

Päivä 12 - Ohjelmistokehitys

Vaatimusmäärittely ja sovellusalueen mallinnus

2022-02-18

AaltoPRO - Websovelluskehitys

Päivä 12

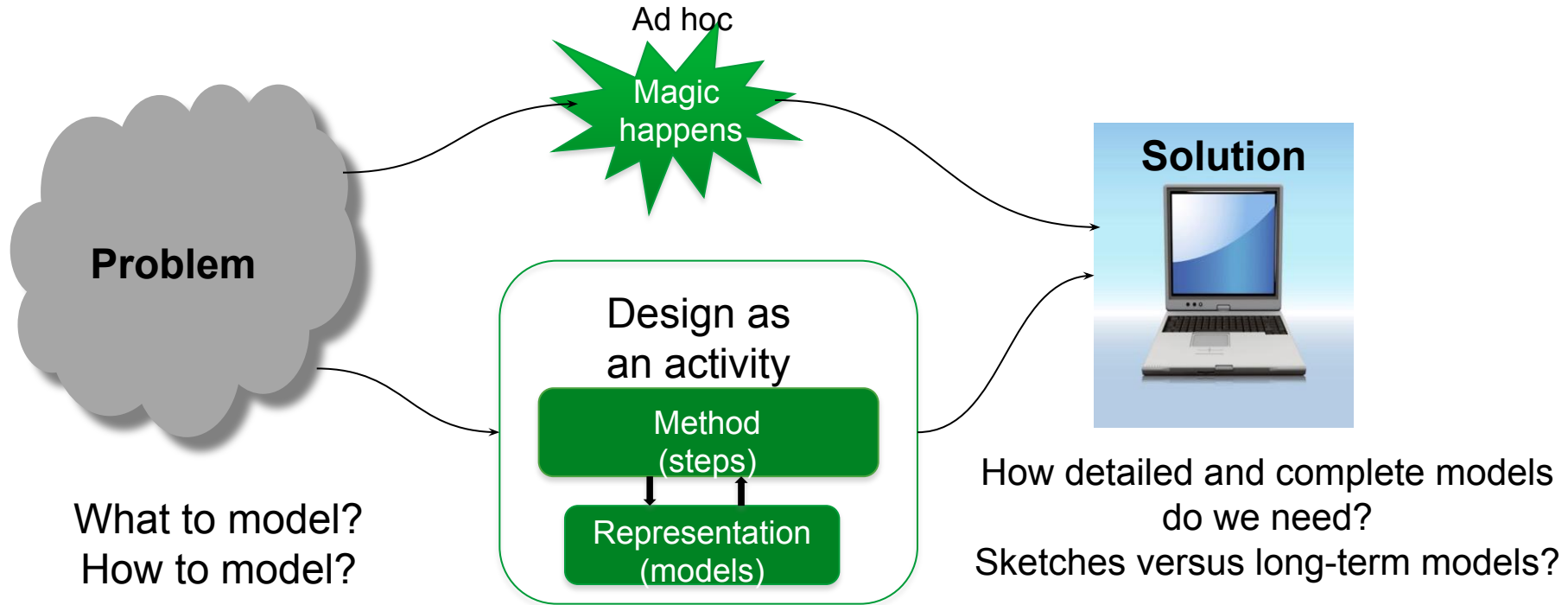
- Aamupäivä: ohjelmistokehitys yleisesti, vaatimusmäärittely
- Lounas
- Iltapäivä: sovellusalueen mallinnus

Ohjelmistokehitys yleisesti

Ohjelmistokehitys (software engineering)

- Yleisesti
 - Koodaus on vain osa ohjelmistokehitystä
 - Erilaisia lähestymistapoja ja käytäntöjä
 - Ohjelmistojen kehittäminen on tiimityöskentelyä

Ohjelmistosuunnittelu aktiviteettina



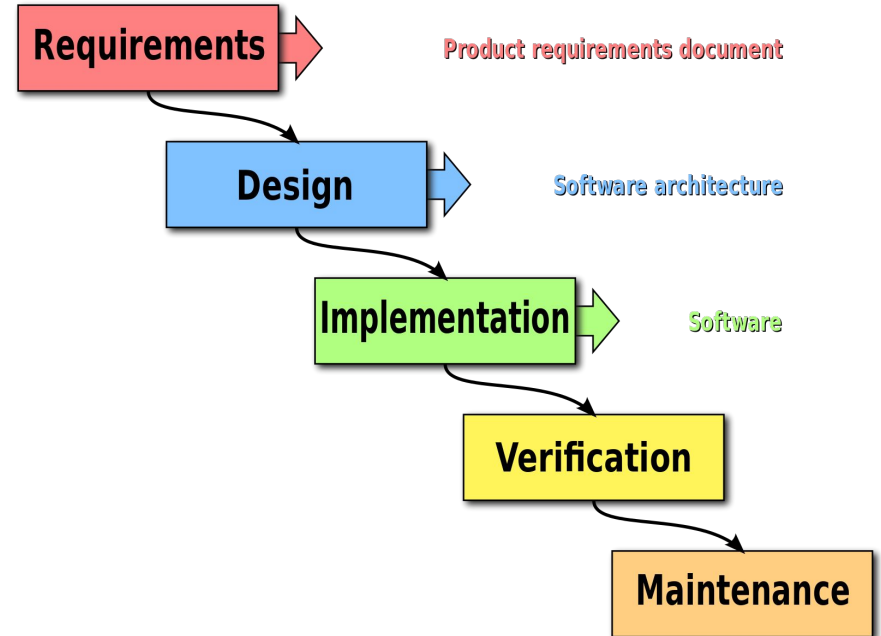
Vesiputousmalli

1. Järjestelmävaatimukset (System Requirements)
2. Ohjelmistovaatimukset (Software Requirements)
3. Analyysi (Analysis)
4. Suunnittelu (Program Design)
5. Ohjelmointi (Coding)
6. Testaus (Testing)
7. Käyttöönotto (Operations)

Toimiiko käytännössä?

Vesiputousmalli ja BDUF

- "Big Design Up Front"
 - Ensin suunnitellaan huolella, sitten vasta toteutetaan
- Argumentteja puolesta ja vastaan?



https://commons.wikimedia.org/wiki/File:Waterfall_model.svg

Ketterän ohjelmistokehityksen julistus (Agile Manifesto)

Löydämme parempia tapoja tehdä ohjelmistokehitystä, kun teemme sitä itse ja autamme muita siinä. Kokemuksemme perusteella arvostamme:

Yksilöitä ja kanssakäymistä enemmän kuin menetelmiä ja työkaluja
Toimivaa ohjelmistoa enemmän kuin kattavaa dokumentaatiota
Asiakasyhteistyötä enemmän kuin sopimusneuvotteluja
Vastaamista muutokseen enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta
arvostamme ensiksi mainittuja enemmän.

<https://agilemanifesto.org/iso/fi/manifesto.html>

Ketterä kehitys (agile)

1. Customer satisfaction by early and continuous delivery of valuable software.
2. Welcome changing requirements, even in late development.
3. Deliver working software frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the primary measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

<https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>

SAFE - The Scaled Agile Framework

#1 Take an economic view

#2 Apply systems thinking

#3 Assume variability; preserve options

#4 Build incrementally with fast, integrated learning cycles

#5 Base milestones on objective evaluation of working systems

#6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 Apply cadence, synchronize with cross-domain planning

#8 Unlock the intrinsic motivation of knowledge workers




#9 Decentralize decision-making

#10 Organize around value

Ketterään kehitykseen on monia erilaisia vaihtoehtoja

- https://en.wikipedia.org/wiki/Agile_software_development#Agile_software_development_practices

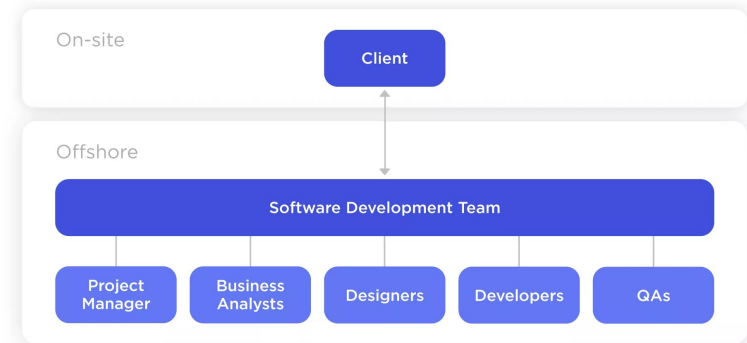
Tiimit ohjelmistokehityksessä

Product Manager <i>Drives the PI and product</i>	Product Owner <i>Drives the Iteration</i>	Agile Team <i>Drives program execution</i>
		
Owns Program Backlog	Owns Team Backlog(s)	Builds Quality-In, evolves Agile architecture
Defines Features, PIs, and Releases	Defines Iterations and Stories	Owns estimates
Owns Vision, Roadmap, pricing, licensing, ROI	Contributes to Vision, Roadmap, ROI	Evolves the Continuous Delivery Pipeline
Collaborates on Enablers	Accepts Iteration increments	
<i>Build the right thing...</i>		<i>...Build the right way</i>

© Scaled Agile, Inc.

RELEVANT

TYPICAL SOFTWARE DEVELOPMENT TEAM STRUCTURE

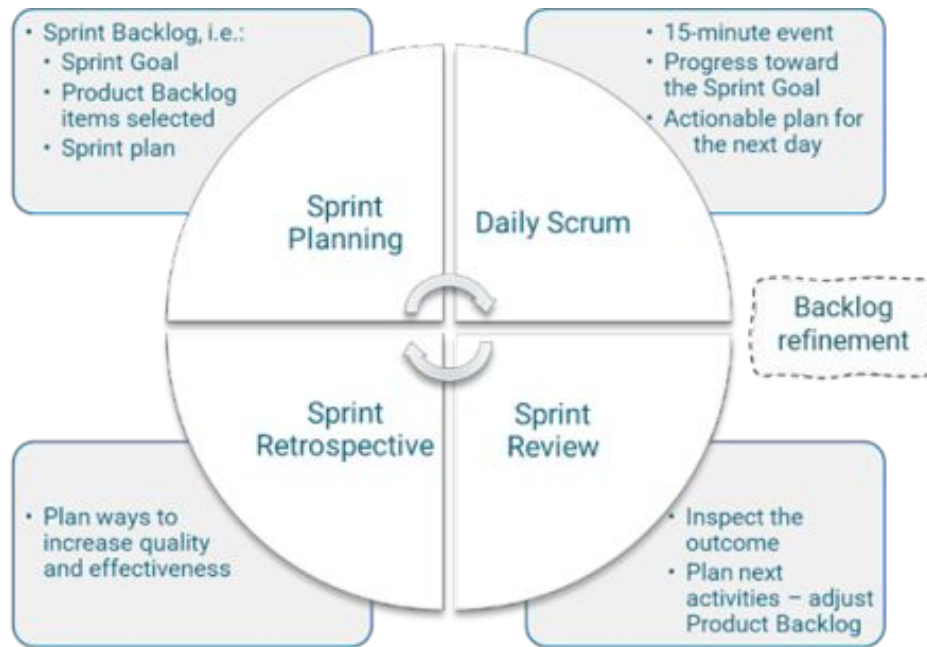


<https://www.scaledagileframework.com/product-owner/>

<https://relevant.software/blog/what-agile-software-development-team-structure-looks-like/>

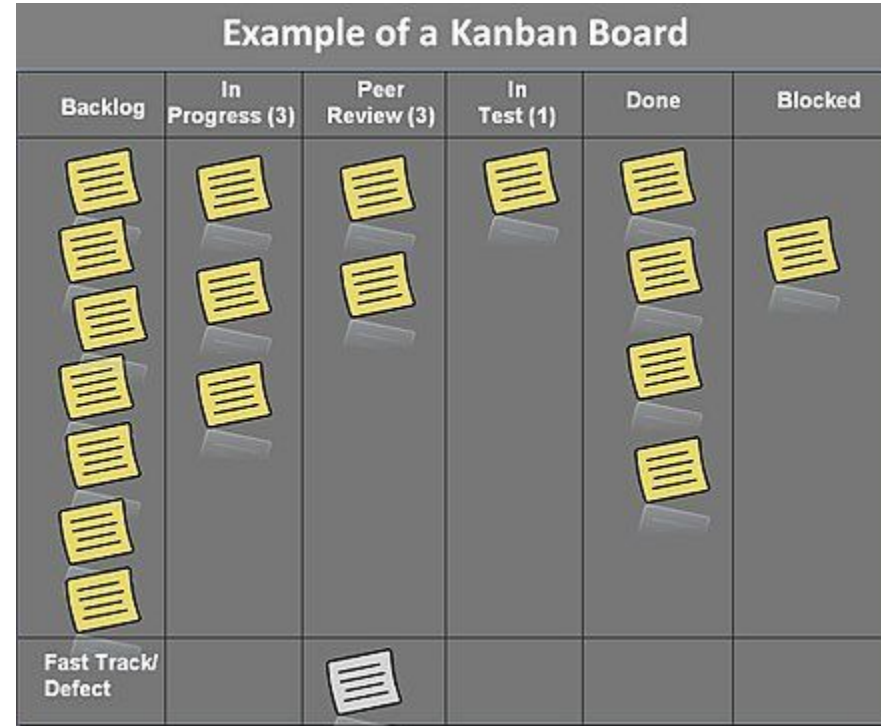
Scrum

- Sprint: ~2-4 viikon mittainen kehityssykli
- Henkilöitä:
 - Product owner (PO)
 - Omistaa product backlogin
 - Priorisoi backlogin
 - ScrumMaster
 - Kehitystiimi
- Sisältää:
 - Product backlog (käyttäjätarinat)
 - Sprint backlog (tärkeimmät käyttäjätarinat, tehdään tässä sprintissä)
 - Burndown chart (tehtävät tässä sprintissä)
- Tapaamisia
 - Sprint planning (käyttäjätarinat ja niiden työläys, mitä seuraavassa sprintissä työskennellään)
 - Daily scrum (mitä on tehty ja mitä on työn alla, onko esteitä työlle)
 - Sprint review (mitä on tehty, seuraavat toimet)
 - Sprint retrospective (mitä on tehty, miten prosessia voidaan parantaa)



Kanban

- "TODO, doing, done"



Ohjelmistokehityksen ja suunnittelun vaiheita

- **Vaatimusmäärittely** (Requirements engineering) - Mitä yritetään ratkoa?
- **Sovellusalueenmallinnus** (domain-mallinnus, domain modeling) - Missä sovellusalueella ollaan?
- **Ohjelmistoarkkitehtuurit** (Software architecture design) - Miten ratkaisu toimii, mistä komponenteista se koostuu?
- **Ohjelmistotestaus** (Software testing) - Miten ratkaisu toimii, miten ratkaisua testataan?

- ... Ja paljon muita taitoja: tiimityöskentely, asiakkaan perspektiivi, kulttuurinen muutos



Vaatimusmäärittely

"En ole huolissani koodarien työpaikoista ennen kuin liiketoiminnan ihmiset osaavat tehdä vaatimusmäärittelyä" -Eräs konsultti

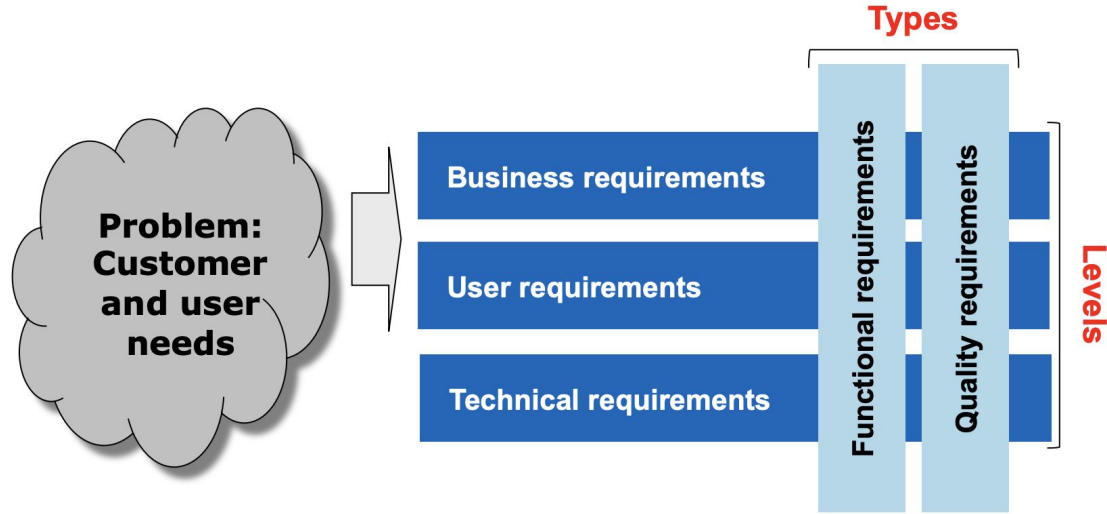
Vaatimusmäärittely

Kolme tasoa vaatimuksille:

- Liiketoiminta
- Käyttäjä
- Teknologia

Kahden tyyppisiä vaatimuksia:

- Toiminnalliset vaatimukset
- Laadulliset (ei-toiminnalliset) vaatimukset



Toiminnalliset vaatimukset (functional requirements)

- Toiminnalliset vaatimukset määrittelevät **mitä tehtäviä** järjestelmän tulee kyetä täyttämään
- Esimerkiksi:
 - Laske kurssin arvosanojen keskiarvo
 - Anna ravintolasuositus lähellä olevista ravintoloista
 - Näytä pankkitilin saldo

Laadulliset/ei-toiminnalliset vaatimukset

- Laadulliset vaatimukset määrittelevät **muuta kuin toiminnallisia** vaatimuksia järjestelmälle
- Esimerkiksi:
 - Suorituskykyyn liittyvät vaatimukset (N yhtäaikaista käyttäjää, järjestelmä vastaa 15 ms sisällä kaikkiin pyyntöihin, ruutu päivittyy 60 kertaa sekunnissa, jne..), järjestelmän vakaus (järjestelmä toiminnassa vähintään 99.99 % ajasta)
 - Luotettavuus (järjestelmän on oltava toiminnassa vähintään 99.99 % ajasta)
 - Turvallisuus
 - Käytettävyys
 - Jne jne

Laadulliset vaatimukset

Havaintoja laadullisista vaatimuksista käytännössä

- Usein sisäänrakennettuja projektiin, mutta saattavat aiheuttaa
 - Isoja yllätyksiä
 - Mittavia muutoksia järjestelmän arkkitehtuuriin
 - Viivästyksiä
 - Projektien epäonnistumisia
- Monilla yrityksillä ei ole hyviä tapoja ja metodeja laadullisten vaatimusten määrittelyyn
- Mihin laadullisiin vaatimuksiin pitäisi keskittyä?
- Kuinka mallintaa kriittiset laadulliset vaatimukset?

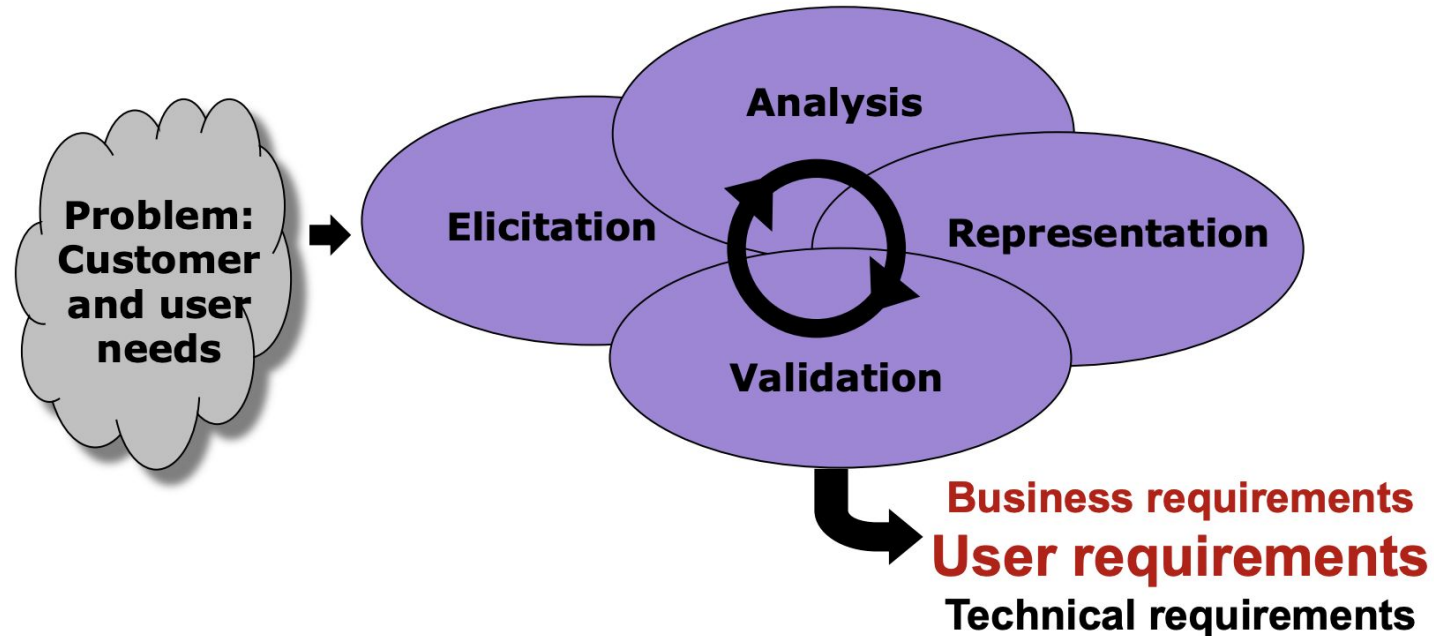
Vaatimusten tasot

- Liiketoiminnan vaatimukset
 - Miksi järjestelmää kehitetään? Voidaan määritellä asiakkaan näkökulmasta tai uutta järjestelmää kehittävä yrityksen näkökulmasta
- Käyttäjien vaatimuksen
 - Mitä järjestelmän tulisi tehdä käyttäjien näkökulmasta? Järjestelmä on käyttäjille musta laatikko (ei tietoa koodista tai järjestelmän sisäisestä rakenteesta)
 - Käyttäjien vaatimukset kuvaavat järjestelmän toiminnallisuuksia tai ominaisuuksia, joiden avulla käyttäjät saavat suoritettua tehtävänsä
- Tekniset vaatimukset
 - Kuvaavat järjestelmän sisäisiä toiminnallisuuksia ja ominaisuuksia. Saatetaan johtaa käyttäjien vaatimuksista, mutta voivat tulla myös muualta kuten sovellusalueesta, standardeista, säädöksistä jne...

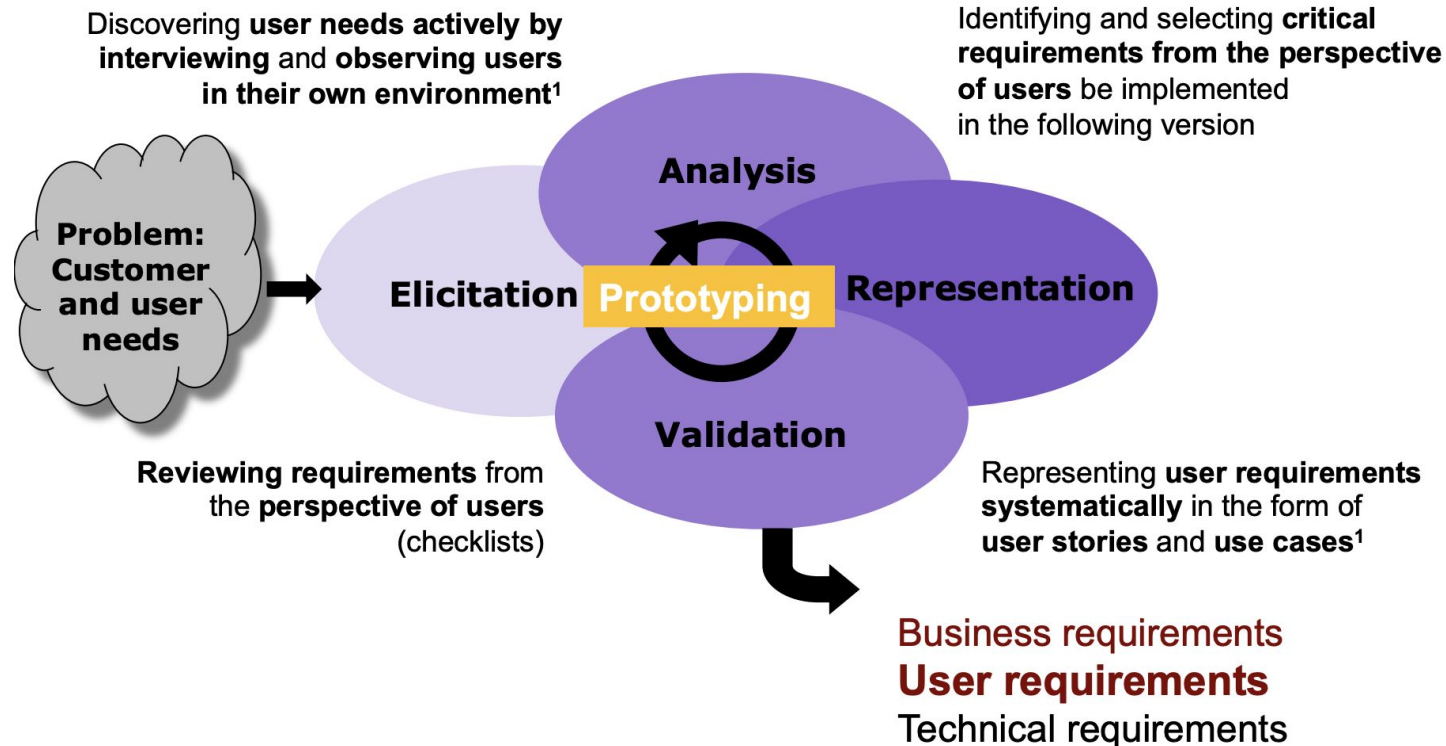
Vaatimusmäärittelyn aktiviteetit (1/3)

- **Elisitaatio**
 - Käyttäjien ja asiakkaiden vaatimusten löytäminen. Vaatimuksia voi löytyä käyttäjien ja asiakkaiden lisäksi sovellusalueesta, markkinatutkimuksista jne
- **Analyysi**
 - Löydettyjen alustavien vaatimusten analysointi ja keskeisten tarpeiden määrittely
- **Esittäminen**
 - Vaatimusten mallintaminen ja dokumentointi systemaattisesti
- **Validaatio**
 - Vaatimusten tarkastelu käyttäjien näkökulmasta, jotta ne todella vastaavat tarpeita. Vaatimuksia voidaan myös tarkastella implementaation, testauksen yms. näkökulmasta

Vaatimusmäärittelyn aktiviteetit (2/3)

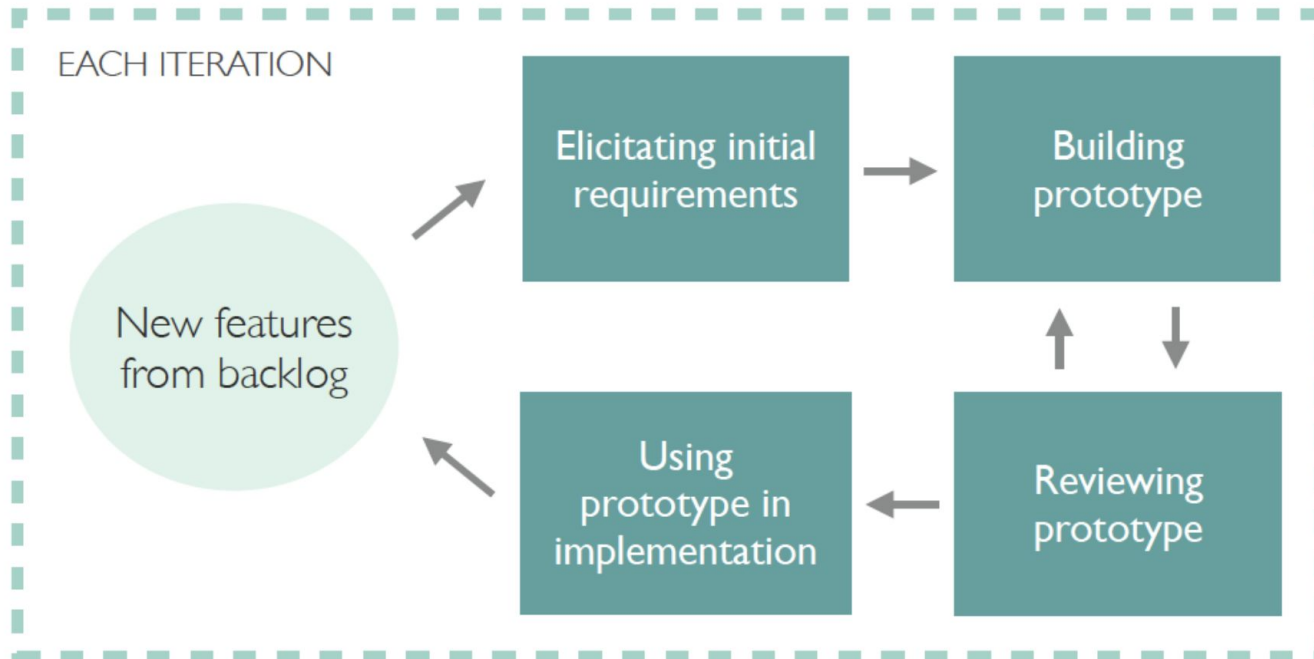


Vaatimusmäärittelyn aktiviteetit (3/3)



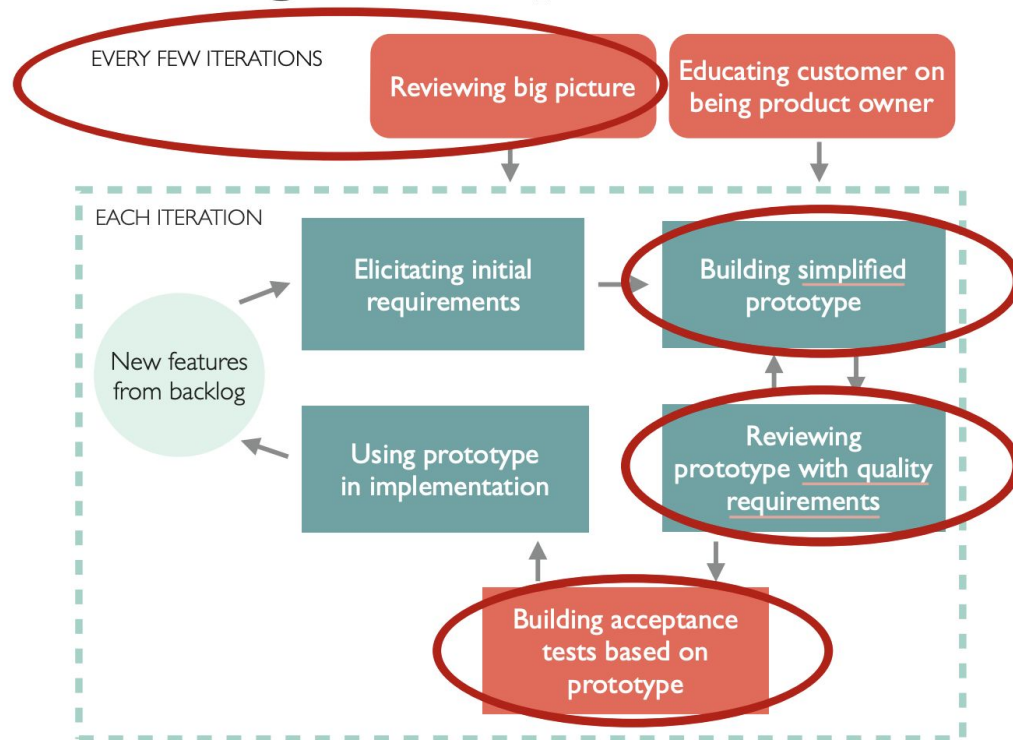
Agile ja vaatimusmäärittely (1/2)

Original agile RE process



Agile ja vaatimusmäärittely (2/2)

Improved agile RE process



Kulttuurinen muutos (cultural change)

- Kulttuurinen muutos tarkoittaa siirtymistä teknologiakeskeisestä lähestymisestä ohjelmistokehityksessä asiakaslähtöiseen näkemykseen
- Vaatimukset on määritellään ja testataan systemaattisesti käyttäjien ja asiakkaiden näkökulmasta

Vaatimusten esittäminen

Vaatimuksia voidaan esittää monilla tavoilla

- Tekstinä ja kuvina
- Listoina vaatimuksia
- **Käyttäjätarinoina (user stories)**
- **Käyttötapauksina (use cases)**
- Muodollisilla kielillä ja metodeilla
- Prototyypeillä (sekä paperiprotoilla että digitaalisesti)
- Videoilla
- Testeillä ja testitapauksilla

Käyttäjätarinat (user story)

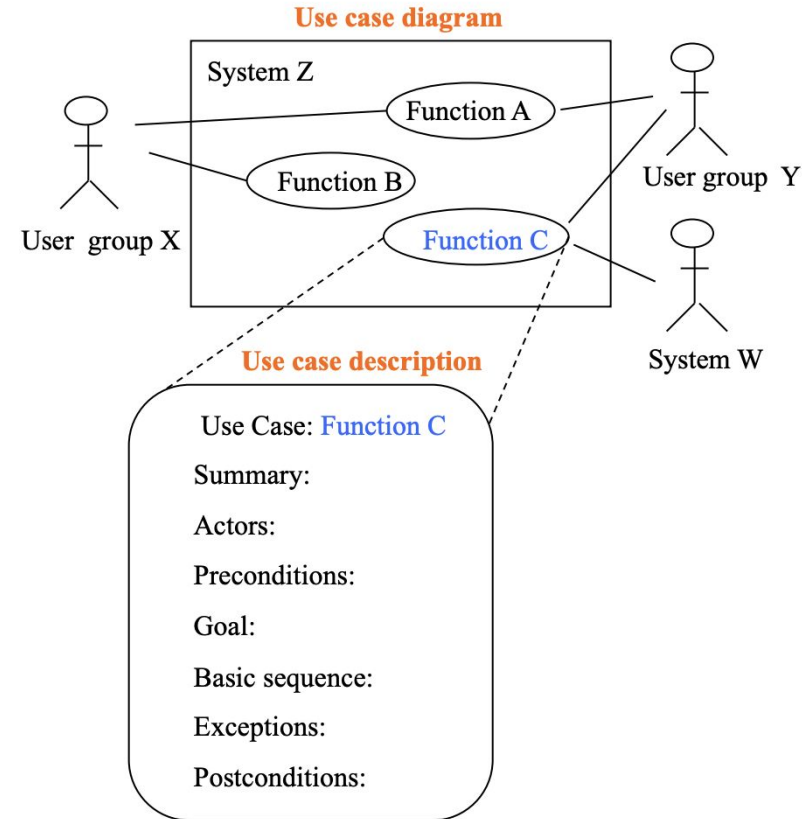
- Käyttäjätarinat kuvaavat toiminnallisuutta, jolla on arvoa joko käyttäjälle tai järjestelmän hankkijalle
- Kolme näkökulmaa käyttäjätarinoille:
 - Kirjallinen kuvaus tarinasta, jota voidaan käyttää suunnittelussa
 - Keskustelut tarinasta auttavat löytämään uusia yksityiskohtia
 - Testaus - tarinan pohjalta voidaan laatia testejä, joilla voidaan esim. varmistaa, että toiminnallisuus on riittävän hyvä ja voidaan siirtyä seuraavan tarinan toteutukseen

Malli käyttäjätarinalle

- Yksi tapa kirjoittaa käyttäjätarina:
 - <Tietynlaisena käyttäjänä> haluan <toiminnallisuutta>, jotta <liiketoiminnallinen arvo>.
- Esim. Kirjan ostajana haluan etsiä ISBN:llä, jotta löydän oikean kirjan nopeasti

Käyttötapaus (Use Case)

- Koostuu kahdesta osasta:
käyttötapausmalli (Use Case Diagram) ja
käyttötapauskuvaus (Use Case Description)
- Käyttötapausmalli
 - Käyttäjät/käyttäjäryhmät
 - Toiminnallisuudet
 - Muut järjestelmät
- Käyttötapauskuvaus
 - Kriittisille ja monimutkaisemmille
(toiminnallisille) vaatimuksille käyttäjän
näkökulmasta



Käyttötapauskaavio (Use case diagram)

BIG PICTURE:

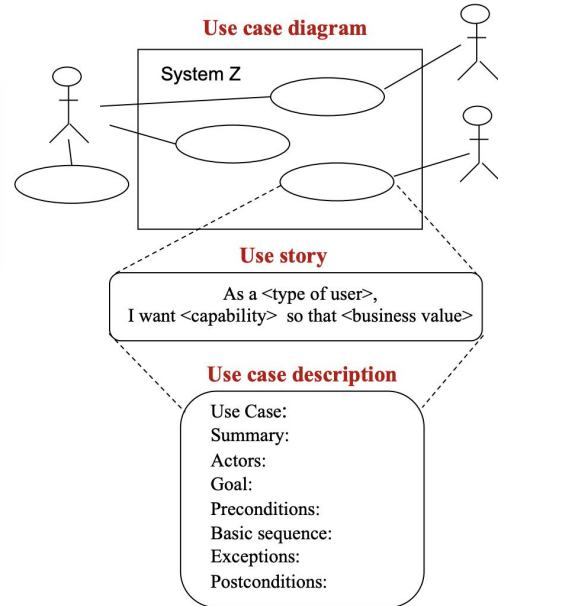
The use case diagram gives an overview of the system.

SCOPING:

which functions are inside and which outside of the system

The use case and user story methods for modelling functional user requirements.

Use case descriptions are good for complex functional user requirements.



Boundary of the system



Actor = user group or system



Function = use case

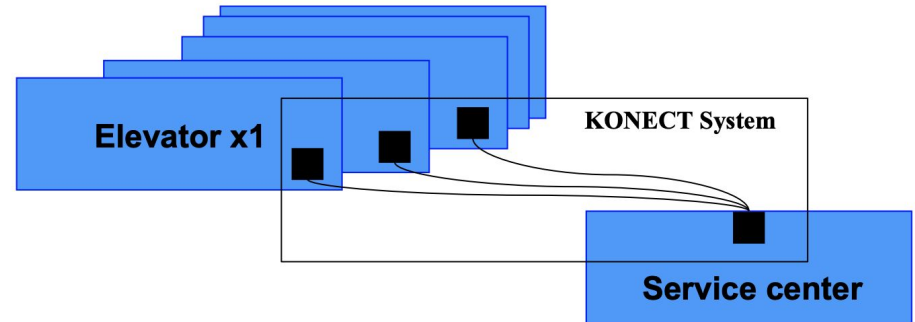
Malli käyttötapauskuvaukselle (Use case description)

- Käyttötapauksen nimi
- Lyhyt yhteenveto
- Toimijat: käyttäjäryhmät ja mahdolliset muut järjestelmät, jotka liittyvät käyttötapaukseen
- Ensisijaisen käyttäjän tavoite: kuvaus miksi toiminnallisuus on tärkeä käyttäjän näkökulmasta
- Normaalisekvenssi: kuvaus keskeisistä tapahtumista/toimista käyttötapauksen kuluessa (olettaen, että kaikki toimii oikein)
- Poikkeukset: mahdolliset poikkeus- tai virhetilat
- Lopputila: kuvaus käyttötapauksen päättyessä

Esimerkki: käyttötapaus

- Uusi monitorointijärjestelmä hisseille
- Liiketoiminnan vaatimukset:
 - Käyttäjät saavat turvallista, luotettavaa ja nopeaa palvelua 24/7
 - Kone Elevators saa uusia asiakkaita ja palvelusopimuksia

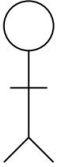
KONECT: On-line monitoring system of thousands of elevators



Esimerkki: käyttötapauksen taustaa


- Pitkä lista vaatimuksia, jotkin näistä käyttäjien vaatimuksia, mutta suurin osa teknisiä vaatimuksia
- Projekti oli pilottivaiheessa
- Uusi testauspäälikkö tarvitsi tukea

Esimerkki: käyttötapauskaavio




Passenger

KONECT System

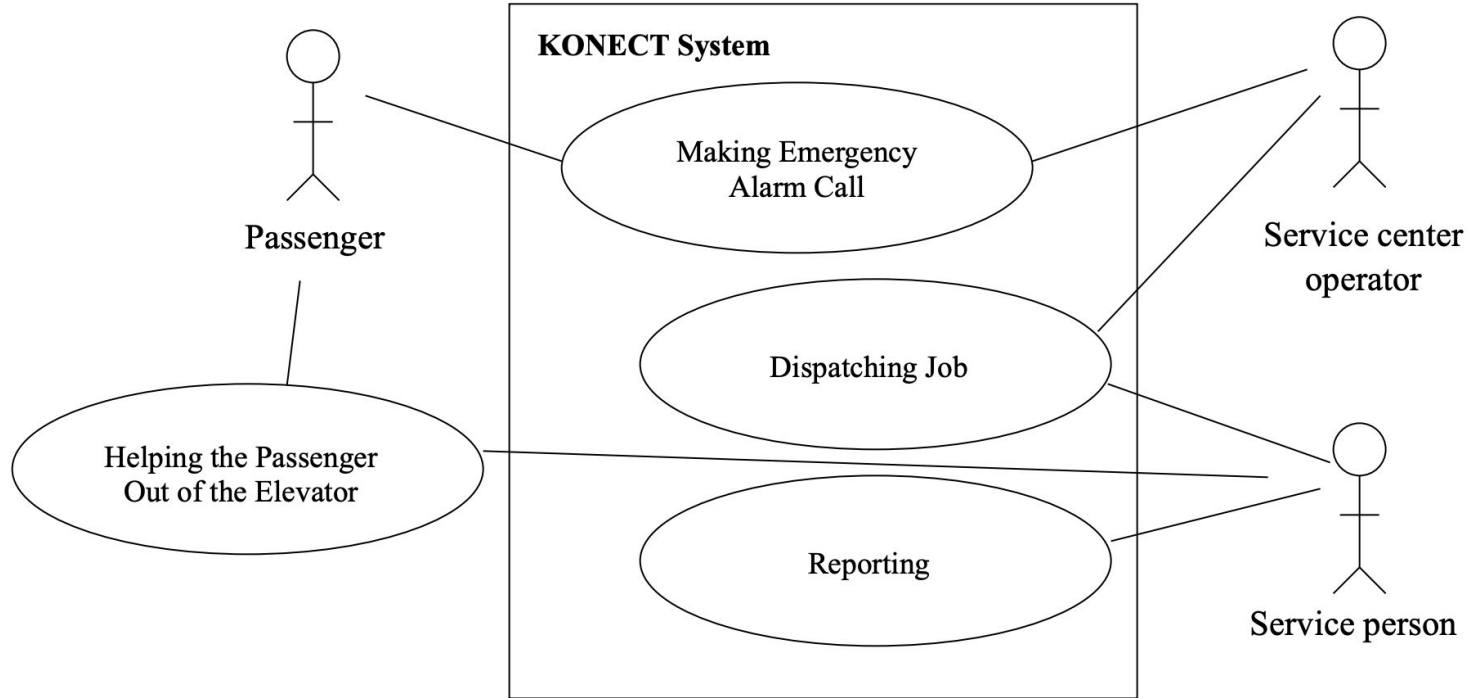


Service center
operator

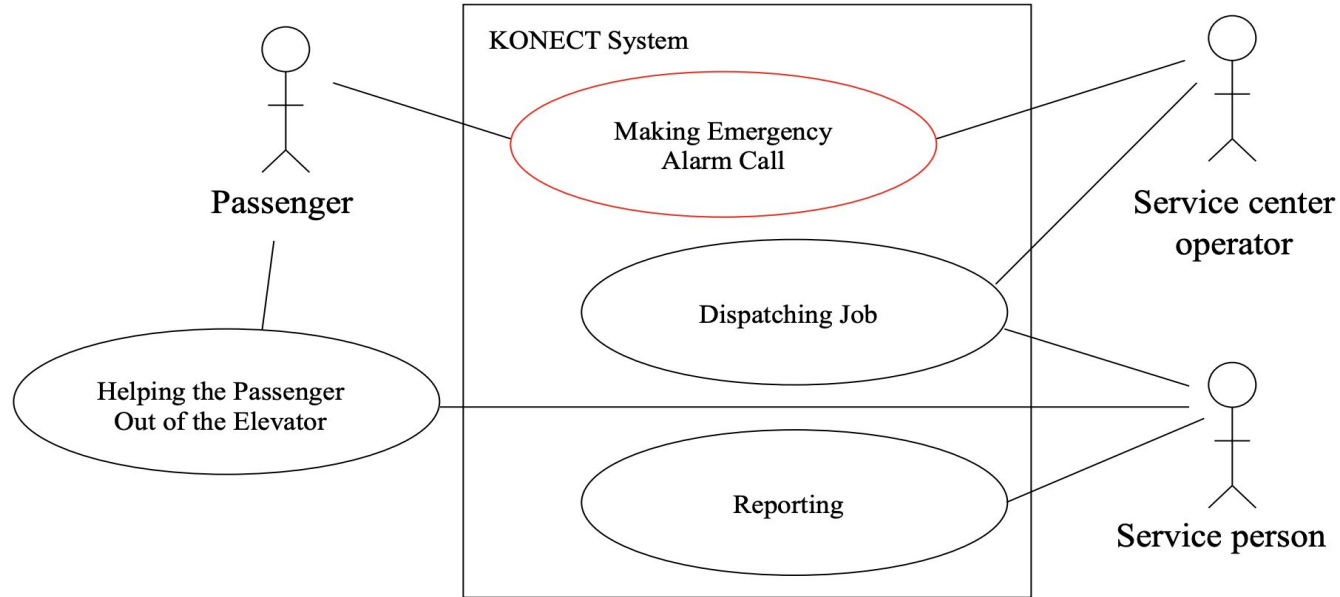


Service person

Esimerkki: käyttötapauskaavio



Esimerkki: käyttötapauskaavio



Tarkistuslista käyttötapauskaavioille

- Ovatko kaikki käyttäjäryhmät ja tärkeät toiminnallisuudet mukana?
- Ovatko kaikki ulkoiset järjestelmät ja rajapinnat näkyvillä?
- Onko käyttötapaukset kirjoitettu käyttäjän näkökulmasta?
- Ovatko toimijoiden nimet kuvaavia?
- Onko kaavio selkeä (ei kasa sekalaisia viivoja)?
- Onko käyttötapauksia liikaa?
 - Suositus maksimissaan 10-12 tapausta
 - Jos enemmän käyttötapauksia -> jaa kaavio kahteen järkevään osaan

Esimerkki: käyttötapauskuvaus

- **Use Case:** Making Emergency Alarm Call
- **Summary:** An entrapped passenger pushes the emergency alarm button in order to get help. A service center operator receives the emergency alarm call and informs the passenger that a serviceman will come and let the passenger out of the elevator.
- **Actors:** Passenger and service center operator
- **Preconditions:** An elevator has stopped between floors and there is a passenger in the elevator.
- **Goal of the primary users:** The passenger wants to get out of the elevator safely and as fast as possible.

Esimerkki: käyttötapauskuvaus - perussekvenssi

Basic sequence:

Step 1: The **passenger** presses the emergency alarm button in the elevator. ¹

Step 2: The **service center operator** gets a visible notification about the emergency alarm call on the screen with an optional audio signal.

Step 3: The **service center operator** accepts the emergency alarm call to be handled.

Step 4: The system opens voice connection between the **service center operator** and the **passenger**.

Step 5: The system indicates the **service center operator** and the **passenger** that voice connection is open.

Step 6: The system guides the **service center operator** what information to ask from the passenger (link to the use case called Guiding the service center operator).²

Step 7: The **service center operator** informs the system that the emergency alarm call is correct.

¹ This is a special case. Elevator standards require the emergency alarm button. Typically, use cases should not contain user interface details.

² The number of steps can be reduced by using use cases hierarchically.

Esimerkki: käyttötapauskuvaus - poikkeukset

Exceptions:

Step 1: If an entrapped passenger does not push the alarm button long enough (more than 3 seconds), the system alerts the passenger with voice announcement.

Step 7: If the passenger has pressed the emergency alarm button by accident, the service center operator informs the system that the emergency alarm call is false. The system resets the emergency alarm call.

Postconditions:

The entrapped passenger knows that service center operator will contact a service person who will help the passenger out of the elevator safely and as soon as possible. The system knows that the alarm call is correct.

Ohjeita käyttötapauskuvaukseen

- Keskity perussekvenssiin ensiksi
- Analysoi poikkeukset askel askeleelta
- Käytä esi- ja loppuehtoja tarkistaaksesi mahdollisia puutteita käyttötapauksen välillä
- Käyttötapaukset kannattaa kirjoittaa yhteistyössä (muiden kehittäjien, asiakkaiden, product ownerin yms. kanssa)

Tarkistuslista käyttötapauskuvauksille

- Onko kriittiset ja monimutkaiset toiminnallisuudet määritelty selkeästi?
- Käytetäänko kaaviossa ja kuvauksessa samoja nimiä (toiminnallisuudet, käyttäjät)
- Onko kuvaus kirjoitettu käyttäjien kielellä?
- Onko mukana käyttöliittymän yksityiskohtia tai toteutukseen liittyvää tietoa -> pois

Käyttötapauksien hyötyjä

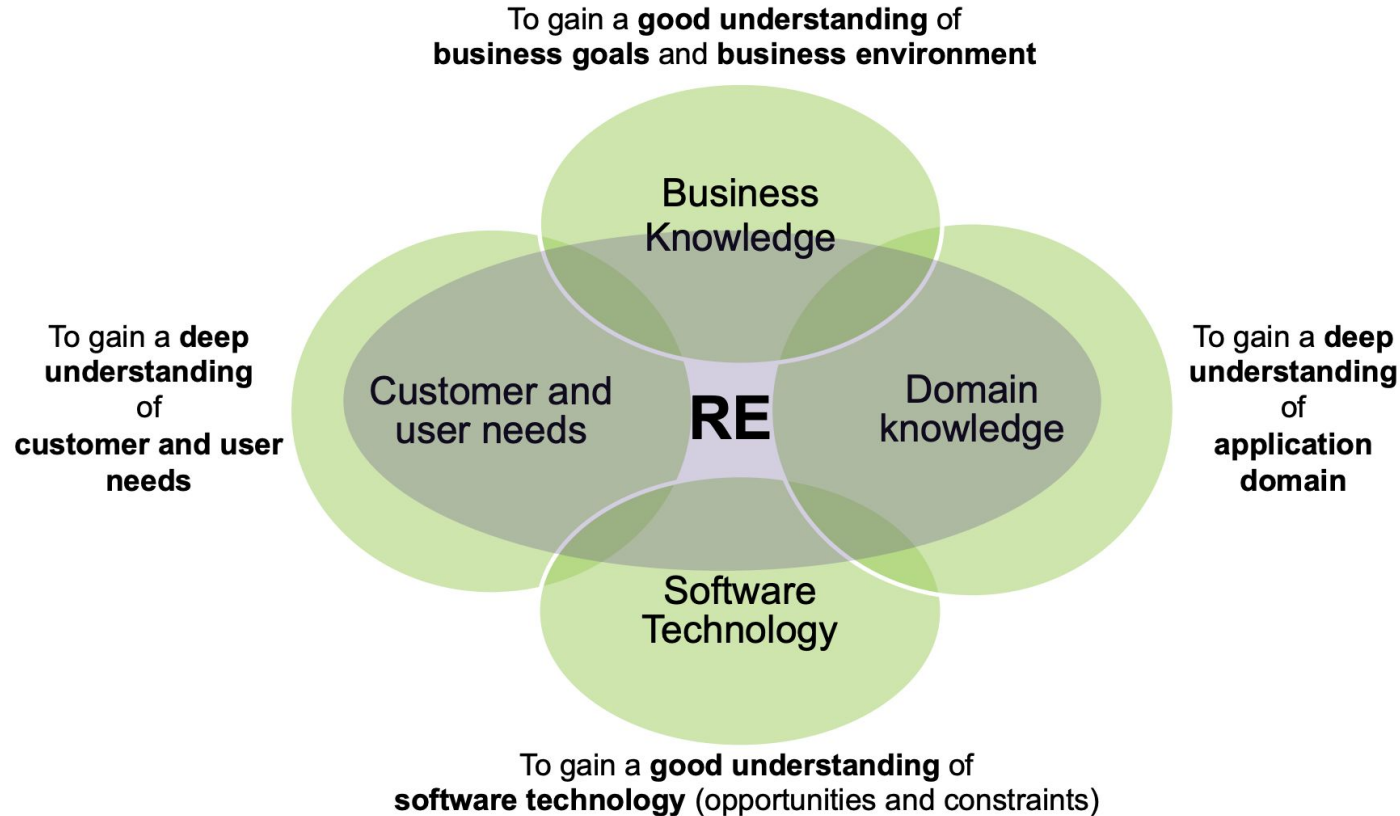
- Tukee vaatimusten systemaattista määrittelyä käyttäjän näkökulmasta (tukee myös kulttuurinen muutos)
- Korkean tason kuvaus järjestelmästä *käyttäjien ymmärtämällä kielellä*
- Tapa kerätä asiakkailta ja käyttäjiltä palautetta
- Tukee ohjelmiston arkkitehtuurin suunnittelua
- Ohjaa testausta (erityisesti hyväksyntätestaus, acceptance testing)
- Tukee projektin suunnittelua yleisesti

Käyttötapausten haasteita

- Hankala määritellä oikealla tasolla
 - Ei käyttöliittymän yksityiskohtia
 - Ei sisäistä toiminnallisuutta
- Haasta määritellä monimutkaisia sekvenssejä, kuten:
 - "Silmukkaprosesseja"
 - Ehdollisia sekvenssejä
- Hyvien käyttötapausten laatiminen vaatii
 - Ymmärrystä käyttäjien tehtävistä ja tavoitteista
 - Harjoitusta, harjoitusta ja harjoitusta

Silmukkaprosessien ja ehdollisten sekvenssien mallintamiseen on parempia tapoja, esim. flow-diagrammit

Vaatimusmäärittely yhdessä kalvossa



Tehtävä: vaatimusmäärittely

- Yrityksen näkökulmasta, luo:
 - 2-3 käyttäjätarinaa
 - Piirrä käyttötapauskaavio, valitse yksi käyttäjätarina ja kirjoita käyttötapauskuvaus tälle
 - Jos aikaa, määrittele ja perustele tärkeimmät laadulliset vaatimukset
- Mieti uuden työntekijän kannalta - mitä olisi ollut hyvä ymmärtää kun itse aloitit
- Käydään yhdessä läpi

Lounas

Sovellusalueen mallinnus (domainmallinnus, domain modeling)

- Pääasiallinen tarkoitus on ymmärtää *miten* sovellusalueella toimitaan, mitä konsepteja siihen liittyy ja millaisia suhteita näiden konseptien välillä on
- Jotkin ohjelmistokehitysmenetelmät lähtevät domainmallinnuksesta ja se ohjaa ohjelmiston kehitystä

Mallin määrittelyä

- Yksinkertaistettu kuvaus jostakin järjestelmästä, tilanteesta tai prosessista näiden ymmärtämiseksi ja kuvaamiseksi
 - Konseptuaalinen tai mentaalinen malli
- Abstrakti kuvaus, joka auttaa kommunikoinnissa (esim. arkkitehtuuri) eri sidosryhmien (stakeholder) välillä

Mallien ominaisuuksia

- Malleilla on kolme tärkeää ominaisuutta (Stachowiak, General Model Theory, 1973)
 - *Kartoitus*
 - Malli pohjautuu "alkuperäiseen"
 - *Rajaus*
 - Malli kuvaa vain tärkeitä tai rajattua joukkoa ominaisuuksia
 - *Käytännöllisyys*
 - Malli tehdään jostain syystä, mallia voidaan hyödyntää jotenkin some purpose
- Kaikki mallit ovat virheellisiä, mutta jotkut mallit ovat hyödyllisiä

Esimerkki mallista



Mallien käyttöä

- Ymmärryksen lisääminen
- Kommunikaatio
- Riskien hallinta
- Visualisointi
- Suunnittelu
- Vaihtoehtojen kartoittaminen nopeasti ja halvemmalla
 - Nopeampaa ja halvempaa kuin vaihtoehtoisten järjestelmien luonti

Erilaisia malleja

- Deskriptiivinen

- Kuvaa rakennetta tai olemassaolevaa järjestelmää
- Esim. vaatimukset, sovellusalueen malli

- Preskriptiivinen

- Kuvaa miten asioiden tulisi olla, esim. miten eri osa-alueiden järjestelmässä tulisi toimia keskenään

- Preskriptiivinen malli voi myöhemmin toimia deskriptiivisenä mallina

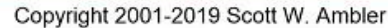
Ketterä mallintaminen (1/2)

- Ei tarkoita mallien välttämistä
- Tukee **ymmärtämistä ja kommunikaatiota**, ei dokumentaatiota
 - Jotkin mallit ovat käytössä vain hetkellisesti (mallit kommunikointiin ja selkeyttämiseen)
 - Toiset mallit ovat pitkäaikaisempia (domain mallit, arkkitehtuuriset suunnitelmat, vaatimuksien määrittelyt)
- Mallinna vain kaikkein tärkeimmät osat (keskeiset konseptit, kriittiset osat implementaatiosta)
- Käytä yksinkertaisimpia mahdollisia työkaluja (post-it laput riittävät monesti)

Ketterä mallinnus (2/2)

- Kommunikaatio! Mallinna yhdessä muiden (esim. asiakkaat) kanssa
- Käytä riittävän hyvää notaatiota ja yleisiä käytäntöjä
 - Tarkka mallinnuskieli ja yksityiskohtaiset merkinnät eivät usein ole tärkeitä
- Muista että kaikki mallit ovat epätarkkoja -> tähtää hyödyllisyyteen
- Ks. myös 14 core practices of Agile Model Driven Development (AMDD): <http://agilemodeling.com/essays/bestPractices.htm>

<http://agilemodeling.com/essays/bestPractices.htm>

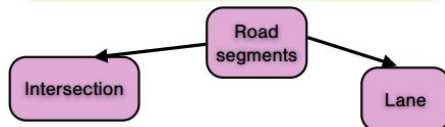


Mallinnuskieliä ja -menetelmiä

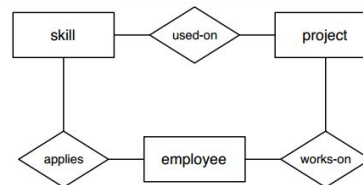
Glossary

Concept	Definition	Synonyms
Intersection	An intersection represents a place where route selection must take place. From each intersection a number of roads can enter and leave.	junction
Lane	Each intersection has one or more lanes entering from one direction.	
Road segment	A road segment is a directed path from one intersection to another.	Street segment
Route	A route represents a navigation concept that shows a path to navigate.	Way (see below)
Way	An ordered set of road segments is a way. Way has a user understandable name, e.g., "Abbey Road".	Street, road, highway, avenue, alley,...

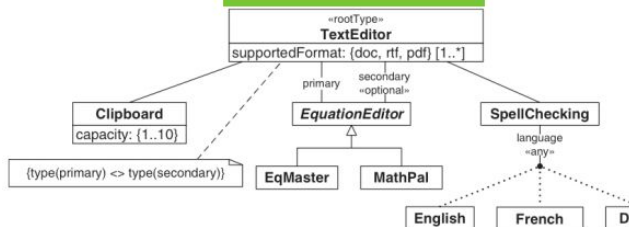
Boxes & lines



Entity-relationship



UML_(ish)



Formal models

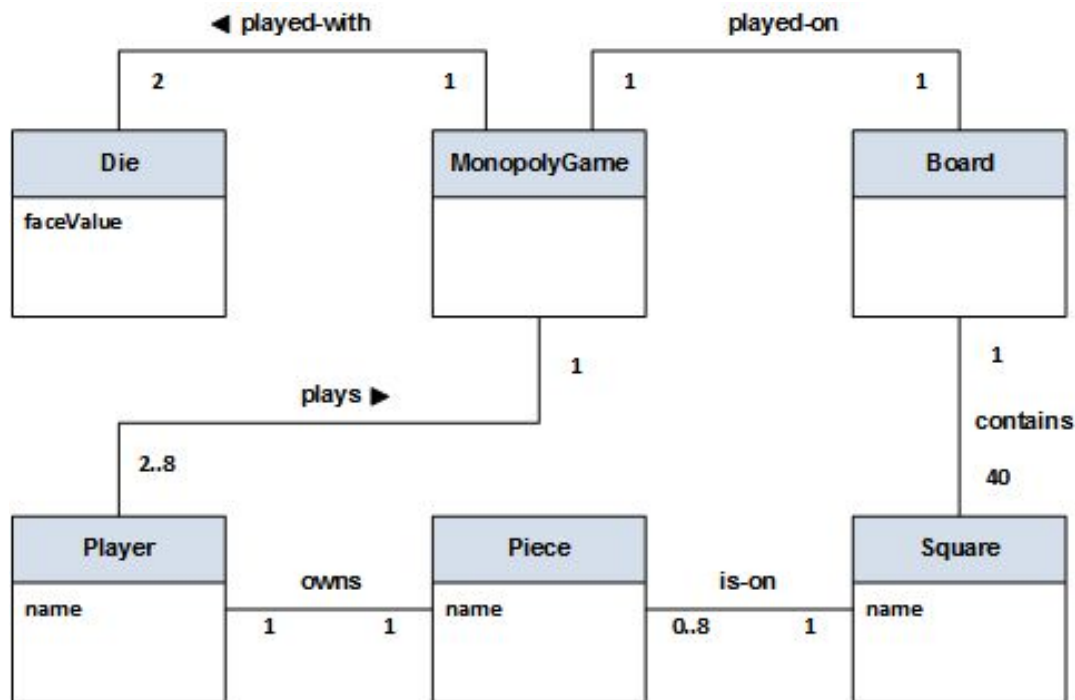
```
instanceOfD(book, productType) ←  
hasValueD(book, taxRate, 7) ←  
hasAttrD(book, nrOfPages, 1, pageDomain, 1, 1) ←  
containsD(pageDomain, 520..530) ←  
abstract(book) ←  
inGSetD(novel, book, style) ←
```

And many more...

Hyvän mallin piirteitä

- Tarkoituksenmukainen
- Käytännöllinen
- Ymmärrettävä
- Yksinkertainen
- Yksiselitteinen
- Konteksti on tärkeä!
- *Tarpeeksi* tarkka
- *Tarpeeksi* yksityiskohtainen
- "Riittävän hyvä"
- Johdonmukainen
- Tuottaa **lisäarvoa**

Yksinkertainen malli monopolista



Sovellusalueen mallit ovat ongelmalähtöisiä

- Esittää **oikean maailman ongelman** käyttäen sovellusalueen terminologiaa
 - Ei kuvaus ohjelmistoartefakteista
- Kutsutaan myös konseptuaaliseksi malliksi
- Esittää avain konseptit (concepts), niiden attribuutit (attributes) ja suhteet (relationships/associations)
- Visuaalinen sanakirja
- Käytetään myös muualla kuin ohjelmistokehityksessä

Syitä domainmallinnukselle

- Saavutetaan **yhteinen ymmärrys** avainkonsepteista ja sovellusalueen sanasto
 - Ohjelmistoja kehitetään monilla eri sovellusalueilla, terveys, talous, logistiikka, jne
- Pienentää eroa mentaalisen mallin ja sovelluksen esityksen välillä
- Erityisen hyödyllinen alueille, jotka eivät ole entuudestaan tuttuja

Esimerkki ohjelmoinnin opetuksen sovellusalueesta

Question 1

Examine the following expression:

```
max(4, 5) * min(6 - pow(5, sqrt(2) + 1), 120 / abs(-10))
```

When this expression is evaluated, which of the following operations take place before the abs (absolute value) operation?

- ☐ choosing the greater number (max)
- ☐ multiplication (*)
- ☐ choosing the lesser number (min)
- ☐ subtraction (-)
- ☐ exponentiation (pow)
- ☐ computing a square root (sqrt)
- ☐ addition (+)
- ☐ division (/)

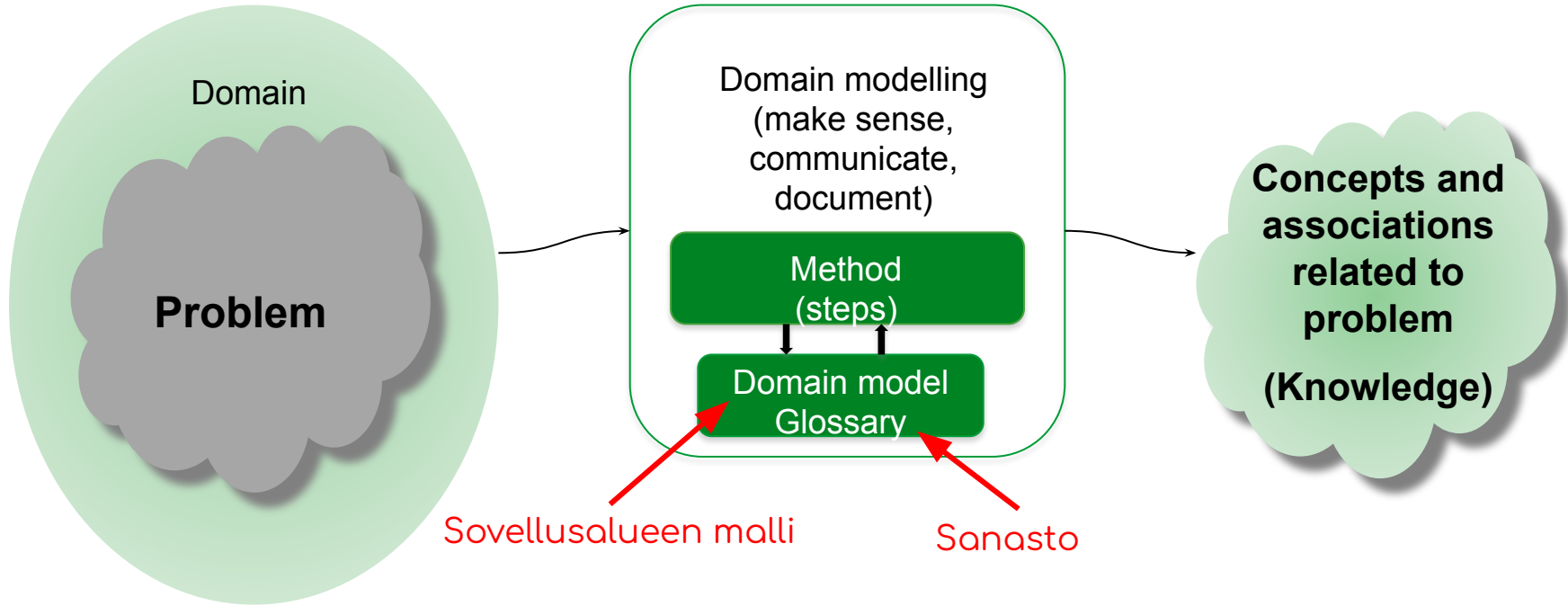
Submit

Mitä tarkoittaa ratkaisu, vastaus, mallivastaus, palautus, kysymys, vaihtoehto, valittu vaihtoehto... jne?

Sanasto

- Aakkostettu lista keskeisistä konsepteista sovellusalueella
- Käytetään tukemaan sovellusalueen mallia

Domain mallinnus – Ongelman ymmärtämistä



Joitakin huomioita mallintamisesta

- Haasteena löytää sopiva abstraktion taso
 - Liian yksityiskohtaiset mallit eivät välttämättä ole hyödyllisiä...
 - ...ja liian yksinkertaiset mallit ovat hyödyttömiä
- Ei ole olemassa "yhtä oikeaa" mallia
- Vaatii kokemusta ja ymmärrystä
- Sovellusalueen mallin päivittäminen on hyödyllistä projektin aikana
 - Kerryttää tietoa ja mahdollisia muutoksia sovellusalueeseen
 - Säilyttää tietoa (esim. uusille tiimin jäsenille)
 - Vältetään liian suuria etukäteissuunnitelmia(Big Design Up Front , BDUF)

Domain mallit - Sanasto

- Kuvaa tarkemmin konsepteja mallissa
 - Mitä konsepteilla tarkoitetaan ja mitä ne kuvastavat
- Tarkoituksena vähentää epäselvyyttä ja parantaa kommunikaatiota
- Eri sidosryhmät saattavat käyttää samaa termiä eri tarkoituksilla
 - Johtaa väärinymmäryksiin
- Lisää sanastoon synonyymit, kun sellaisia löytyy
- Saattaa sisältää suuren määrän konsepteja

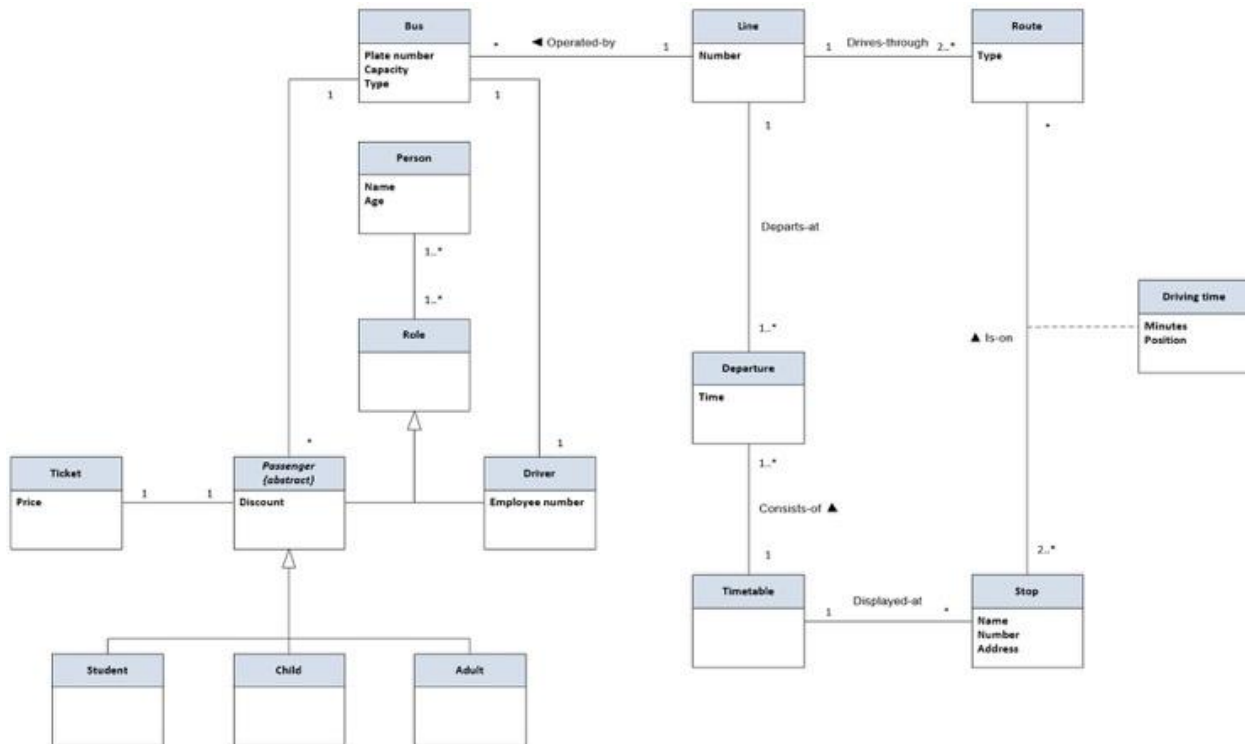
Sanasto - Esimerkki

Concept	Definition	Synonyms
Intersection	An intersection represents a place where route selection must take place. From each intersection a number of roads can enter and leave.	junction
Lane	Each intersection has one or more lanes entering from one direction.	
Road segment	A road segment is a directed path from one intersection to another.	Street segment
Route	A route represents a navigation concept that shows a path to navigate.	Way (see below)
Way	An ordered set of road segments is a way. Way has a user understandable name, e.g., "Abbey Road".	Street, road, highway, avenue, alley,...

Sovellusalueen malli, domainmalli

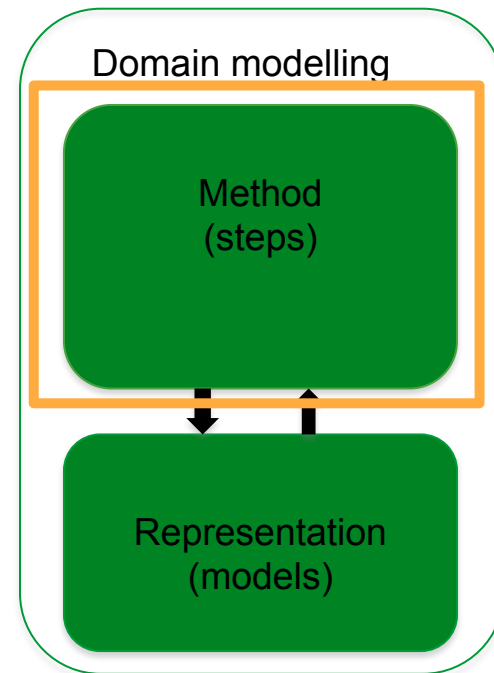
- UML luokkadiagrammi
 - Luultavasti yleisin
- Entity-relationship kaavio
 - Voidaan käyttää domain mallien esittämiseen
 - Hyvä pohja tietokannan mallille
 - Domainmallit eivät ole pelkästään tietokantamalleja
- Painotus on kommunikaatiolla
- Käytä merkintätapoja, jotka kaikki ymmärtävät
- Lisää selityksiä symboleille ja muille erityisille merkintätavoille

Domainmalli - Esimerkki



Domainmallinnus - menetelmä

1. Tunnista avainkonsepti
 - Kerää ja analysoi kandidaatteja
 - Poista synonyymit ja monikot
 - Valitse tärkeimmät konseptit mallinnettavaksi
2. Tunnista tärkeimmät attribuutit konsepteille
3. Tunnista tärkeimmät suhteet/assosiaatiot konsepteille
 - Suhteet, jotka on syytä muistaa ja ymmärtää
4. Määrittele suhteiden ominaisuudet



* The order of concept refining steps may vary depending on the modeler and the amount of information

DM menetelmä: 1. Tunnista avainkonseptit

- Tunnista substantiivit
 - Esimerkiksi vaatimuksista, alan asiantuntijoilta, teknisistä dokumenteista ja standardeista, säädöksistä ja laeista
- Uudelleenkäytä ja muokkaa olemassaolevia malleja*
- Analysoi ja tunnista konsepteja olemassa olevista järjestelmistä

* e.g. http://www.databaseanswers.org/data_models/ (data models)

Konseptit

- Niminä käytetään substantiiveja
 - Poista synonyymit
 - Nimeä yksiselitteisesti
- **EIVÄT** ole ohjelmointiin liittyviä luokkia
 - Esim. Java
- Joillain sovellusalueilla käytetään myös ohjelmistoon liittyviä konsepteja
 - Esim. reitittimet telekommunikaatiossa

Kartoittajan ohjenuorat

- Käytä alueen nimitystä
 - Esim. matkustaja eikä asiakas, kun puhutaan bussiliikennöinnistä
- Rajaa epärelevantit ominaisuudet ulkopuolelle
 - Esim. moottori ei ole oleellinen, jos tarkoituksena on kuvata bussien reittejä ja aikatauluja
- Älä lisää sovellusalueen ulkopuolisia konsepteja
 - Jos kyse on ainoastaan bussiliikennöinnistä, ei ole syytä lisätä esim. laivaliikenteeseen liittyviä konsepteja

Konsepteja bussiliikennöinnissä?

Reittioapas

Route

15

STOPS TIMETABLE DISRUPTIONS

Jupperi → Otaniemi

Right now	Stop	Leaves	Next
	Sellimäki E2402 Vanhan-Markkian tie	5 min	11:45
	Visamäki E2069 Kolovälsäntie	6 min	11:46
	Tuohimäki E2071 Kalevalantie	8 min	11:48
	Kaskenkaataja E2110 Pohjantie	9 min	11:49
	WeeGee-talo E2108 Pohjantie	11:28	11:50
	Pohjantie (M) E2106 Pohjantie	11:29	11:51
	Tuulimäki E2014 Etelävalatie	11:30	11:52
	Sateentie E2010 Tapiolantie	11:32	11:54
	Kontiontie E2030 Tapiolantie	11:34	11:56
	Otsolahdentie E2029 Tapiolantie	11:34	11:56
	Ilkäranta E2027 Tapiolantie	11:35	11:57
	Tekniikantie E2210 Tekniikantie	11:36	11:58
	Vuorimies E2212 Vuorimiehentie	11:37	11:59
	Kemisti E2230 Vuorimiehentie	11:37	11:59

Timetables and routes

Route maps Terminals Printable timetables

Route search

From?

To?

Time Departure time Arrival time

Date

Search route

Timetables

Search timetables

Stops

Search stops

Disruption info

Printable timetables

- Summer 2017
- Winter 2017 - 2018

Changes to routes and stops in Pasila

Tickets and fares

Employee authorized commuter ticket Mobile ticket

1h 2h 1wk 7wk 3wk

Single tickets
For single trips

Day tickets
For 1-7 day periods

Travel cards
For regular use

Mahdollisia konsepteja bussiliikennöinnissä

- Bus
- Driver
- Stop
- Passenger
- Route
- Receipt
- Line
- Ticket
- Timetable
- Departures
- Customer
- Schedule

Mahdollisia konsepteja bussiliikennöinnissä

- Bus
- Driver
- Stop
- Passenger
- Route
- Receipt
- Line
- Ticket
- Timetable
- Departures
- Customer
- Schedule

DM Menetelmä 2: Tunnista tärkeimmät attribuutit

- Konseptin loogisia ominaisuuksia, jotka ovat tärkeitä ongelman kannalta
- Sisällytä kaikki attribuutit, jotka ovat vaatimusten (käyttäjätarinat, käyttötapaukset) kannalta tärkeitä
- Liitä toisiinsa liittyvät konseptit yhteen relaatioilla eikä attribuuteilla

Attribuutit bussiliikennöinnissä?

Bus

Line

Departure

Ticket

Passenger

Driver

Stop

Route

Timetable

Attribuutit bussiliikennöinnissä

Bus
Plate number Capacity Type

Line
Number

Departure
Number

Ticket
Price

Passenger
Discount

Driver
Employee number

Stop
Name Number Address

Route

Timetable

DM Menetelmä: 3. Tunnista tärkeimmät assosiaatiot

- Merkitsee jotain tärkeää tai mielenkiintoista yhteyttä oikeassa elämässä
- Koska assosiaatio on tärkeä?
 - Kun sitä tarvitaan jonkin suhteen ymmärtämisessä, asian laskennassa, jne
- Vältä turhia assosiaatioita
 - Heikentää luettavuutta

Esimerkkejä erilaisista assosiaatiosta

- A on fyysinen tai looginen osa B:tä (stop-route)
- A sisältyy fyysisesti tai loogisesti B:n (passenger – bus)
- A kuuluu (joukkoon) B (bus driver – employee)
- A käyttää tai hallinnoi B:tä
- A on osa transaktiota tai raporttia B (ticket – receipt)

Bussiliikennöinti - assosiaatiot

Bus
Plate number Capacity Type

Line
Number

Departure
Number

Ticket
Price

Passenger
Discount

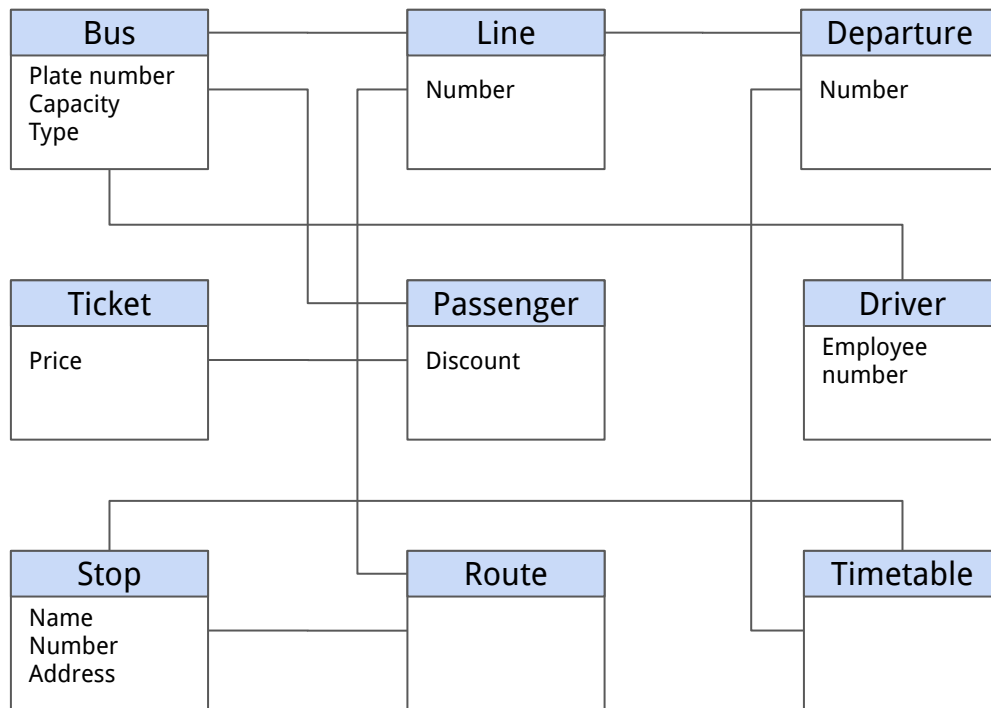
Driver
Employee number

Stop
Name Number Address

Route

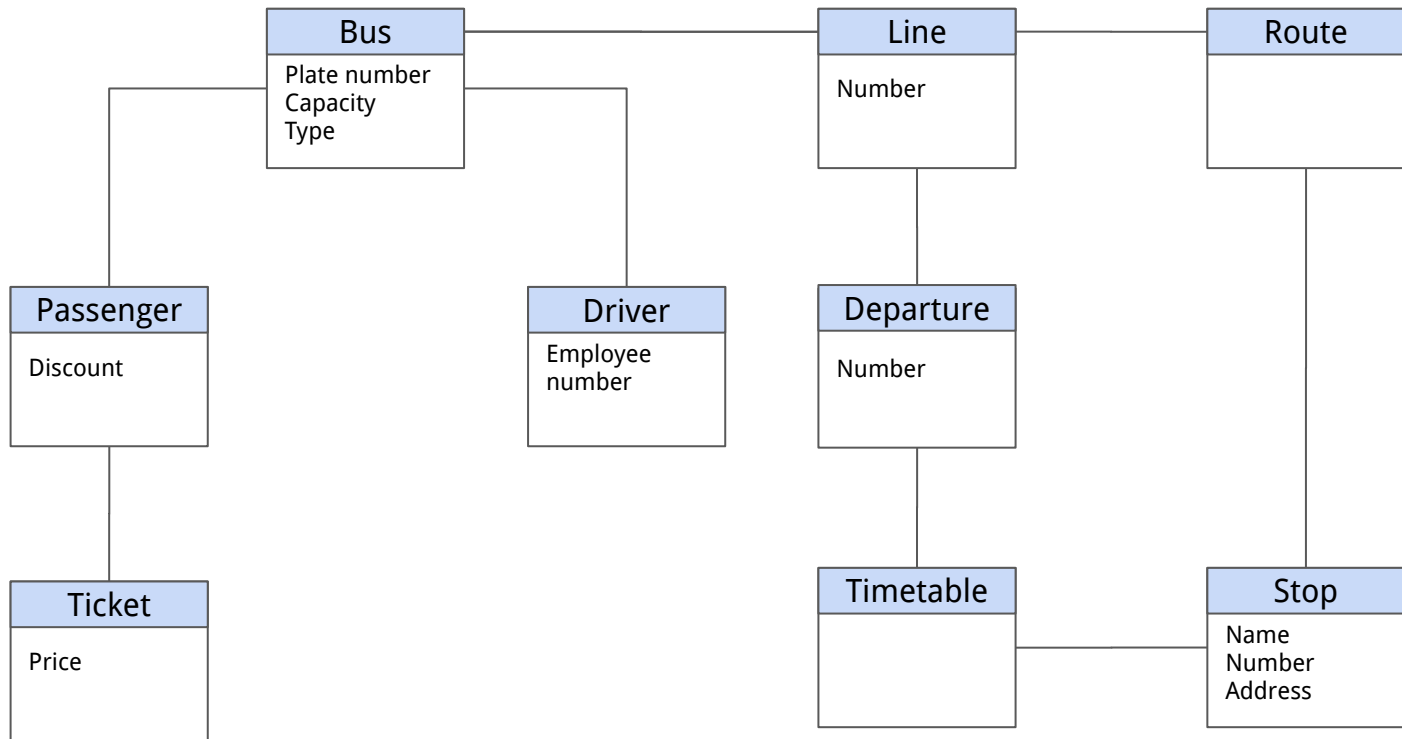
Timetable

Bussiliikennöinti - assosiaatiot



Selkeä?

Bussiliikennöinti - assosiaatiot



Parempi?

DM Menetelmä: 4. Assosiaation ominaisuudet - nimeäminen

- Jos suhde ei ole täysin ilmeinen, nimeä assosiaatio
 - E.g., maksaa (asiakas-lippu)
 - Vältä yleisiä nimiä ("käyttää", "on")
- Lisää assosiaatiolle suunta, jos sitä ei lueta vasemmalta oikealle

DM Menetelmä: 4. Assosiaation ominaisuudet - kardinaliteetit cardinalities

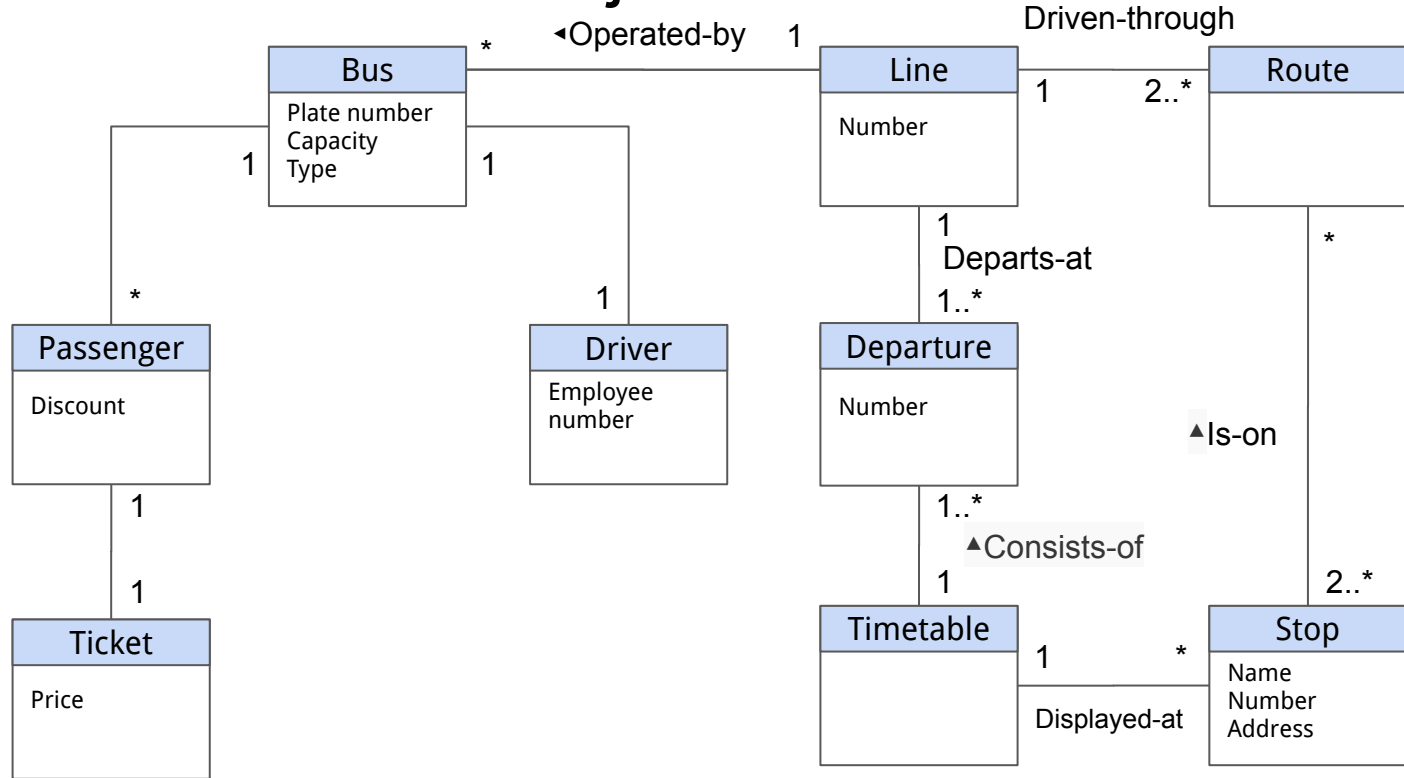
- Kardinaliteetit (monikot)

- Määrittelee kuinka monta A:n instanssia liittyy yhteen B:n instanssiin tietyllä hetkellä
- Kommunikoi sovellusalueen rajoituksia, jotka *saattavat* vaikuttaa myös ohjelmistoon

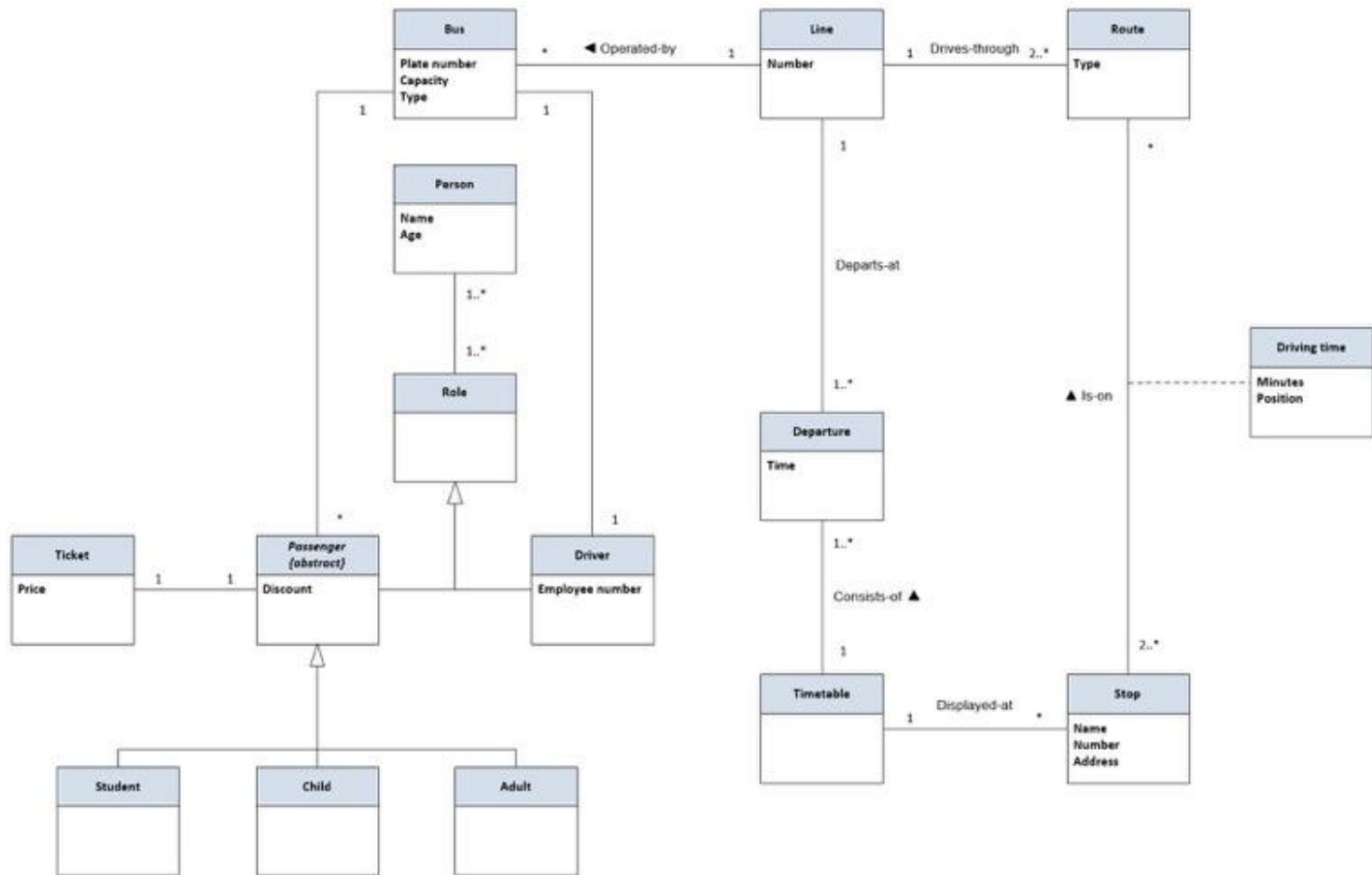
- Esimerkkejä

- * 0 tai monta
- 1..* 1 tai monta
- 1..40 Väliltä 1 - 40 (inklusiivinen)
- 5 tasan 5
- 1,5,13 tasan 1, 5 tai 13

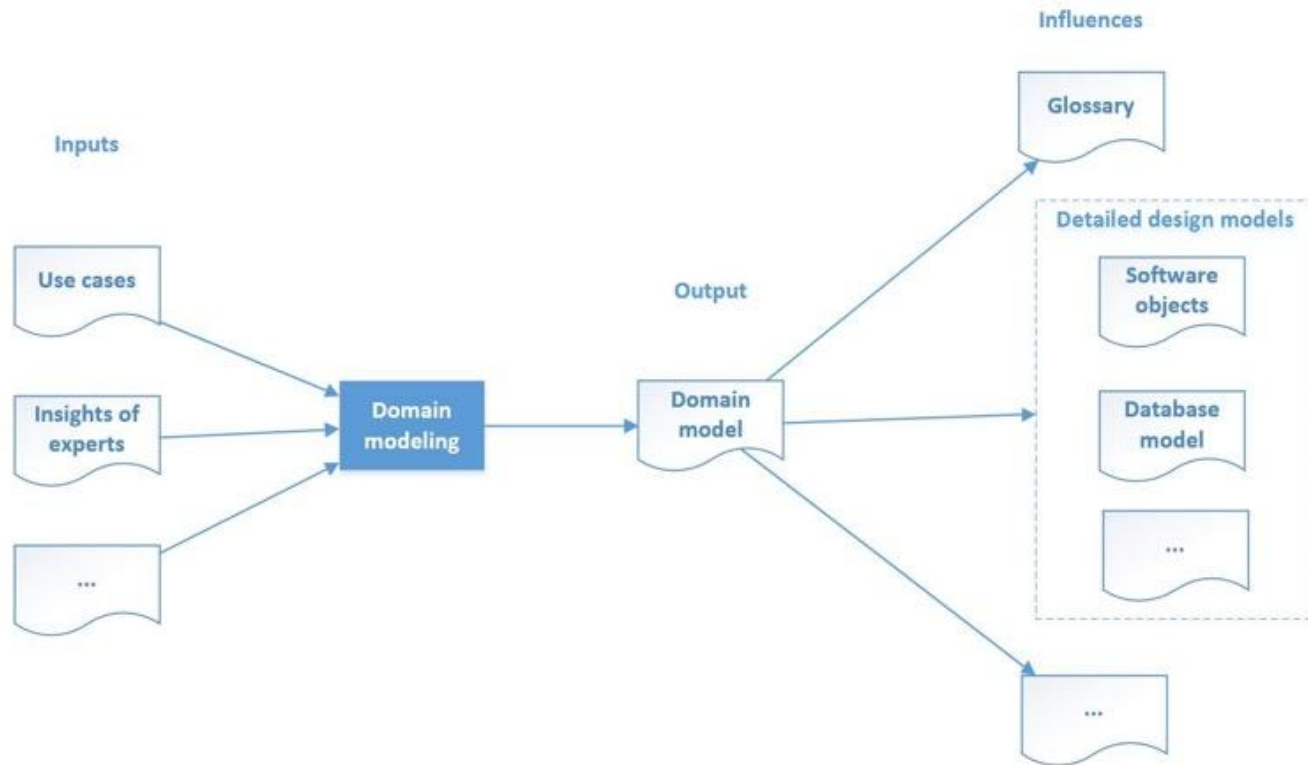
Assosiaatioiden nimet ja kardinaliteetit



Domainmalli bussiliikennöinnille



Domainmallinnus tukee muuta ohjelmistokehitystä



Tarkistuslista sanastolle

- Löytyykö kaikki avainkonseptit?
- **Onko konseptit kuvattu lyhyesti ja ytimekkäästi asiakkaiden ja käyttäjien kielellä?**
- Onko synonyymit listattu?
- Löytyykö kaikki mallissa käytetyt konseptit sanastosta?
- Ovatko konseptit substantiiveja?

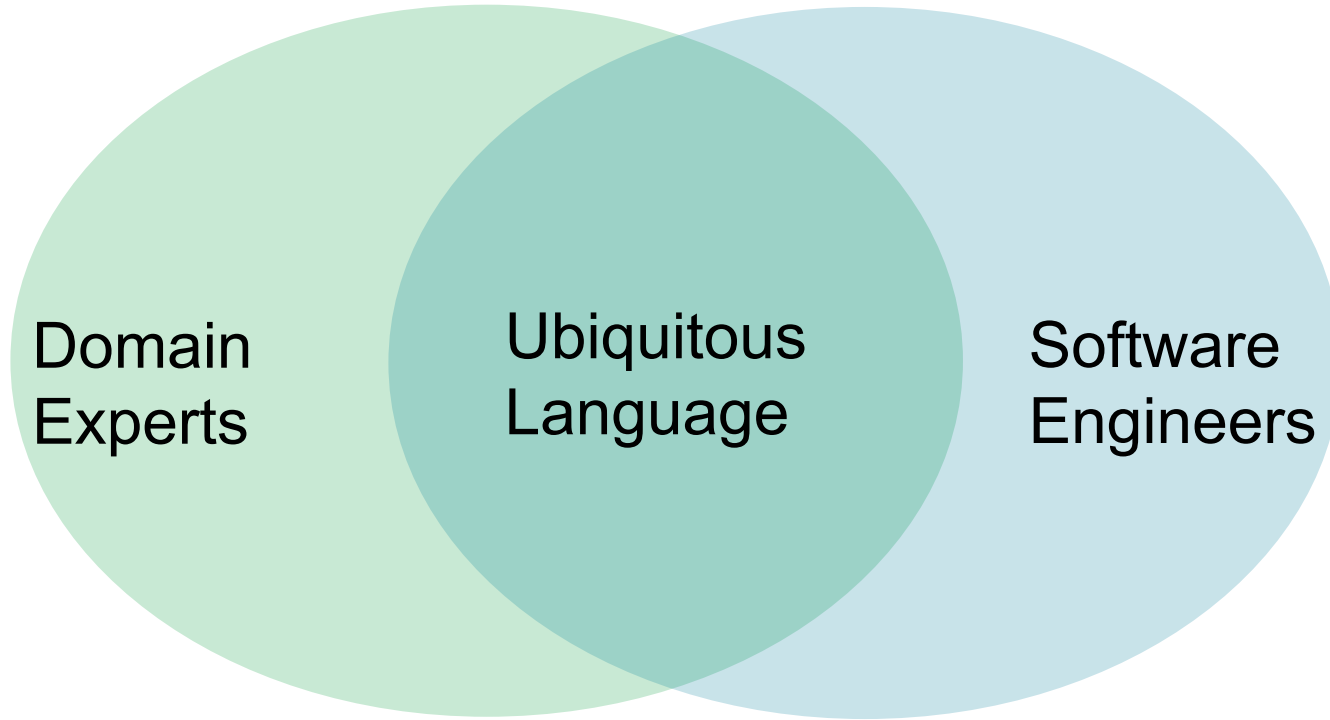
Tarkistuslista sovellusalueen malleille (1/2)

- Onko avainkonseptit mallinnettu?
- Onko kaikki tärkeät ominaisuudet tunnistettu?
- Onko tärkeät assosiaatiot kuvattu selkeästi?
- Onko malli selkeä ja helppo ymmärtää?
- Onko kardinaliteetit kuvattu?
- Onko epäselvät assosiaatiot nimetty?

Tarkistuslista sovellusalueen malleille (2/2)

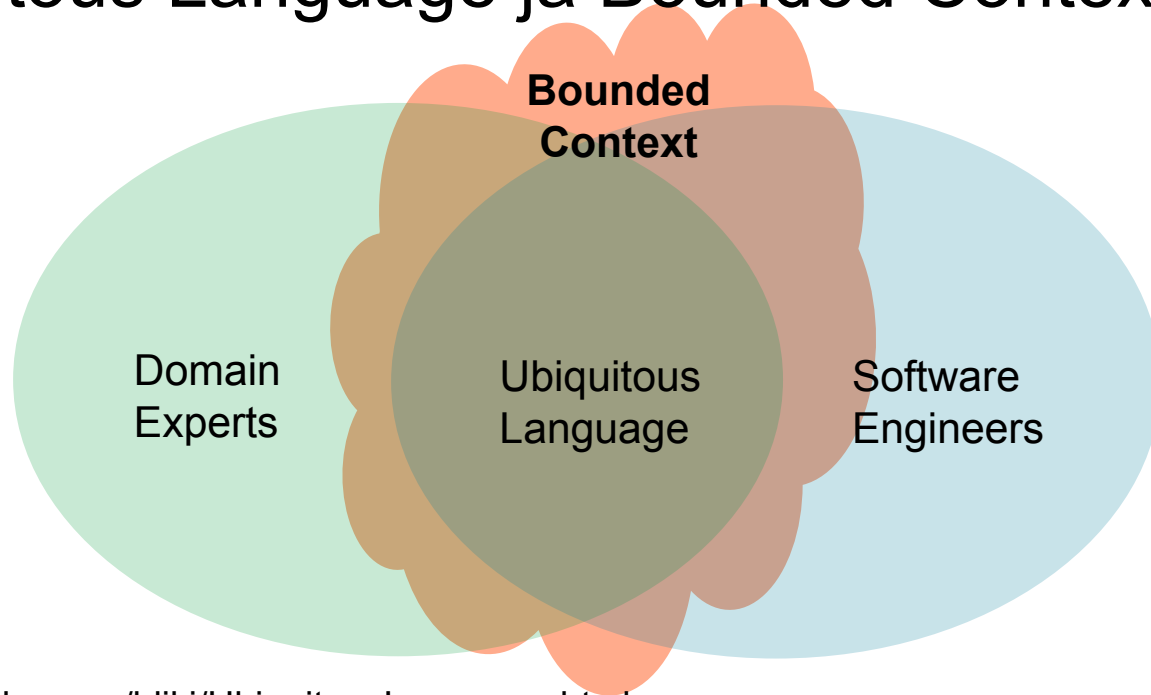
- Onko konsepteja käytetty johdonmukaisesti kaikessa dokumentoinnissa?
- Onko malli johdonmukainen sanaston kanssa?
- Onko malli johdonmukainen käyttötapauksien ja käyttäjätarinoiden kanssa?

Ubiquitous Language ja Bounded Contexts



Evans E. (2004) **Domain-driven design**: tackling complexity in the heart of software, Addison-Wesley Professional, Chapter 3, **Binding Model and Implementation**

Ubiquitous Language ja Bounded Contexts



<https://martinfowler.com/bliki/UbiquitousLanguage.html>

<https://martinfowler.com/bliki/BoundedContext.html>

Evans E. (2004) **Domain-driven design**: tackling complexity in the heart of software, Addison-Wesley Professional, Chapter 14, **Maintaining Model Integrity**

Tehtävä: Sovellusalueen malli

- Yrityksen näkökulmasta, luo:
 - Sovellusalueen malli
 - Sanasto
- Mieti uuden työntekijän kannalta - mitä olisi ollut hyvä ymmärtää kun itse aloitit
- Käydään yhdessä läpi

Yhteenvetoa päivästä

+ *seuraava päivä lyhyesti*