Prosessien ohjaamana…

Tässä alkupuheenvuorossa tarkastellaan ohjelmistotekniikan yleisen kehityksen erästä osa-aluetta (= ohjelmistoprosessia) pyrkien assosioimaan ohjelmistotekniikan viitekehyksessä esiintyvää käsitteistöä myös muihin konteksteihin. Aloitetaan määritelmällä: *IEEE Standard Glossary for Software Engineering Terminology*:n mukaan

> Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Vuonna 1968 NATO:n organisoimassa konferenssissa todettiin ohjelmistotekniikan olevan kriisissä. Laajojen ohjelmistojen tuottaminen oli vaikeaa, ne olivat pullollaan virheitä, niiden toimitusajat venyivät ja valmistuskustannukset karkasivat käsistä. Seuraavina vuosikymmeninä tilannetta pyrittin parantamaan mm. rakenteisen ohjelmoinnin, CASE (Computer Aided Software/Systems Engineering) -työkalujen ja olio-paradigman avulla. Kuitenkin esimerkiksi Tiede 2000 -lehden numerossa 7/2002 julkaistun käännösartikkelin, jonka alkuperänä oli Technology Review -lehden heinä-elokuun numero, otsikko kuului “Onneton ohjelmointi vie rahat ja hermot”. Ohjelmistotuotannon ongelmat ovat siis edelleen olemassa, ja tehdyt virheet saattavat olla jopa hengenvaarallisia tai ainakin aiheuttaa suuria taloudellisia menetyksiä, kuten esimerkiksi Ariane V -kantoraketin itsetuhoutumisen mitättömän ohjelmointi- ja versionhallintavirheen seurauksena.

Nykypäivänä tietokoneohjelmien tekeminen on helppoa. Jo ala-asteikäiset niitä toteuttavat ja erilaisia sovelluskehittimiä ja “velhoja” (engl. *wizard*) käyttäen komealta näyttävä graafinen käyttöliittymä syntyy muutamassa minuutissa. Mikä siis edelleen mättää?

Yleensä katsotaan, että vuonna 1970 ilmestyi (pohjautuen tosin jo 1950-luvulla tehtyyn työhön) ohjelmistotekniikan ensimmäinen prosessimalli, ns. vesiputousmalli. Malli kuvasi ensimmäistä kertaa ne vaiheet, määrittelyn, suunnittelun, toteutuksen, testauksen, integroinnin ja ylläpidon, jotka läpikäymällä voitaisiin rakentaa laadukkaita ja toimivia ohjelmia. Varsinainen *ohjelmistoprojekti* on vain prosessin eräs ilmentymä, jossa vaiheet käydään läpi yhden sovellusohjelman tuottamiseksi. Niin vesiputousmalliin kuin jokaiseen muuhunkin vaihemalliin liittyvät keskeisinä käsitteet *validointi* ja *verifiointi*. Validoinnilla tarkoitetaan sitä, että jokaisen rakennusvaiheen yhteydessä pyritään varmistumaan siitä, että toteutettavana on ja pysyy sama ohjelma (“Are we building the right software?”). Verifioinnilla puolestaan tarkoitetaan sitä, että kyseinen vaihe itsessään on oikein toteutettu (“Are we building the software right?”).

Vesiputousmallin lähtökohtana on, että prosessi kuljetaan kerran läpi alusta loppuun. Pian kuitenkin huomattiin, että käytännössä tämä ei toimi: järjestyksessä edeltäviä vaiheita ei yleensä pystytä toteuttamaan täydellisesti kerralla, vaan vaiheita ja siten koko prosessia täytyy käydä läpi useamman kerran. Kuuluisin tällainen ns. *iteratiivinen* prosessimalli on Barry Boehmin 1988 esittelemä spiraalimalli, jossa tavoitteiden ja rajoitteiden, vaihtoehtojen ja riskien, kehityksen ja testauksen sekä suunnittelun vaiheita toistetaan *inkrementaalisesti* useamman kierroksen ajan. Tällä hetkellä mielestäni paras ehdokas minkä tahansa iteratiivisen ja inkrementaalisen prosessimallin *toiminnallis-rakenteelliseksi ajuriksi* ovat Ivar Jacobsonin 1967 alkaen kehittämät käyttötapaukset, joiden avulla voidaan kuvata ohjelman toimintaa sen tulevan käyttäjän näkökulmasta ilman teknisiä yksityiskohtia.

Vuonna 1987 Frederick Brooks julkaisi kuuluisan artikkelinsa otsikolla “No silver bullet: Essence and accidents of software engineering”, jonka keskeisenä viestinä oli, että yhdellä yksittäisellä tekniikalla tai menetelmällä ei koskaan voida taata kaikissa tapauksissa kaikilta osiltaan onnistunutta ohjelmistotuotantoa. Ohjelmistoalalla hyvin yleinen näkemys kuitenkin on, että niin hyvin määritellyt prosessimallit kuin ns. formaalien menetelmien (eli sen, kuinka yksikäsitteisesti ja tarkasti ohjelmaa

kuvataan sen eri kehitysvaiheissa) asiantunteva ja valikoiva soveltaminen ovat parhaita ehdokkaita hopealuotien valumuoteiksi.

Aloitetaanpa sitten jo esitettyjen asioiden miettiminen muissa konteksteissa, aluksi vaikkapa urheiluvalmennukseen liittyen. Niin kaksikymmentä vuotta sitten kuin nykyisinkin urheilija pyrkii rakentamaan huippukunnon esimerkiksi peruskuntokauden I, peruskuntokauden II, lajivoimakauden, tekniikkakauden, kilpailuunvalmistautumiskauden ja varsinaisen kilpailukauden kautta. Vaiheiden verifiointi eli vaikkapa tarkkojen lääketieteellisten testien käyttäminen urheilijan ominaisuuksien kehityksen jatkuvaan seurantaan yksittäisen vaiheen aikana herättää edelleen paljon keskustelua “akateemisten valmentajien” ja “vanhan kansan” valmentajien kesken. Niin suurelle yleisölle kuin itse urheilijallekin on kuitenkin paljon tärkeämpää valmennusprosessin validoituminen eli urheilijan kokonaiskehityksen edistyminen ja siten menestymismahdollisuuksien paraneminen. Iteratiiviseksi tilanne muuttuu esimerkiksi silloin, kun kilpailukunto puristetaan esiin niin hallikaudelle kuin kesäkaudelle. Melko monet ovat jopa sitä mieltä, että spiraalimallin mukaisesti yksilöurheilijankin tulisi olla lähellä kilpailukuntoa myös eri harjoittelukausien aikana.

Myös (ohjelmistotekniikankin, tottakai) oppiminen voidaan nähdä prosessina, joka koostuu esimerkiksi motivoitumis-, orientoitumis-, sisäistämis-, soveltamis-, arviointi- ja kontrollivaiheista. Mitä paremmin yksittäinen oppija löytää itselleen parhaan oppimisprosessimallin - oppii oppimaan - sitä paremmat mahdollisuudet hänellä on kasvattaa osaamispääomaansa inkrementaalisesti *metakognitiivisten* taitojen kehittyessä. Kun sitten tulee opinnäytetyön kirjoittamisen aika, voi tätä lähestyä joko vesiputousmallisesti käyttämällä useita vuosia niin laajan aineiston keräämiseen, että nalli on kastunut ja tarvitaan graduhautomoa, tai tieteellisen kirjoittamisen spiraalimallin mukaisesti aiheen valinta, metodin pohdinta, kirjallisuuden lukeminen, aineiston keruu, aineiston analyysi ja kirjoittaminen -vaiheita iteroimalla.

Kaikki eri alojen ja kontekstien vaihejakomallit voidaan itse asiassa nähdä yhden ja saman metamallin - *ongelmanratkaisuprosessin* - ilmentyminä. Ongelmanratkaisuprosessin yleistä jakautumista eri osavaiheisiin voidaan tarkastella esimerkiksi seuraavasti:
1. Mikä ongelma halutaan ratkaista? (vaatimukset)
2. Miten ratkaistava ongelma jakaantuu osaongelmiin? (suunnittelu)
3. Miten osaongelmat ratkaistaan? (totetus)
4. Mikä saadaan alkuperäisen ongelman ratkaisuksi?

Kehitysvaiheiden "hyvyyttä" ratkaisua kohti ponnisteltaessa voidaan tarkastella vaikkapa seuraavien kysymyksien avulla:
1. Onko ratkaistava ongelma kuvattu ymmärrettävällä, yksikäsitteisellä ja ristiriidattomalla tavalla käyttäen ongelma-alueen terminologiaa?
2. Päteekö sama osaongelmiin ja niiden välisiin riippuvuuksiin?
3. Miten kukin osaongelma ratkaistaan tavalla, joka on varmasti toimiva ja luvallinen tehtyjen oletuksien suhteen?

Tarkasteltaessa prosessin uudelleenkäyttöä on selvää, että mitä varhaisemmasta vaiheesta ketjua pystytään hyödyntämään uudelleen sen parempi. Parhaimmillaan tämä voi tapahtua jopa rakenteisista ajureista alkaen.

Ongelmanratkaisuprosessi eri variantteineen ilmenee yliopistokontekstissa erityisesti tutkimuksen tekemisen yhteydessä. Esimerkiksi tieteellisen laskennan perinteinen tutkimusskeema fysikaalisen ilmiön hallitsemiseksi koostuu matemaattisen mallin, laskennallisen mallin, mallin numeerisen ratkaisumenetelmän ja sen soveltamisen sekä ilmiön ohjaamisen vaiheista. Kaikkien vaiheiden, niin ilmiön irrottamisen fysikaalisesta kontekstista kuin ratkaisumenetelmien kehittämisen, täytyy liittyä saumattomasti toisiinsa, validoitua, jotta ketjua voidaan turvallisin mielin soveltaa esimerkiksi “oikeisiin’ teollisuusprosesseihin. Eri osavaiheiden verifioinnissa käytetään vahvasti matemaattisia menetelmiä, koska niiden avulla tarkasteltavia vaihetuotteita voidaan kuvata yksikäsitteisesti.

Yleistasolla tarkasteltuna tutkimuksen vesiputousmallinen ongelmanratkaisuprosessi koostuu $i)$ tausta; mallit ja teoriat $ii)$ tiedon hankkiminen $iii)$ tiedon analysointi ja $iv)$ johtopäätösten tekeminen -vaiheista. Iteratiivisuus seuraa luonnollisesti siitä, kun samaa prosessia sovelletaan useisiin tutkimusongelmiin, joiden taustaan tietämyksen kumuloituminen vaikuttaa. Toisinaan osaongelmien ratkaisemiseen sovelletaan sellaisia menetelmiä, jotka ovat tutkijalle niin vaativia (mutta kuulostavat hyviltä...), että niiden sisältämien pohjaoletusten tunnistaminen ja validointi varsinaisiin tutkimusongelmiin on lähes mahdotonta - analogisesti hienon graafisen käyttöliittymän ja sen varsinaisesti tarjoaman toiminnallisuuden kanssa. Lisäksi, jos tutkimusalueen kuvaamiseen käytetty käsitteistö ei ole yksiselitteinen, ei meillä ole mahdollisuutta myöskään tarkastella tehdyn työn validointia - formaalien menetelmien metaforan mukaisesti.

Ongelmia ratkaisevat ihmiset ja ongelmaratkaisuprosessia voidaankin tarkastella myös siihen osallistuvan ryhmän kautta. Luulenpa, että alkuperäisen ongelman ratkaiseminen tai ainakin tarkastelu osaongelmien kautta voi olla korkeintaan niin laadukasta, kuin eri osaongelmien ratkaisijoiden kyvyt edellyttävät, sillä prosessin validointi tapahtuu minimiperiaatteen mukaisesti. Tämä näkemys linjaa myös eri rooleissa olevien toimijoiden välistä yhteistyötä ohjelmistoprojektin suhteen: osaaminen omalla osa-alueella asettaa rajat saavutuksille myös koko projektin osalta. Tämä ei kuitenkaan sellaisenaan riitä, sillä projektin onnistumisen erästä perusedellytystä voidaan tarkastella myös *proaktiivisen johtamisen* näkökulmasta: Kenellä on vastuu projektin etenemisestä? Mistä löytyvät ne projektipäälliköt, jotka pystyvät integroimaan V&V:hen talous-, henkilöstö- ja aikatauluhallinnan riittävällä mutta joustavalla tavalla?