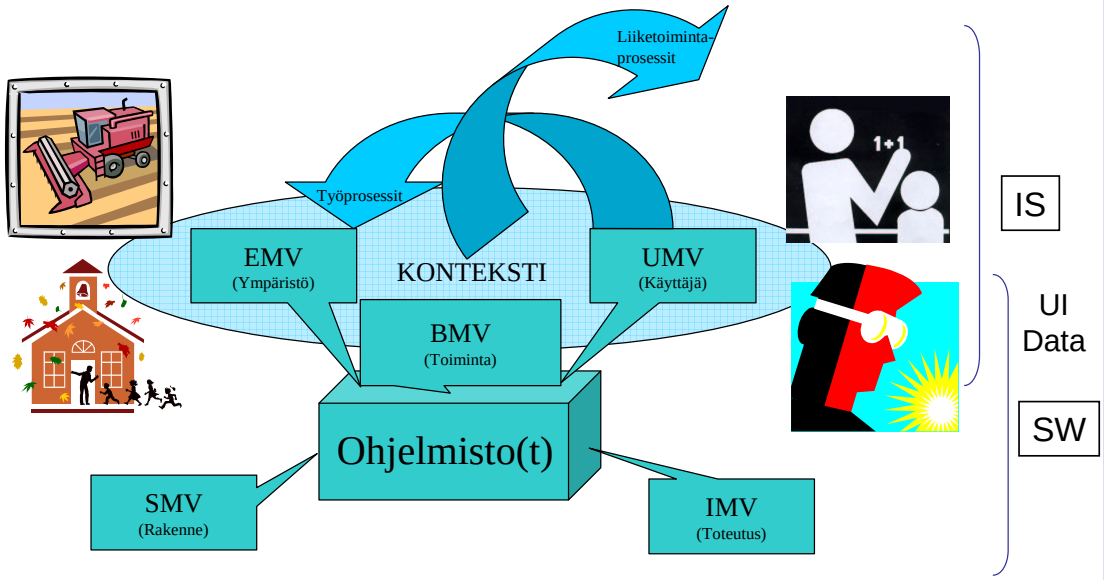
Luennot 2. ja 3: lisä- ja yhteenvetomateriaalia

Tommi Kärkkäinen
Jyväskylän yliopisto
tietotekniikan laitos

- Mikä ohjelmistokehityksessä on vaikeinta? Miksi?

  - "The hardest single part of building a software system is deciding precisely what to build." (Brooks: "No Silver Bullet", 1987)
    - Tässä ei puhuta vaatimuksista tai niiden dokumentoinnista vaan päättämisestä!
      - Vaatimusmääritykset välittävät tietoa asiakkaiden ja kehittäjien välillä ja näin tukevat päätöksentekoa & dokumentoivat tehtyjä päätöksiä
    - Tässä ei puhuta siitä, milloin päätöksiä tehdään
      - Kun ohjelma kehittyy ja sitä pääsee vähän testailemaan, konteksti ja usein myös mieli muuttuu sen suhteen mitä sitä oikein haluttiinkaan…
        - iteratiivinen ja inkrementaalinen kehitys rulez!

Onnistunut vaatimusmäärittely: oikeat ohjelmistot oikeille käyttäjille oikeisiin ympäristöihin!



 (muok. Pressman, "Software Engineering – Practitioner's Approach", 2000. Alkup. idea Äyrämön Samin!)

Reqs NOK: Kaaoshan siitä saattaa syntyä

Project Success Factors	% of Responses
1. User Involvement	15.9%
2. Executive Management Support	13.9%
3. Clear Statement of Requirements	13.0%
4. Proper Planning	9.6%
5. Realistic Expectations	8.2%
6. Smaller Project Milestones	7.7%
7. Competent Staff	7.2%
8. Ownership	5.3%
9. Clear Vision & Objectives	2.9%
10. Hard-Working, Focused Staff	2.4%
Other	13.9%

1995

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

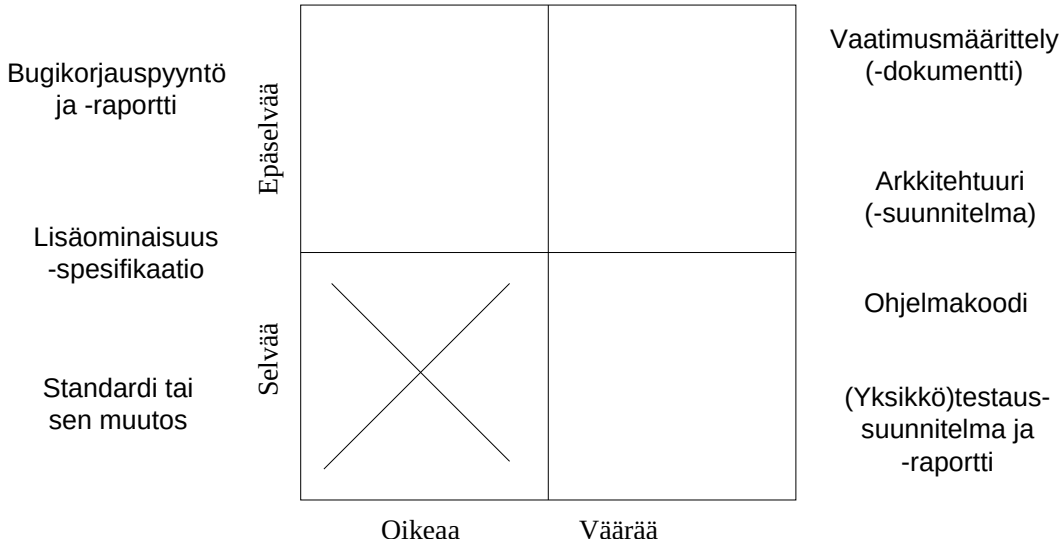
CHAOS Ten

User Involvement	20 Points
Executive Support	15 Points
Clear Business Objectives	15 Points
Experienced Project Manager	15 Points
Small Milestones	10 Points
Firm Basic Requirements	5 Points
Competent Staff	5 Points
Proper Planning	5 Points
Ownership	5 Points
Other	5 Points

1999

 IT executive manager nonacademic surveys, i.e. Chaos reports, by The Standish Group

Tuotosten sisällöstä: *laatu ~ 1/määrä*

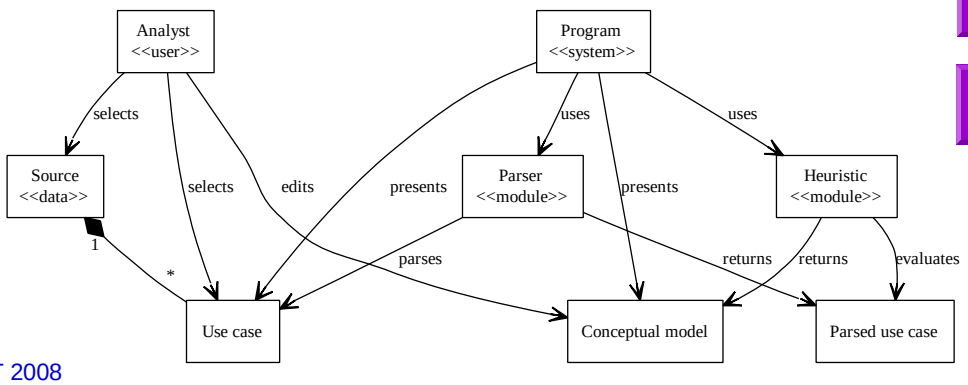


Käyttötapauksista käsitemalliin

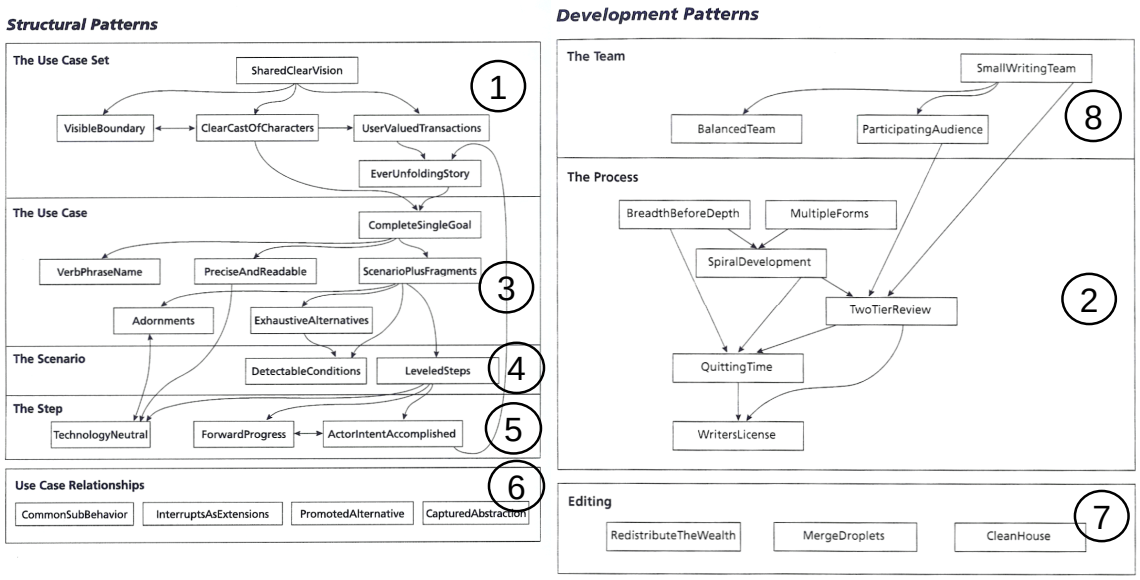
[name]
2 Create Conceptual Model (ei Main flow!)
3 [id]
4 1
5 [steps]
6 User selects the use case. (2)
7 Program processes the use case. (3)
8 Program shows the conceptual model.
9 User edits conceptual model.
10 Program stores the conceptual model.
11 [end]
12
13 [name]
14 Select use case
15 [id]
16 2

17 [steps]
18 User selects the source of the use cases.
19 Program presents the list of the use cases contained by the source.
20 User selects use case from the list of use cases.
21 [end]
22
23 [name]
24 Process use case
25 [id]
26 3
27 [steps]
28 Program passes the use case to the parser.
29 Parser returns the parsed use case.
30 Program passes the parsed use case to the heuristic.
31 Heuristic returns the conceptual model.
32 [end]

(s. 11)
(s. 12)
(s. 13)
(s. 15)
(s. 16)



Malliperheet



Mallien luokittelu (ja toteutusvaiheanalogia)

1. **Miltä laadukas tuotos (esim. käyttötapausjoukko joka kuvaa SuD:n (engl. System-under-Development) toiminnallisia vaatimuksia) näyttää?**
2. **Miten kehitysvaiheen työt tulisi organisoida, jotta laadukas tuotos saadaan aikaan?**

1. **Rakenne: Käyttötapausjoukko:** Yleisimmät ja (onnistumisen kannalta) tärkeimmät mallit yhteenveto-tasolla (Cockburn), jotka kuvaavat vaihetuotosta kokonaisuudessaan.
2. **Kehitys: Prosessi:** Tärkeimmät mallit yhteenveto-tasolla, jotka kuvaavat vaihetuotoksen tuottamisen ydinaktiviteetteja.
3. **Rakenne: Käyttötapaus:** Mistä käyttötapaus koostuu? Miltä se näyttää?
4. **Rakenne: Skenaario:** Miltä pääskenaario näyttää?
5. **Rakenne: Askel:** Miltä skenaarion yksittäinen askel näyttää?
6. **Rakenne: Käyttötapaussuhteet:** Minkätyyppistä uudelleenmuokkausta käyttötapausjoukkoon voidaan soveltaa?
7. **Kehitys: Muokkaus:** Kuinka uudelleenmuokkaus organisoidaan?
8. **Kehitys: Tiimi:** Kuinka työ yleisesti organisoidaan?

(Huom: erilainen kuin monisteessa)

1. Arkkitehtuuri?
2. Toteutuksen, testauksen, integroinnin, versionhallinnan ym. organisointi?
3. Luokat/funktiot/aliohjelmat ja rajapinnat?
4. Tietorakenteet ja algoritmit?
5. Muuttujat?
6. Refaktorointi?
7. Muutosten jälkeisen eheyden varmistamisen organisointi?
8. Kuka tekee ja mitä?

Käyttötapausjoukko: **JaettuSelkeäVisio**

- kehitettävästä sovelluksesta tarvitaan selkeä näkemys

Ratkaisu: *Valmistele järjestelmästä tarkoituslausuma, joka kuvailee selkeästi järjestelmän päämäärät ja tukee organisaation tehtävää. Jaa sitä vapaasti kaikille projektin osallisille.*

Esimerkki: **http://sovellusprojektit.it.jyu.fi/ucot/projekti/files/UCOT-projektisuunnitelma-0.04.pdf, luku 1.1**

Käyttötapausjoukko: **NäkyväRaja**

- kehitettävä sovellus on osa olemassa olevaa teknis-toiminnallista ympäristöä

Ratkaisu: *Perusta näkyvä raja järjestelmän ja ympäristön välille nimeämällä laitteistot ja toimijat, jotka ovat järjestelmän kanssa vuorovaikutuksessa.*

Esimerkki: **http://www.vtt.fi/inf/pdf/tiedotteet/2008/T2442.pdf**, luku 7.3.1 ISs

Käyttötapausjoukko: **SelkeäRoolijako**

- kehitettävää sovellusta käytetään osana työtehtäviä

Ratkaisu: *Jos analysoidaan vain järjestelmän käyttäjiä eikä heidän roolejaan käyttäjinä, voi jäädä huomaamatta järjestelmän tärkeää käyttäytymistä tai syntyä turhaa päällekkäistä käyttäytymistä.*

Esimerkki: Opiskelija-roolin prosessit, aktorit, dokumentit, tietojärjestelmät, … **http://prosessit.it.jyu.fi/julkaistut/ (esim. Prosessi 4.3.2)**

Käyttötapausjoukko: **KäyttäjälleArvokkaitaTapahtumia**

- kehitettävän sovelluksen on tarjottava käyttäjän tarvitsemaa toiminnallisuutta

Ratkaisu: *Tunnista hyödylliset palvelut, jotka järjestelmä välittää aktoreille tyydyttääkseen heidän (liike)toimintatarpeensa.*

Esimerkki: Luentomoniste: (Tietokanta-)CRUDeista palvelu(liike)toimintaan!

Käyttötapausjoukko: **JatkuvastiAvautuvaTarina**

- kehitettävän sovelluksen toiminnallisuutta voi tarkastella monilla tarkkuustasoilla

Ratkaisu: *Järjestä käyttötapausjoukko hierarkkiseksi tarinaksi, joka voidaan joko avata, jotta saadaan lisää yksityiskohtaisuutta, tai taittaa kokoon, jotta yksityiskohdat piiloutuvat ja asiayhteys näkyy paremmin.*

Esimerkki: IT-prosessit: Prosessitaulukkohierarkkia

Prosessi: **LaajuusEnnenSyvyyttä**

- kehitettävän sovelluksen kokonaistoiminnallisuus ohjaa yhteis- ja kehitystyötä

Ratkaisu: *Säästä energiaa kehittämällä ensin yleiskuva käyttötapauksista ja lisää sitten yksityiskohtia progressiivisesti/tarvittaessa, työskennellen toisiinsa liittyvien käyttötapausten parissa.*

Esimerkki: Kalvon 5 käyttötapauskuvaukset riittivät UCOT-projektissa

Prosessi: **SyklinenKehitys**

- näkemykset kehitettävän sovelluksen toiminnallisuudesta muuttuvat kehitystyön edetessä (käyttökontekstin muutos)

Ratkaisu: *Kehitä käyttötapaukset iteratiivisella laajuus-ensin tavalla, jolloin jokainen iteraatio parantaa käyttötapausjoukon tarkkuutta ja paikkaansapitävyyttä progressiivisesti.*

Esimerkki: UCOT-sovellus: Projektisuunnitelman tarkoituslausuma, olioanalyysilähtökohta ja päätyminen yleisempään sovellusaluekäsite-rakenteeseen.

Prosessi: **MoniaLomakkeita**

- eri sovellukset vaativat erilaista vaatimusmäärittelylähestymistapaa

Ratkaisu: *Valitse formaatti, joka perustuu projektin riskeihin ja mukana olevien ihmisten mieltymyksiin.*

Esimerkki: http://alistair.cockburn.us/index.php/Basic_use_case_template

Prosessi: **KaksikerrosKatselmukset**

- kehitysprojektin osapuolia kiinnostaa omaan intressipiiriin liittyvien päätösten tekeminen

Ratkaisu: *Pidä kahdentyyppisiä katselmointeja: sisäisiä läpikäyntejä mahdollisesti monta kertaa, koko asianosaisryhmälle hyväksymisorientaatiolla vain kerran.*

Esimerkki: Luentomoniste: IT-prosessien työmuodot&-ryhmät; TTL/sovellusprojektit: projektiryhmän ja vastuullisen/teknisen ohjaajan sisäiset palaverit (face2face)

Prosessi: **AikaLopettaa**

- ”Projektin hallinta / käyttötapauksien kehittäminen yli asianosaisten ja kehittäjien tarpeiden hukkaa resursseja ja viivästyttää projektia.”

Ratkaisu: *Lopeta käyttötapausten kehittäminen, kun ne ovat valmiita ja täyttävät yleisön vaatimukset.*

Prosessi: **KirjoittajanLisenssi**

- ”Liiallinen tyylikysymysten painotus – koskien niin käyttötapauksia kuin muita projektin vaihetuotteita – haittaa tarpeettomasti projektin etenemistä.”

Ratkaisu: *Kirjoitustyyleissä on väistämättäkin pieniä eroja. Kun käyttötapaus läpäisee mallin* **AikaLopettaa** *testin, kirjoittaja voi vaatia ”kirjoittajan lisenssin” koskien pieniä tyylillisiä eroja.*

Esimerkki: Kalvon 5 käyttötapauskuvaukset, joita viilattiin useita tuntia kehittäjien ja asiakkaiden toimesta, riittivät UCOT-projektissa

Käyttötapaus: **SaavutaYksiTavoite**

- käyttötapauksen/kehitysvaiheen tavoite voi jäädä epämääräiseksi, jolloin kaikki muukin tämän jälkeen tehty on hataralla pohjalla (vrt. kalvo 5)

Ratkaisu: *Kirjoita yksi käyttötapaus kutakin hyvin määriteltyä tavoitetta kohti. Tavoite voi olla millä tahansa tasolla mallissa* **JatkuvastiAvautuvaTarina**.

Esimerkki: Cockburnin käyttötapauskirjoitusprosessin suositukset 3.-4.

Käyttötapaus: **VerbiFraasiNimenä**

- käyttötapauksen/artefaktin nimeämisen tulisi aktivoida lukijan relevantit (kognitiiviset) tietorakenteet

Ratkaisu: *Nimeä käyttötapaus aktiiviverbilauseella joka edustaa pääaktorin tavoitetta.*

Esimerkki: UCOT-UC: ”Main flow” → ”Create Conceptual Model”

Käyttötapaus: **SkenaarioJaTäydennykset**

- käyttötapauksen eri-bittisissä esityksissä on eroja – erityisesti luettavuudessa

Ratkaisu: *Kuvaa perustoiminta yksinkertaisena skenaariona harkitsematta epäonnistumista. Sijoita tämän alle täydennyksiä, jotka kertovat mitä eri vaihtoehtoja voi tapahtua ja miten ne hoidetaan kuntoon.*

Esimerkki: (lähes kaikki) IT-prosessit

Käyttötapaus: **EsitäVaihtoehdot**

- jos käyttäjä mokaa tai yhteydet bragaa niin pitää sopia miten jatketaan

Ratkaisu: *Etsi kaikki vaihtoehdot ja epäonnistumiset, jotka pitää käsitellä käyttötapauksessa.*

Esimerkki: IT-prosessit: 4.2.7.1.3 Opiskelupaikan vastaanottaminen

Käyttötapaus: **Täydennykset**

- toiminnallisuutta jäsennettäessä nousee esiin muitakin kiinnostavia tietoja

Ratkaisu: *Luo ylimääräisiä skenaariotekstin ulkopuolisia kenttiä käyttötapauspohjaan, jotka sisältävät täydentävää tietoa joka on hyödyllistä liittää käyttötapaukseen.*

Esimerkki: IT-prosessit: Rajoitteet-kenttä, esim. 1.2.2.4 Lomat

Käyttötapaus: **TarkkaJaLuettava**

- liian monimutkaiset tai yksityiskohtaiset interaktiokuvaukset ja/tai niiden kehitysorientoituneet esitysmuodot sekoittavat järjestelmärakennustyötä

Ratkaisu: *Kirjoita käyttötapauksesta tarpeeksi luettava, jotta asianosaiset vaivautuvat lukemaan ja arvioimaan sen, sekä tarpeeksi tarkka, jotta kehittäjät ymmärtävät mitä ovat rakentamassa.*

Esimerkki: IT-prosessit: 1.2.1.3.4 Virantäyttö/nimittämispäätös

Skenaario: **TodennettavatEhdot**

- mitä tiloja toteutus sisältää ja minkä prosessointien/interaktion avulla niiden välillä liikutaan

Ratkaisu: *Ota mukaan vain havaittavat tilat. Yhdistä tilat, joilla on sama nettovaikutus järjestelmän toimintaan.*

Esimerkki: Käyttötapauksen ja käsitemallin tilanmuutokset UCOT-sovelluksessa

Skenaario: **SamantasoisetAskeleet**

- lyhytkestoisella muistilla ja kognitiivisen kuorman kantavuudella on inhimilliset rajoitteensa

Ratkaisu: *Pidä skenaariossa kolmesta yhdeksään askelta (7 ± 2 –periaate). Ihanteellisesti askeleet ovat kaikki vastaavilla tasoilla ja abstraktiotasolla joka on juuri käyttötapauksen tavoitteen alapuolella.*

Esimerkki: IT-prosessit: 2.3 Atk-hankinnat

Askel: **KäyttäjänTavoiteSaavutettu**

- passiivit tai epäselvät toiminnallisuuskuvaukset sekoittavat flow:n

Ratkaisu: *Kirjoita jokainen askel näyttääksesi selkeästi, mikä aktori toimii ja mitä aktori saa aikaiseksi.*

Askel: **EteenpäinMenevästi**

- ”askeleiden on syytä svengata kuin hirvi – uimasilla”

Ratkaisu: *Eliminoi tai yhdistä askeleet jotka eivät edistä aktorin tavoitetta. Yksinkertaista kohtia jotka harhauttavat lukijaa.*

Esimerkki: UCOT-kuvaukset ja IT-prosessit 1.2.2.4.1.1: Vuosilomaesityksen …

Askel: **TeknologisestiNeutraali**

- teknologia- ja toteutusratkaisujen (sis. käyttöliittymävalinnat) sisällyttäminen kuvauksiin estää Separation-of-Concerns:n toteutumisen

Ratkaisu: *Kirjoita askel tavalla, joka on neutraali teknologian suhteen.*

Esimerkki: UCOT-kuvaukset (käsitemalli, jäsennin, …)

Käyttötapaussuhteet: **YhteinenOsaKäyttäytyminen**

- useasti toistuvia samoja askeljoukkoja/koodirivejä ei kannata hallita tuplasti

Ratkaisu: *Ilmaise jaetut toiminnot alempitasoisilla, sisällytetyillä käyttötapauksilla.*

Käyttötapaussuhteet: **KeskeytyksetLaajennuksina**

- useasti toistuva vaihtoehtoinen käyttäytyminen sirottelee yksityiskohtia ja aiheuttaa hämmennystä

Ratkaisu: *Luo laajennuskäyttötapaus kun vaihtoehtoinen toiminto keskeyttää usean askeleen skenaariossa.*

Esimerkki: Luentomoniste: Liite; UML-speksi, v2.1.1, luvut 16.3.5 ja 16.3.4:
http://www.omg.org/docs/formal/07-02-05.pdf

Käyttötapaussuhteet: **YlennettyVaihtoehto**

- jotkut (itse asiassa aika monet) ongelmat ovat vaikeita – niin kuvata kuin ratkaista – erityisesti lyhyesti ja ytimekkäästi

Ratkaisu: *Harkitse monimutkaisten, liikaa käyttötapausta hallitsevien vaihtoehtojen siirtämistä erilliseen käyttötapaukseen.*

Esimerkki: IT-prosessit: yhteenvetotason aliprosessilistat linkkeineen, esim. 4.2.7

Käyttötapaussuhteet: **KaapattuAbstraktio**

- kaksin aina kaunihimpi – paitsi käyttötapauskuvauksen etenemispolun ja tavoitteen osalta

Ratkaisu: *Harkitse abstraktin yleistetyn käyttötapauksen luomista. Laita jokainen erillinen vaihtoehtoskenaario, joka erikoistaa abstraktiota, omaan erikoistuneeseen käyttötapaukseensa.*

Esimerkki: Luentomoniste: Liite

Muokkaus: **JaaVaurausUudelleen**

- tarpeettoman pitkä käyttötapaus/koodipläjäys kaipaa muokkaavaa ylläpitoa

Ratkaisu: *Siirrä pitkä vaikeasti käsiteltävä osa tai liian monimutkainen laajennos omaan käyttötapaukseensa.*

Esimerkki: IT-prosessit: 4.2.7 Opiskelijavalintoihin liittyvät prosessit

Muokkaus: **YhdistäPalaset**

- tarpeettoman pieni käyttötapaus/koodipläjäys kaipaa muokkaavaa ylläpitoa

Ratkaisu: *Yhdistä toisiinsa liittyvät pikkuriikkiset käyttötapaukset tai käyttötapauksen katkelmat, joilla on sama tavoite.*

Esimerkki: Luentomoniste: Liite

Muokkaus: **TyhjennäTalo**

- ”lumihanki, poliisi ja viimeinen erhe” – eiku tarpeetonta toiminnallisuutta/koodia

Ratkaisu: *Poista ne käyttötapaukset, jotka eivät lisää arvoa järjestelmääsi tai jotka eivät ole aktiivisen kehityksen piirissä.*

Esimerkki: IT-prosessit: 4.2.7 Opiskelijavalintoihin liittyvät prosessit

Ryhmä: **PieniKirjoittajaRyhmä**

- liian monta silmää näkee jo liikaakin – erityisesti toiminnallisuutta ja toteutettavia ominaisuuksia

Ratkaisu: *Rajoita yhtä työtuotetta hiovien henkilöiden määrä kahteen tai kolmeen henkilöön. Käytä* **KaksikerrosKatselmukset***-mallia uusien ihmisten mukaan ottamiseen.*

Esimerkki: IT-prosessit: työmuodot&-ryhmät 1. ja 2. sekä mallintaja-henkilövaihdokset

Ryhmä: **OsallistuvaYleisö**

- osapuolia ei voi tyydyttää ilman heidän kuuntelemista ja tarpeiden tyydyttämistä

Ratkaisu: Ota asiakkaasi ja sisäiset asianosaiset aktiivisesti mukaan käyttötapausten kehitysprosessiin aina kun mahdollista.

Esimerkki: IT-prosessit: esittely tiedekuntakokouksessa, muokkausta varten:
Prosessialue 3: toimintaprosessien prosessit

Ryhmä: **TasapainoinenRyhmä**

- ohjelmistokehityksessä(kin) tarvitaan monenlaisia osaajia ja näkökulmia

Ratkaisu: *Aseta ryhmään eri erikoisosaamista omaavat henkilöt puolustaaksesi asianosaisten etuja kehitysprosessissa. Varmista, että ryhmässä on sekä suunnittelijoita että loppukäyttäjiä.*

Esimerkki: Ei suoraan balanssoidusta ryhmädynamiikasta, mutta jaetun käsitteistön (ontologian) tärkeydestä (Liite B, sivu 37):

http://sovellusprojektit.it.jyu.fi/ucot/projekti/files/UCOT-projektiraportti-1.00.pdf

Yhteenveto

Mallikasta käyttötapauskuvausta karakterisoivat ydinkriteerit ovat seuraavat:

- R = Käyttötapausjoukon yleisRakenne: Yleisen tason ((liike)toiminta)käyttötapauksen esittäminen ja sitä tarkentavien käyttötapausten ja näiden välisten suhteiden jäsentäminen
- M = Askelten Muotoilu: ”Kuka tekee mitä, mitä tavoitetta varten”
- S = Sovelluksen rooli interaktiosekvensseissä: Mitä käyttäjä-rooli tekee, mitä sovellus tekee, jotta käyttötapauksen tavoite saavutetaan.
- N = Käyttötapausten Nimeäminen: VerbiFraasiNimenä

Esimerkkejä:

- EI NÄIN: http://www.math-cs.gordon.edu/courses/cs211/ATMExample/UseCases.html
- SINNE PÄIN: http://www.mit.jyu.fi/opetus/kurssit/jot/2006/tentti/tokavast.txt
- http://www.mit.jyu.fi/opetus/kurssit/jot/2007/, Luentotehtävä&esimerkkivastaus
- http://www.vtt.fi/inf/pdf/tiedotteet/2008/T2442.pdf, luku 7

Harjoitustyön alkuja

Tehtävä: Esitä *Opiskelijan tulojen- ja menojenhallintasoftaan* liittyen yhteenveto-tason pääkäyttötapaus (Business Use Case) sekä tarkenna sen vähintään kahden askeleen toimintaa omilla käyttötapauksillaan.

Formaatin tulee olla seuraavan speksin mukainen:

[name]
<käyttötapauksen nimi>
[steps]
<Askeleet rivinvaihdolla eroteltuna.> {<Poikkeuksen id>} (<Viittaus askelta tarkentavaan käyttötapaukseen>)
[end]
<Poikkeuksen id>: <Poikkeuksen aiheuttaman toiminnan kuvaus>

[name]
<tarkentavan käyttötapauksen nimi>
...
[concepts]
<lista käsitteistä; valinnainen mutta hyödyllinen jatkon kannalta>

Kulmasulkeet <> jätetään pois; niillä osoitetaan vain yllä teksti, joka korvataan sisällöllä. Askeleeseen liittyvät poikkeukset ja viittaukset tarkennuksiin tulevat askeleen kanssa samalle riville.

The Writing Process

1. Name the system scope and boundaries.
   Track changes to this initial context diagram with the in/out list.
2. Brainstorm and list the primary actors.
   Find every human and non-human primary actor, over the life of the system.
3. Brainstorm and exhaustively list user goals for the system.
   The initial Actor-Goal List is now available.
4. Capture the outermost summary use cases to see who really cares.
   Check for an outermost use case for each primary actor.
5. Reconsider and revise the summary use cases. Add, subtract, or merge goals.
   Double-check for time-based triggers and other events at the system boundary.
6. Select one use case to expand.
   Consider writing a narrative to learn the material.
7. Capture stakeholders and interests, preconditions and guarantees.
   The system will ensure the preconditions and guarantee the interests.
8. Write the main success scenario (MSS).
   Use 3 to 9 steps to meet all interests and guarantees.
9. Brainstorm and exhaustively list the extension conditions.
   Include all that the system can detect and must handle.
10. Write the extension-handling steps.
    Each will end back in the MSS, at a separate success exit, or in failure.
11. Extract complex flows to sub use cases; merge trivial sub use cases.
    Extracting a sub use case is easy, but it adds cost to the project.
12. Readjust the set: add, subtract, merge, as needed.
    Check for readability, completeness, and meeting stakeholders' interests.