (functionality compliance); luotettavuus (reliability): saatavuus (availability), palautuminen (recoverability) ja luotettavuuden vaatimustenmukaisuus (reliability compliance); käytettävyys (usability): ymmärrettävyys (understandability), hallittavuus (manageability), vetovoimaisuus (attractiveness) ja käytettävyyden vaatimustenmukaisuus (usability compliance); tehokkuus (efficiency): resurssien hyödyntäminen (resource utilization) ja tehokkuu-

den vaatimustenmukaisuus (efficiency compliance); ylläpidettävyys (maintainability): analysoitavuus (analysability), muunneltavuus (changeability) ja ylläpidettävyyden vaatimustenmukaisuus (maintainability compliance) ja siirrettävyys (portability): sopeutuvuus (adaptability) ja siirrettävyyden vaatimustenmukaisuus (portability compliance).

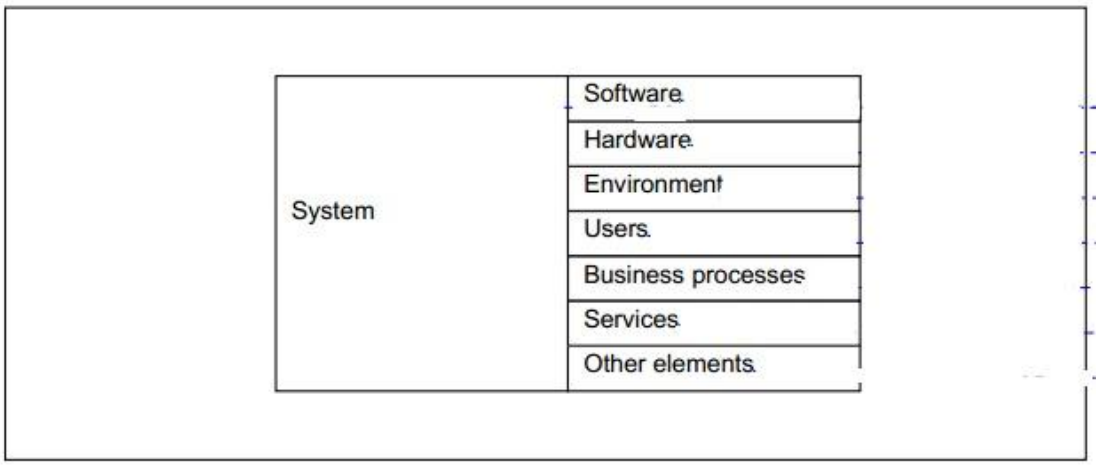
Tiedon käyttölaatu voidaan mallintaa samalla tavalla kuin sisäiset ja ulkoiset laatutekijät. Käyttölaatu on jaettu neljään osaan: tehokkuus (effectiveness): täsmällisyys (timeliness), soveltuvuus (adequacy) ja tehokkuuden vaatimustenmukaisuus (effectiveness compliance); tuottavuus (productivity): relevanssi (relevance), tulkinnallisuus (interpretability) ja tuottavuuden vaatimustenmukaisuus (productivity compliance); turvallisuus (safety): turvallisuuden merkittävyys (safety relevance) ja turvallisuuden ohjeidenmukaisuus (safety compliance) ja tyytyväisyys (satisfaction): uskottavuus (credibility), saavutettavuus (accessibility) ja tyytyväisyyden vaatimustenmukaisuus (satisfaction compliance).

5 OHJELMISTOJEN LAATUVAATIMUSTEN KEHYS

Ohjelmisto on yleensä osa suurempaa järjestelmää. Sen takia saattaa olla hyödyllistä ottaa näkökulmaksi järjestelmän näkökulma. Järjestelmä määritellään yhdistelmäksi vuorovaikutuksessa olevia elementtejä, joiden tavoitteena on saavuttaa yksi tai useampia asetettuja päämääriä. Määritelmä antaa runsaasti vapausasteita tulkita järjestelmä-käsitettä. Tulkinta riippuu pitkälti ohjelmiston toiminnallisuuden huomioonottavasta tarkastelukulmasta. (Azuma, Boegh 2005, 2)

Azuma & Boegh (2005, 2) havainnollistavat järjestelmä – käsitettä seuraavan kolmen esimerkin avulla. Yksi esimerkki on lentokoneen moottorin hallintajärjestelmä, toinen esimerkki on lentokoneen moottori kokonaisuudessaan ja kolmas esimerkki on koko lentokone. Lentokone voidaan nähdä yhdistelmänä elementtejä (moottorit, siivet, ym.) Jokainen elementti voidaan myös nähdä omana järjestelmänään. (Azuma, Boegh 2005, 2)

Kuvassa 9 on demonstroitu, mitä järjestelmä voi tarkoittaa. Jokainen saattaa tulkita sanan järjestelmä eri tavalla, jolloin järjestelmään kuuluvien elementtien esitetty määrä voi vaihdella. Pääpiirteisesti järjestelmä koostuu tietenkin itse laitteistoista, sovelluksista, ohjelmistoista, käyttäjistä, toiminnoista ja muista elementeistä. Toisin sanoen järjestelmä koostuu osista ja osien välisistä suhteista, jotka muodostavat kokonaisuuden.

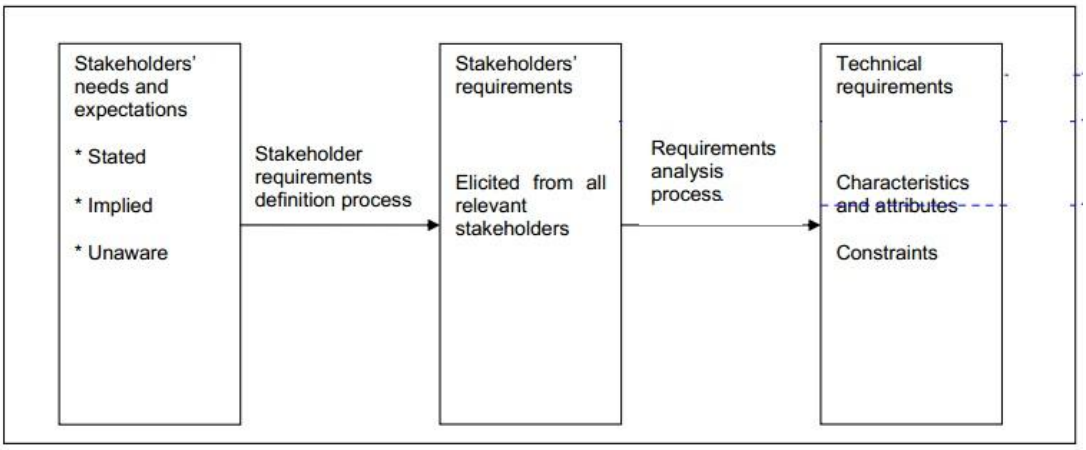


Kuva Järjestelmän elementit (ISO/IEC 25030, 2)

5.1 Sidosryhmien vaatimukset ja tekniset vaatimukset

Analyysiprosessin tavoitteena on muuttaa sidosryhmien vaatimukset teknisiksi järjestelmävaatimuksiksi, joita voidaan käyttää halutun järjestelmän toteuttamiseksi (katso kuva 9). Teknistä näkökulmaa vaatimuksista kutsutaan teknisiksi vaatimuksiksi. Tekniset vaatimukset ovat todennettavia ja kertovat, mitä ominaisuuksia järjestelmällä täytyy olla sidosryhmien vaatimusten täyttämiseksi. (Azuma, Boegh 2005, 3.)

Järjestelmä muodostuu usein erilaisista elementeistä, joista jokaisella on erityiset ominaisuudet ja jotka palvelevat erilaisia tarkoituksia järjestelmässä. Jotta järjestelmä olisi toimintakuntoinen, tekniset vaatimukset tulee muotoilla järjestelmän elementtikohtaisiksi vaatimuksiksi. Kun eri elementit toimivat yhdessä tarjotakseen järjestelmälle toimintoja, vaatimuksia yksittäiselle elementille ei voida nähdä eristettynä vaan ainoastaan laajempana näkökulmana, jossa ovat mukana myös vaatimukset muille järjestelmän elementeille. (Azuma, Boegh 2005, 3.)



Kuva Sidosryhmien vaatimukset (ISO/IEC 25030, 3)

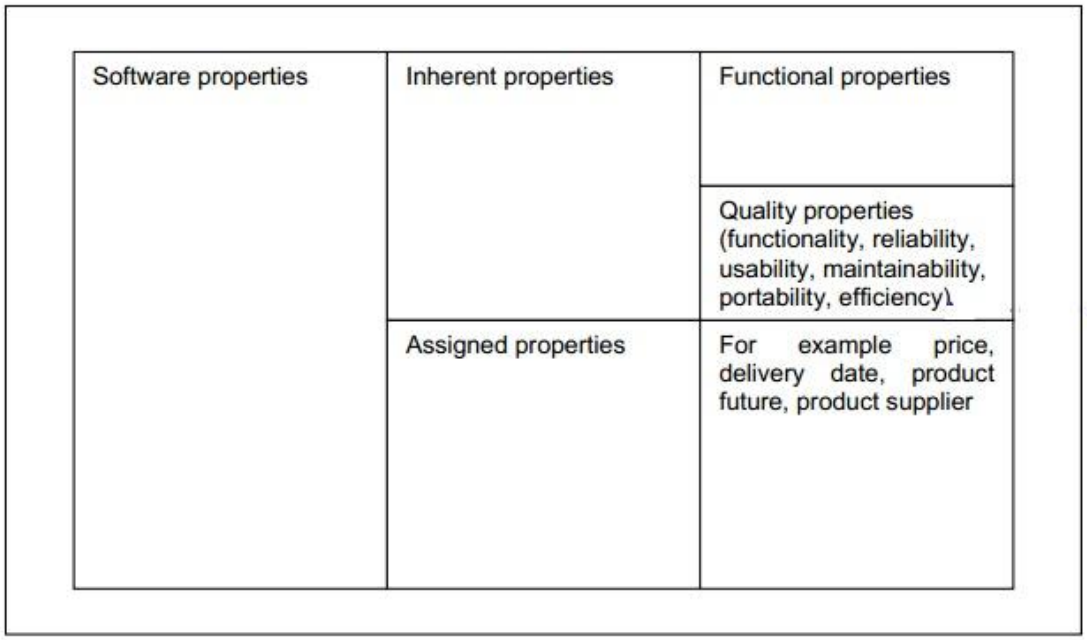
Teknisistä vaatimuksista järjestelmälle voi seurata vaatimuksia myös esimerkiksi ohjelmistolle. Järjestelmävaatimus otetaan vastaan sidosryhmiltä, mutta aina ei ole selkeää, asettaako järjestelmävaatimus lisäksi ohjelmistovaatimuksia. (Azuma, Boegh 2005, 3.) Ohjelmistovaatimukset ovat yleensä ohjelman toimintoja, jotka määrittelevät, miten kyseinen asiakasvaatimus toteutettavassa ohjelmistossa ”tarjoillaan” käyttäjälle. (Haikala, Mikkonen 2011, 62.) Järjestelmävaatimukset voidaan ottaa käyttöön eri tavoin, esimerkiksi laitteistona, ohjelmistona tai liiketoimintaprosessina. (Azuma, Boegh 2005, 3.)

5.2 Ohjelmiston ominaisuudet

Jotkin ohjelmistotuotteiden ominaisuudet ovat sisäisiä ja jotkut ohjelmistolle annettuja ulkoisia. Ohjelmistotuotteen laadukkuus määrittää sen sisäiset ominaisuudet. Sisäiset ominaisuudet muodostuvat ohjelmistolle määritellyistä vaatimuksista (toiminnalliset ja ei-toiminnalliset). Annetut ominaisuudet taas ovat ulkoisia, ohjelmistolle jollain perusteella annettuja ja sen varsinaiseen laatuun vaikuttamattomia. Toki toimitusajan piteneminen tai liian kallis hinta saattavat vaikuttaa laatutuntemukseen, jonka asiakas tuotteesta saa. Esimerkkejä sisäisistä ominaisuuksista ovat koodirivien määrä ja ohjelmiston tarjoama laskennallinen tarkkuus. (Azuma, Boegh 2005, 4.)

Sisäiset ominaisuudet voidaan siis luokitella joko funktionaalisiksi eli toiminnallisiksi ominaisuuksiksi tai laatuominaisuuksiksi (ei-toiminnallisiksi). Toiminnalliset omi-

naisuudet määrittävät mitä ohjelmisto pystyy tekemään. Nämä ominaisuudet ovat sidonnaisia ympäristöön, jossa ohjelmistoa voidaan suorittaa (esim. käyttöjärjestelmä), muiden ohjelmistojen rajapintoihin ja toimintoihin, joita ohjelmisto pystyy suorittamaan. Laatuominaisuudet määrittävät, miten hyvin ohjelmisto suoriutuu tietyn laatuominaisuuden näkökulmasta. Toisin sanoen laatuominaisuudet esittävät sitä astetta, jolla ohjelmisto pystyy tarjoamaan ja ylläpitämään määriteltyjä palveluja. Laatu on luontainen ominaisuus ohjelmistotuotteelle. Ulkoista ominaisuutta ei sen takia nähdä laatuominaisuutena ohjelmistossa, sillä sitä pystytään muuttamaan ilman ohjelmiston muuttamista. Kuva 11 kuvaa tätä luokittelua ohjelmiston ominaisuuksissa. (Azuma, Boegh 2005, 4.)



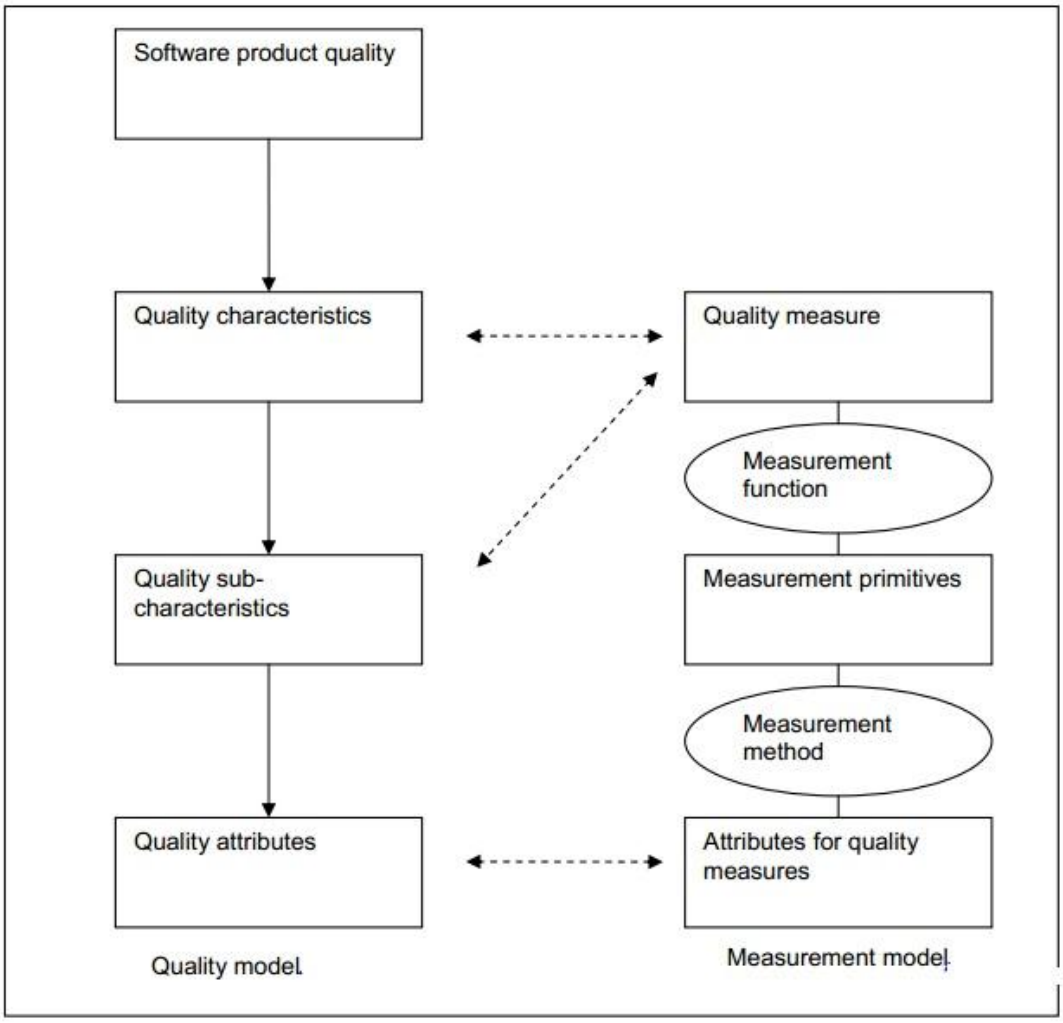
Kuva Ohjelmiston ominaisuudet (ISO/IEC 25030, 5)

5.3 Ohjelmiston laadun mittausmalli

Ohje ohjelmistotuotteen laadusta on määritelmä ohjelmiston kyvystä suorittaa ja ylläpitää tiettyä palvelun tasoa. Ohjelmiston laatu voidaan siis ilmaista arvoasteikolla, josta selviää, miten hyvin ohjelmisto pystyy suorittamaan ja ylläpitämään tiettyä palvelutasoa. Lähestymistapa ohjelmiston laadun mittaukseen on antaa ominaisuuksille arvoja siitä, kuinka ohjelmisto täyttää sille asetetut vaatimukset. (Azuma, Boegh 2005, 5.)

Sisäisiä ohjelmiston ominaisuuksia, joita pystytään kvantitatiivisesti (määrällisesti) tai kvalitatiivisesti (laadullisesti) erottamaan, kutsutaan attribuuteiksi. Laatuattribuutteja mitataan antamalla niille arvo (degree), siitä, miten hyvin ohjelmistotuote kykenee tarjoamaan ja ylläpitämään sille määriteltyä palvelun tasoa. Laatuattribuutit kuuluvat yhteen tai useampaan (ali-)ominaisuuteen. (Azuma, Boegh 2005, 5.)

Laatuattribuutteja mitataan käyttämällä mittausmetodeita (kuva 12). Mittausmetodi on looginen sarja toimintoja, joilla kvantifioidaan (määritetään) attribuutin arvo suhteessa määritettyyn asteikkoon. Mittausmetodin soveltamisen tuloksesta käytetään termiä alkeismittaus (base measure). Laadun ominaisuudet ja aliominaisuudet voidaan kvantifioida soveltamalla mittausfunktioita. Mittausfunktio on algoritmi, jolla yhdistetään joukko alkeismittauksia. Mittausfunktion tulosta kutsutaan laatuarvoksi. Tällä tavoin laatuarvoista tulee kvantifikaatioita (määrittelyjä) laatuominaisuuksille ja aliominaisuuksille. Laatuominaisuuden tai aliominaisuuden mittaamiseen voidaan käyttää useampaa kuin yhtä laatuarvoa. (Azuma, Boegh 2005, 5.)



Kuva Laadun mittausmalli (Azuma, Boegh 2005, 6)

Mittausfunktio antaa yhden tulkinnan ohjelmiston laatuominaisuudesta ja laatumittarin kohdearvo ilmaisee laatuvaatimusta. Samoin laatumittauksen todellinen arvo kuvaa havaittua ohjelmiston laatua. (Azuma, Boegh 2005, 6.)

Ohjelmiston laatuvaatimukset samoin kuin kaikki muutkaan vaatimukset, eivät näy eristettynä, vaan niitä täytyy katsoa laajemmassa kontekstissa. Ohjelmiston laatuvaatimuksilla on erityisen läheinen suhde funktionaalisiin (toiminnallisiin) vaatimuksiin. (Azuma, Boegh 2005, 6.)

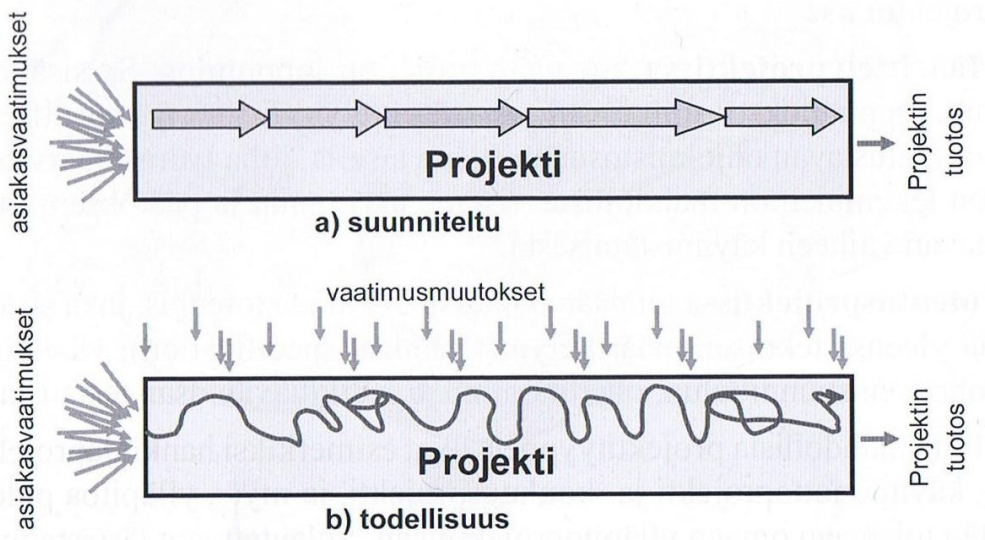
Joissain tilanteissa on järkevää määritellä laatuvaatimus valmiille ohjelmistotuotteelle, kun taas toisissa tilanteissa laatuvaatimus soveltuu vain osaan ohjelmistotuotetta. Esimerkiksi jotkin funktiot ovat relevantteja vain tietyille käyttäjille, ja niillä on tie-

tyt laatuvaatimukset, jotka ovat eriäviä muille käyttäjille ja muihin tarkoituksiin tarkoitettujen funktioiden kanssa. Tämän takia on tärkeää määritellä, mikä osuus ohjelmistotuotteesta on relevanttia ohjelmiston laatuvaatimusten suhteen. Toisin sanoen laatuvaatimus liittyy osaan ohjelmistotuotetta (joukkoon funktioita). Esimerkiksi jotkin funktiot voivat olla tarkoitetut yleisesti loppukäyttäjille ja voivat näin ollen vaatia matalaa virhetoleranssia, kun taas toinen joukko funktioita voi olla tarkoitettu asiantuntijoille ja näin ollen sallii suuremman virhetoleranssin. Molemmissa tapauksissa virhetoleranssimekanismi ja virhetoleranssin arviointi täytyy määritellä tarkasti. (Azuma, Boegh 2005, 7.)

6 VAATIMUSMÄÄRITTELY LAADUN PERUSTANA

Toimittajan kannalta ohjelmistoprojektin lähtökohta ovat asiakkaalta tulevat vaatimukset, jotka kuvaavat projektin sisällön mahdollisimman tarkasti asiakkaan kannalta. Jos vaatimukset ovat tarpeeksi yksityiskohtaiset, projekti on ainakin periaatteessa kuvaus vaatimuksista toteutukseksi. (Haikala, Mikkonen 2011, 21.)

Käytännössä vaatimusten kuvaaminen näin tarkasti on mahdotonta, ja vaatimuksiin tulee aina lisäyksiä, tarkennuksia ja muutoksia koko projektin ajan, sillä usein myös yrityksen omiin lähtökohtiin tulee muutoksia ajan myötä. Myös monet toteutuksen yksityiskohdat voivat vaikuttaa vaatimuksiin tavalla tai toisella, ja vaikka vaatimukset muuten tunnettaisiinkin, näiden yksityiskohtien ottaminen huomioon ennen toteutusta on vaikeaa. Lisäksi uudelleenkäyttö, kattaen sekä mukana olevan toteutusryhmän toimintatavat sekä mahdollisesti jo olemassa olevat ohjelmistot, voivat johtaa vaatimusmuutoksiin. Siksi usein mieluummin tyydytään ”riittävän tarkkoihin” vaatimuksiin ja yritetään varautua muutoksiin, joita joka tapauksessa tulee projektin aikana. (Haikala, Mikkonen 2011, 22.)



Kuva Tuotantoprosessi (Haikala, Mikkonen 2011, 22.)

Jatkuvien muutosten lisäksi toinen nykyohjelmistotuotantoa luonnehtiva seikka on, että juuri koskaan uusia ohjelmistoja ei määritellä, suunnitella, ja toteuteta kokonaan puhtaalta pöydältä, vaan ne perustuvat johonkin ohjelmistoalustaan, yleiseen sovelluskehykseen, jo olemassa olevaan, aiempaan versioon – tai sitten johonkin samantapaiseen järjestelmään, joka on saatavilla johonkin toiseen ympäristöön tai hivenen erilaiseen käyttötarkoitukseen. (Haikala, Mikkonen 2011 23.)

Monessa mielessä ohjelmistojen tekeminen on spesifikaatioiden tekemistä, alkaen karkean tason määrittelystä päättyen ohjelmointikielellä tehtyyn määrittelyyn, joka on suoraan käännettävissä suoritettavaan muotoon. Spesifikaatiota työstettäessä siihen lisätään yksityiskohtia, jotka muokkaavat sitä lähemmäs koneellisesti toteutettavaa muotoa. Mitä lähempänä toteutusta ollaan, sitä enemmän määrämuotoisuutta spesifikaatiolta yleensä vaaditaan. (Haikala, Mikkonen 2011 70.)

Koska määrämuotoisuus vie spesifikaatiota lähemmäs toteutusta, voivat abstrakteimmat ja yleisimmät spesifikaatiot olla varsin vapaamuotoisia. Niiden pohjimmainen tarkoitus on muodostaa ensimmäinen sopimus asiakkaan ja toteuttajan välille. Siksi spesifikaatio tulisi aina tehdä siten, että se on asiakkaan kannalta ymmärrettävä ja yksiselitteinen. Käytännössä spesifikaatioihin jää usein aukkoja ja epätäsmälli-

syyksiä, joita korjataan ja tarkennetaan projektin edetessä. (Haikala, Mikkonen 2011 70.)

Vaatimusmäärittelyn tuloksena on yleensä dokumentti, jota kutsutaan nimellä toiminnallinen määrittely. Se sisältää tavallisesti sekä asiakas- että ohjelmistovaatimukset. Muita määrittelyn yhteydessä mahdollisesti syntyviä dokumentteja ovat alustava käyttöohje ja testaussuunnitelma. (Haikala, Mikkonen 2011 68.)

7 DOKUMENTOINTI OSANA OHJELMISTOSUUNNITTELUA

Ohjelmistosuunnittelun ja dokumentoinnin välinen suhde on usein ongelmallinen. Osana ongelmanratkaisua syntyy usein paljonkin dokumentaatiota, mutta jotta siitä olisi jotain hyötyä, dokumentaation pitäisi olla ajantasaista eikä heijastella järjestelmän kehitysvaihetta silloin, kun dokumentti on kirjoitettu. Toisaalta kaikkea dokumentaatiota ei luultavasti kannatakaan ylläpitää, sillä osa siitä on ollut tarpeen vain osana oppimis- ja kehitysprosessia. (Haikala, Mikkonen 2011, 194.)

Koska dokumentaation ylläpito vaatii merkittävän työpanoksen, mitään turhaa ei kannata dokumentoida. Tärkeä periaate on ”say it once and only once”, eli kuvauksen tulisi olla niin tiivis kuin mahdollista. Samalla vältetään ristiriitaisuudet, joita ylläpidon myötä dokumentaatioon muuten voisi jäädä. (Haikala, Mikkonen 2011, 194.)

Kaikista ohjelmistotyössä syntyvistä dokumenteista keskeisin on arkkitehtuurin kuvaus. Se kattaa suunnittelijoiden näkemyksen järjestelmästä sekä sen perusrakenteen ja – filosofian ja toimii siten myös ylläpito-ohjeena ja perehdytysmateriaalina järjestelmään. Toisin sanoen tekninen määrittely on paitsi arkkitehtuurin kuvaus myös sen käyttöohje. Siksi dokumenttia tehdessä tulisi keskittyä nimenomaan niihin asioihin järjestelmässä, joihin ei saisi tulla muutoksia, ja vastaavasti surutta rajata dokumentin ulkopuolelle kokonaisuuden kannalta tarpeettomat yksityiskohdat. Tällöin kuvausta voidaan käyttää myös ilman, että sitä joudutaan ylläpitämään, eikä se vanhene niin helposti pienten muutosten myötä. (Haikala, Mikkonen 2011, 195.)

8 REVERSE ENGINEERING

Reverse engineering on menetelmä, joka tarkoittaa valmiin tietojärjestelmän purkamista takaisin alkutekijöihinsä, esimerkiksi lukemalla olemassa olevia dokumentteja ja lähdekoodia tai ajamalla järjestelmää.

Tavoitteena takaisinmallinnuksella on hallita tietojärjestelmän monimutkaisuutta. Mitä syvemmälle järjestelmää pääsee, sitä paremmin voi tarjota vaihtoehtoisia näkemyksiä ja jäljittää kadonneita tietoja.

Kuva Reverse Engineering -malli (Harsu 2011, 1.)

Kuvassa 14 on luonnosteltu takaisinmallintamista. Prosessien edetessä vaatimuksista aina tuotantokäytössä olevan tuotteen tai järjestelmän käyttäytymiseen ympäristössä, voidaan tuote tai järjestelmä purkaa takaisin alkutekijöihinsä, jotta sen toimintaperiaate ja elementtien toiminnot saadaan selvitettyä. Tätä ketjua kutsustaan siis nimellä reverse engineering.

9 REVERSE ENGINEERINGIN HYÖDYNTÄMINEN OHJELMISTOMAAILMASSA

Aikaisemmin mainitut laatutekijät ovat toinen osa opinnäytetyötäni. Sain mahdollisuuden osallistua Porin yliopistokeskuksen tietojärjestelmien käytettävyystutkimukseen, jossa pääsin soveltamaan laatutekijöitä ja niiden mittareita tuotantokäytössä olevaan järjestelmään. Aikaansaamistamme analyyseista tehtiin tarvittavat muutokset. Salassapitosopimuksen vuoksi en voi mainita toimeksiantajaa tai tietojärjestelmän nimeä. Kuvailen pintapuolisesti järjestelmää ja sen toimintaa. Tässä työssä en paneudu tyylimuutoksiin, vaan keskityn vain laatua määrittäviin toiminnallisuuksiin ja niiden parantamiseen. Analysoin järjestelmän käytettävyyttä sisäisen laadun näkökulmasta (kuva 8. (usability)).

Käytössämme ollut tietojärjestelmä on pääasiassa tapahtumatiedon ja sidosryhmätiedon ylläpitoa varten. Käyttäjillä on sidoksia eri ryhmiin, järjestöihin, tapahtumiin ja organisaatioihin. Käyttäjät voivat kommunikoida viestien välityksellä, ilmoittautua erilaisiin tapahtumiin ja lukea heille suunnattuja infotietoja tulevista tapahtumista foorumin tapaisissa keskusteluosioissa. Järjestelmä on periaatteessa kolmiosainen, millä tarkoitan, että järjestelmää käyttää kolme eri organisaatiota. Toiminnallisuudeltaan järjestelmä on yhtenäinen, lukuunottamatta muutamia pieniä poikkeuksia, jotka on tilattu toimeksiantajalta tietylle organisaatiolle. Tähdellisin laatutekijä tässä järjestelmässä projektimme kannalta on käytettävyys. Järjestelmästä tullaan tekemään kaikkia laatutekijöitä kuvaava analyysi (toiminnallisuus, käytettävyys, tehokkuus, turvallisuus, jne.). Tämä prosessiketju on kuitenkin erittäin laaja ja se on tällä hetkellä niin sanotusti ”ongoing”, joka tarkoittaa, että se on jatkuvaa. Projektiosuus päättyy kesään mennessä, joten emme voi vaikuttaa muihin laatutekijöihin. Ryhmämme tehtäväksi siis tuli järjestelmän purkaminen ja käytettävyyden mittareiden analysoiminen, tarvittavien muutoksien tekeminen ja käytettävyyden attribuuttien vaikutuksen heijastuminen käyttäjän näkökulmasta. Järjestelmä on siis jo tuotantokäytössä. Meillä ei ollut käytössä minkäänlaisia käyttöohjeita ja edessämme oleva järjestelmä oli niin sanotusti valmis. Reverse engineering toimi siis perustana järjestelmän käytettävyyden mallintamista varten. Projektityössä käyttämiemme UML - luokkakaavioiden tarkoitus oli avata järjestelmä mahdollisimman pieniksi osiksi, eli mikä oli keskeistä järjestelmässä ja mikä oli irrallisena järjestelmän ydintoiminnasta.

Lisäkseni ryhmään kuului kolme henkilöä. Työnjako projektissa oli melko selkeä, sillä kahdella ryhmäläisistäni on ohjelmointitaustaa, jolloin dokumentointi ja luonnostelu jäivät minun ja työparini harteille. Aikaisemmin työssäni painotin dokumentoinnin tärkeyttä ja lähes täydellistä vaatimusten määrittelyä. Käyttämästämme tietojärjestelmästä ei kuitenkaan ollut dokumentointia, joten jouduimme aloittamaan valmiista tuotteesta. Edellä mainitun voi nähdä sekä hyvänä että huonona asiana. Hyvää siinä tietenkin on, että saa tuntumaa, kuinka tietojärjestelmiä ylipäänsä dokumentoidaan. Aikataulumme oli jaettu ydinosiin, jolloin deadlinet olivat selkeitä. Tämän ansiosta pystyimme laatimaan hyvän iteraatiosuunnitelman, mikä helpottaa projektin tekemistä huomattavasti. Projekti käynnistyi vuoden 2012 alussa.

Täysin uutta tietojärjestelmää käytettäessä ensimmäisenä tietenkin tulee esiin itse käyttöliittymään totutteleminen. Järjestelmä tuntui aluksi todella monimutkaiselta ja hankalalta käyttää, mutta ajan kuluessa käytettävyys nopeutui ja selkeytyi. Oli selvää, että järjestelmä oli saatava purettua auki ja muutettua täten selkeämmäksi käyttää tulevaa asiakasta ajatellen. Tietojärjestelmään annetaan käyttökoulutusta, mutta selkeyttämisen vuoksi, niin visuaalisen kuin käytettävyydenkin, takaisinmallinnus oli tehtävä.

Projektin suunnittelu

Jokainen ryhmän jäsen osallistui silmänliiketestiin, joka tarkoitti kylmiltään tutustumista käytettävään tietojärjestelmään reilun tunnin ajan ja näiden perusteella jokainen luonnosteli oman arvionsa vanhasta ulkoasusta. Tarkempaa analyysia ei kuitenkaan vielä laadittu, vaan pohdittiin pelkästään pintapuolisesti mitä järjestelmästä tulee mieleen. Ensimmäinen iteraatio (iterointi on yleinen nimitys menetelmille, joissa samoja työvaiheita toistetaan kunnes haluttu tulos on saavutettu), johon alustavat analysoinnit, ensivaikutelmat ja johtopäätökset kuuluivat, palautettiin 25.1.2012.

Tämän jälkeen järjestelmästä luotiin paljon tarkempi analyysi. Ulkoasua analysoitiin pääpiirteisesti ja annettiin rakentavaa kritiikkiä. Analyysit mietittiin ensin yksin, jonka jälkeen sovittiin tapaamisia ja keskusteltiin itse järjestelmästä. Yhdessä pohtimamme asiat, jotka päätyivät analyysilistaukseen, olivat opittavuus (kuinka helppo järjestelmän käyttö oli oppia), tehokkuus (kun käyttö oli opittu, kuinka tehokkaasti

sitä voitiin käyttää), muistettavuus (kuinka helppo järjestelmän käyttö on muistaa, jos sitä käyttää harvakseltaan), virheet (kuinka usein ja kuinka pahoja virheitä käyttäjät tekevät järjestelmää käyttäessään) ja tyytyväisyys (kuinka mielekästä järjestelmää on käyttää).

Löydökset koottiin yhteen käyttöliittymäanalyysiksi (kuvat 15 ja 16). Itse analyysi koostui yhdeksästä eri painokertoimin pisteytetystä osa-alueesta: ohjeiden puuttuminen, epäselvä värinkäyttö, tasapainottomuus ja epäsymmetrisyys, sivujen hierarkian epäselvyys, lomakkeiden validointi, järjestelmän toiminnallisuuden epäloogisuudet, maallikolle tuntemattomat kentät, kirjoitusvirheet sekä virheilmoitukset ja bugit.

Lauri Limnell	Vakavuus	Toistuvuus	Yht
Ohjeiden puuttuminen	2	3	6
Epäsopiva värinkäyttö	2	4	8
Tasapainottomuus ja epäsymmetrisyys	3	3	9
Sivujen hierarkian epäselvyys	3	2	6
Lomakkeiden validointi	3	2	6
Järjestelmän toiminnallisuuden epäloogisuudet	4	3	12
Maallikolle tuntemattomat kentät	3	3	9
Kirjoitusvirheet	2	4	8
Virheilmoitukset ja bugit	5	2	10

Kuva Käyttöanalyysi 1.1 (numerot kuvitteelliset)

Kuvat 15 ja 16 näyttävät analyysin osa-alueet, niiden vakavuuden, toistuvuuden ja yhteispistemäärän parannusehdotuksineen. Yhteispistemäärä saatiin kertomalla vakavuusarvo toistuvuusarvon kanssa (esimerkiksi ohjeiden puuttuminen: 2*3=6).

Esimerkkejä	Parannusehdotukset
Epäselvä sivu, ei ohjeita	Kuvaukset paremmiksi
Epäselvät värien yhdistelmät	Tarkka väriskaala
Elementit sekaisin	Sommittelua
Navigointi pitää opetella "ulkoa" että selkeä	Mahdollinen polutus
Tyhjiä lomakkeita läpi	Tarvittavat kentät
Listaan / Paluu samat funktiot	Hienosäätöä nimiin
lorem ipsum	Paremmet nimet ja ohjeistus
lorem ipsum	Oikoluku
lorem ipsum	Korjaus ja testaus

Kuva Käyttöanalyysi 1.2

Käyttöliittymäanalyysistä laadittiin käytettävyysraportti, joka havainnollisti havaittujen ongelmien priorisoinnin, ongelmien vakavuusluokittelun sekä järjestelmän ele-

mentit ja niiden ongelmat sekä mahdolliset parannusehdotukset. Tämä kokonaisuus palautettiin 22.2.2012. Se kuului toiseen iteraatioon, jossa palautettiin järjestelmän analyysi sekä karkea spesifikaatio ja mahdolliset luonnokset, jotka olivat syntyneet ohessa.

Projektista laadittiin kuukausien aikana siis useita raportteja, esimerkkinä juuri edellä mainittu käytettävyysraportti, josta tulevat ongelmat ja niiden mahdolliset parannusehdotukset on poimittu. Loppuraportti ja näkymäraportti on jätetty pois tästä tutkielmasta, koska ne eivät koske laatutekijöitä käytettävyyden näkökulmasta mitenkään eivätkä ne heijastu järjestelmän käytettävyyteen ja sen parantamiseen, joka oli tämän projektin ydin. Suurin osa ongelmista on saatu korjattua ja järjestelmää on päivitetty lisää toiveiden perusteella.

Kriittiset ongelmat

Taulukoista voitiin päätellä, että mielestämme vakavimpia ongelmia olivat epäloogisuudet järjestelmän toiminnassa (käyttäjä odottaa järjestelmän toimivan eri tavalla kuin se käytännössä toimii). Niiden toistuvuus on kuitenkin selvästi vähäisempää kuin esim. epäsopivan värinkäytön ja tasapainottomuuden epäsymmetrisyyden toistuvuus (alkuperäiset arvot erosivat kuvan 13 arvojen kanssa). Muitakaan ongelmia ei ohitettu vaan kaikki ongelmat otettiin yhtä vakavasti. Sivujen tasapainottomuus ja epäsymmetrisyys tarkoitti järjestelmän ulkoasun sommittelua eli sitä kuinka miellyttävältä ja tasapainoiselta sivut näyttävät ja kuinka hyvin elementit on sijoitettu samalle tasolle pysty- ja vaakasuunnassa.

Merkittävät ongelmat

Kriittisimmistä ongelmista siirryttiin meidän mielestämme merkittäviin ongelmiin, jotka olivat sivujen hierarkian epäselvyys (käyttäjälle ei muodostu selvää kuvaa sivuston hierarkiasta eli siitä miten eri sivut ovat kytköksissä toisiinsa) sekä virheilmoitukset ja bugit. Vaikkakin virheilmoitukset saivat suhteellisen kovat arvot käyttöanalyysissa, niiden katsottiin kuitenkin kuuluvan vähemmän kriittisiin ongelmiin kuin epäloogisuudet ja tasapainottomuus. Virheilmoitukset ja bugit tarkoittavat käyttäjälle näkyviä virhetekstejä ja järjestelmän vääränlaista toimintaa. Käyttämässämme

järjestelmässä niitä oli todella harvoin, mikä oli erittäin hyvä asia, mutta jokainen bugi täytyi tietenkin ottaa tarkasteltavaksi, sillä ne heikentävät käyttäjän luottamusta järjestelmään. Pahimmassa tapauksessa bugit myös estävät käyttäjää käyttämästä järjestelmää.

Kohtalaiset ongelmat

Epäsopiva värinkäyttö on melko subjektiivinen käsite, sillä ei ole olemassa yhtä totuutta siitä, mitkä värit ovat sopivia ja mitkä epäsopivia. Tästä huolimatta väitimme, että käyttämässämme järjestelmässä käytetyt väriyhdistelmät olivat keskenään niin väriopillisesti kuin maallikonkin silmiin hieman huonosti valittuja, eivätkä ne olleet yhteensopivia. Vaikutus oli lähinnä visuaalinen, mikä heikensi järjestelmän mielekkyyttä. Oikein valittuina värit selkeyttävät käyttöä ja myös opastavat käyttäjiä. Järjestelmän toimintopainikkeilla voidaan viestiä, onko käyttäjä tekemässä jotain peruuttamatonta ja mahdollisesti haitallista tai tuhoavaa, mikä on syytä harkita uudelleen vai voiko nappia painaa sen suuremmin miettimättä. Väreillä voidaan myös ryhmitellä kokonaisuuksia helpommin ymmärrettäviksi ja lisäksi huolella valittu väripaletti rakentaa yhtenäistä ”brändiä”.

Lomakkeiden validointi tarkoittaa lomakkeissa käytettäviä opasteita ja merkintöjä siitä, mitkä kentät käyttäjän on pakko täyttää, ja toisaalta myös palautemekanismia, joka kertoo tallentamisen epäonnistuttua, mitä puutteita käyttäjän täyttämässä lomakkeessa on. Varhaisimmillaan lomakkeiden validointi oli useassa kohdassa puutteellista ja järjestelmä antoi käyttäjän tallentaa lomakkeen, vaikka pakollisen kentän sisältö oli tyhjä. Tämä huomattiin onneksi ajoissa ja saatiin nopeasti korjattua toimeksiantajan ripeyden vuoksi.

Vähäiset ongelmat

Vähäisimmiksi luokitellut ongelmat ja niiden painokertoimet jäivät luomassamme prioriteettitaulukossa alhaisiksi, mutta niiden merkitystä käyttäjäkokemukseen ei voi silti aliarvioida. Vaikeaselkoiset kentät, selitteet tai kirjoitusvirheet luovat kuvaa huolimattomasti tehdystä järjestelmästä ja syövät järjestelmän luotettavuutta. On varmaa, että yhdenkään käyttäjän työskentely ei keskeydy näiden ongelmien takia, mutta vii-

meistelyyn ja laatuvaikutelmaan näillä on huomattava merkitys. Käyttäjälle saattaa jäädä yksittäisen kentän, toiminnon tai koko sivun funktio epäselväksi. Osa asioista voidaan selvittää kokeilemalla, mutta kaikkia asioita käyttäjä ei uskalla kokeilla.

Sivujen alkuun voidaan lisätä lyhyt kuvausteksti, joka kertoo, mihin tämä näkymä on tarkoitettu. Kokeneet käyttäjät voivat kytkeä sivuohjeet pois käytöstä. Yksittäisten toimintojen tai tietokenttien yhteyteen voidaan lisätä lyhyitä ohjeita esimerkiksi tool-tip -tyylisenä, jolloin ne eivät häiritse käyttäjää ja näkyvät vain tarvittaessa.

Järjestelmässä voi olla linkkejä, tekstinpätkiä tai termejä, jotka eivät avaudu maallikolle eivätkä välttämättä edes kokeneemmalle käyttäjälle. Linkkien nimeämisessä kannattaa käyttää nimiä, jotka ovat yleisesti käytössä, jos tämä vain on mahdollista. Vaikeaselkoisiin nimiin ja kohtiin voidaan laittaa aputeksti, jolla tarkoitus avautuu käyttäjälle.

10 TOIMINNAN MALLINTAMINEN UML-LUOKKAKAAVIOILLA

Käyttöjärjestelmä tuntui aluksi monimutkaisesti rakennetulta, eli sen alivalikkojen ja komentopainikkeiden takaa avautuvien uusien sivujen taltiointi oli työlästä. Polkuluonnokset piirrettiin ensin paperille ja niitä hiottiin useaan otteeseen. Tämän jälkeen luonnokset muutettiin UML –luokkakaavioiksi.

Tietojärjestelmän laajuuden vuoksi jouduimme jakamaan UML -luokkakaavio -dokumentoinnin. Kumpikin dokumentaatiota tekevä henkilö valitsi oman luonnosteluohjelmansa ja kaavioiden piirtäminen saatiin näin vauhtiin. Tarkoituksena oli saada 11.4.2012 mennessä dokumentointi alustavasti palautettavaan kuntoon. Tehtävä ei ollut helppo kaavioiden laajuuden vuoksi. Vaikka kaavioita yritettiin saada mahtumaan pienempään tilaan, kaavioista tuli tästä huolimatta valtavia usean sivun kokoisia luonnoksia, joista saimmekin huomautusta. Kaaviot täytyi saada luettavaan kuntoon, eivätkä ne saaneet näyttää verkostoilta. Ainoaksi vaihtoehdoksi jäi ottaa omia vapauksia, jotka hyväksytettiin myös toimeksiantajalla. Tietosuojan vuoksi luonnoksia ei voida julkaista.

Järjestelmän purkaminen mahdollisimman pieniin osiin oli kaikkein kannattavinta. Jatkokehittäminen ei koske enää meidän työkuvaamme, joten uusien järjestelmän parissa toimivien henkilöiden tulisi pystyä käyttämään järjestelmää mahdollisimman tehokkaasti ja löytää sieltä tietoa.

Projekti eteni mallikkaasti huhtikuun ajan ja saimme projektin valmiiksi jo kuukausi ennen määräaikaa. Projektin lopetusajankohdaksi oli kaavailtu toukokuun puoltaväliä. Dokumentaatio saatiin koottua kaiken muun materiaalin kanssa jo 14.4.2012. Projekti katsottiin onnistuneeksi ja loppuraportit viimeisteltiin palautuskuntoon.

11 JATKOKEHITTÄMINEN

Ryhmämme sai tiedon huhtikuun lopulla, että firma, jolle teimme tietojärjestelmän käytettävyystutkimusta, oli myynyt ohjelmistoliiketoimintansa. Tietojärjestelmä oli toteutettu erinomaisesti, joten firman siirtyminen osaksi suurempaa tietotaloa ei tullut yllätyksenä kenellekään. Näin ollen oikeudet ja osa työntekijöistä siirtyi muualle. Tämä ei vaikuttanut projektimme jatkumiseen mitenkään. Tiedostimme oman ryhmämme onnistuneen, sillä oma panoksemme käytettävän järjestelmän luonnostelusta ja dokumentoinnista edesauttaa järjestelmän kehitystä yhä pidemmälle. Projekti antoi hyvän näkökulman ohjelmistojen tuotannosta, laatutekijöiden merkityksestä yrityksille ja yritysten tarpeista tietotekniikan alalla. Projekti saatiin päätökseen mallikkaasti ja työ oli kaikin puolin onnistunutta. Henkilökohtaisesti työmäärä tuntui aluksi valtavalta, sillä käyttämämme tietojärjestelmä oli jo tuotantokäytössä. Ylläpitäjän oikeuksilla järjestelmän testaaminen oli kuitenkin palkitsevaa ja samaan aikaan stressaavaa. Valmiin tuotteen purkaminen osiksi osoitti näyttävästi, miten paljon työtunteja järjestelmään on kulunut.

On sanomattakin selvää, että järjestelmä tuskin koskaan tulee olemaan täysin valmis. Ideat kehittyvät ja toteutusmahdollisuuksia tulee koko ajan lisää. Ryhmämme keskuudessa heräsi kysymys ja mielenkiinto mahdollisesta mobiilituesta järjestelmässä, josta tehtiinkin alustavat määrittelyt ja mobiilinäkymien suunnittelu aloitettiin.

LÄHTEET

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Helsinki: TALENTUM

Azuma, M. & Scalet, D. 2004. Software engineering – Software product Quality Requirements and Evaluation (SQuaRe) – Guide to SQuaRE

ISO/IEC FDIS25010. 2010. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models

Nevalainen, R. 2009. ISO/IEC 25000 sarja (SQUARE): IT-standardit liiketoiminnan jatkuvuuden takeena. Viitattu 15.3.2012 http://www.sfsedu.fi/www/fi/liitetiedostot/SFS/SC7_ISO_IEC_25000_SQUARE_esittely.pdf Haettu 15.3.2012

ISO/IEC CD25012. 2006. Software engineering: Software Quality Requirements and Evaluation (SQuaRE) - Data Quality Model.

ISO/IEC FCD 25030. 2005. Software engineering - Software quality requirements and evaluation (SQuaRE) - Quality requirements

Harsu, M. 2011. OHJ-3100 Ohjelmien ylläpito ja evoluutio. Viitattu 16.4.2012 http://www.cs.tut.fi/~evo/kalvot/takmalli2.pdf Haettu 16.4.2012.