Ohjelmistoprosessit ja käyttöliittymäsuunnittelu

Seuraavissa kuvauksissa oletetaan, että projektissa ei ole systemaattisen käyttötilanteisiin perustuvan kälisuunnittelun osaamista. Lopuksi palataan kysymykseen, mitä tapahtuu, jos prosessimalliin lisätään käyttötilanneselvitykset ja jokin systemaattinen menettely käliratkaisujen johtamiseksi niistä.

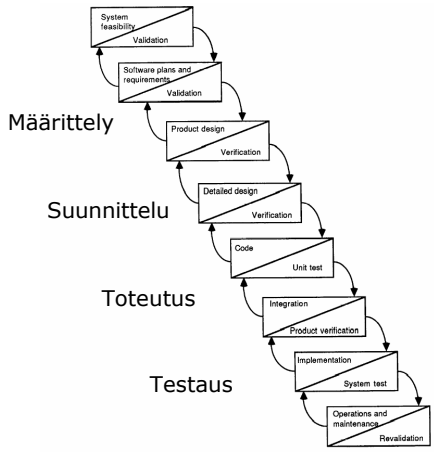
Alkuosan lähde:

Boehm B. W.,
A Spiral Model of Software Development and Enhancement.
IEEE Computer, Vol. 21, No. 5, May 1988, s. 61-72.
http://www.sce.carleton.ca/faculty/ajila/4106-5006/Spiral Model Boehm.pdf

Prosessimallit
”Koodaa ja korjaa” -malli

- Toistuvat vaiheet
  - Koodin kirjoittaminen
  - Ongelmien korjaaminen koodista
- Raportoituja ongelmia
  - Huonosti jäsennetyn koodin muuttaminen kallista
    - Korjauskierrosten tuloksena syntyvää huonosti jäsentynyttä koodia alkaa olla muutaman syklin jälkeen vaikeaa ja aikaavievää muuttaa
    - Tarvitaan *suunnitteluvaihe* ennen koodausta
  - Käyttäjiltä tulee lisää muutostarpeita koodiin
    - Koodattu toiminnallisuus ei vastaa käyttäjien tarpeita, minkä seurauksena tehtyä työtä joudutaan heittämään pois tai sitä joudutaan ainakin muuttamaan paljon
    - Tarvitaan *vaatimusanalyysivaihe* ennen suunnitteluvaihetta

Prosessimallit
Vesiputousmalli

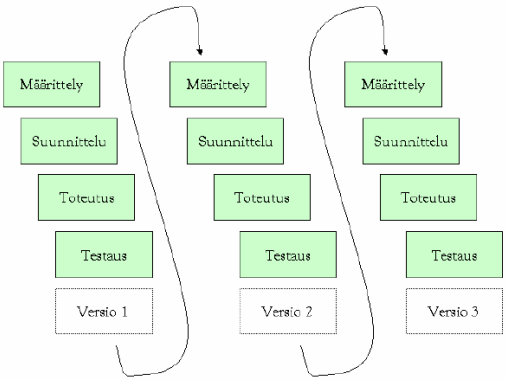


- Erityisesti interaktiivisten järjestelmien kehittämisessä ongelma: yksityiskohtaiset dokumentit vaiheiden lopputiloina.
- Suunnitteluvaiheen ongelmia:
  - Perusteellinen dokumentointi myös huonosti ymmärretyistä kohdista, esim.
    - yksityiskohtaisesti kuvatut epämääräiset käliratkaisut (esim. laatikkokaavioita, navigointikarttoja) tai
    - huonosti ymmärrettyjä toteutusratkaisuja, joista laaditaan yksityiskohtaiset toteutussuunnitelmat.
  - => Käyttökelvotonta koodia, speksit ja koodi uusiksi myöh.
  - Mitä *pitäisi* tehdä?

Määrittelyvaiheen ongelmia:

- Vaikea tuottaa kattavaa vaatimusjoukkoa.
- Tulos näkyy vasta lopuksi!

Prosessimallit
Evoluutiomalli



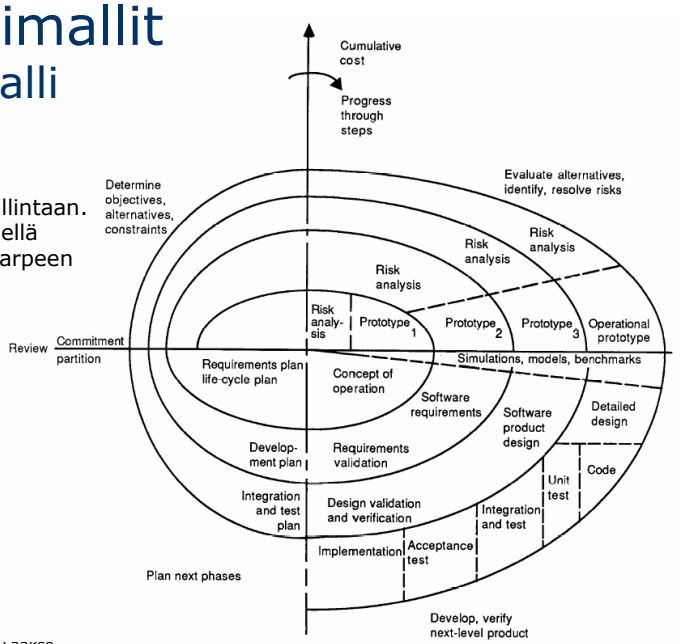
(Haikala, 1998)

- Ohjelmistotuotetta laaditaan inkrementaalisesti: aluksi tehdään osa toiminnallisuutta, jonka päälle seuraavassa vaiheessa uutta jne.
- Lähteessä mallin sanotaan sopivan tilanteisiin, joissa käyttäjät kokevat: *"I can't tell you what I want, but I'll know when I see it."* – Missä ongelma oikeasti on?
- Raportoituja ongelmia mm.
  - Pitkän tähtäimen arkkitehtuurisuunnittelu puuttuu. Uusien lisäysten seurauksena koodista alkaa tulla jäsentymätöntä.
  - Koodatun version ongelmien korjaamisesta seuraa uudelleenkoodausta, kuten vesiputousmallissakin (mutta vähemmän ja paloittain).

Prosessimallit
Spiraalimalli

(Boehm, 1988)

Painottuu riskienhallintaan. Hyödyntää mm. edellä kuvattuja malleja tarpeen mukaan.



Spiraalimalli
Vaihe 1

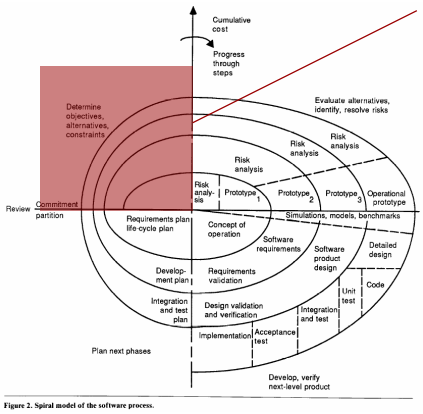


Figure 2. Spiral model of the software process.

- Jokainen spiraalin kierros alkaa määrittelemällä...
  - **tavoitteet** (esim. kaksinkertaistaa tuottavuus 5 vuodessa),
  - **ratkaisuvaihtoehdot** tavoitteen saavuttamiseksi (esim. valmiin ohjelman ostaminen, ohjelmiston kehittäminen itse, työkäytäntöjen muutos, ohjelman vaihtoehtoinen design A, design B),
  - vaihtoehtoihin liittyvät **rajoitteet** (esim. kustannukset 10 000 $ per hlö, aikataulu).

Spiraalimalli
Vaihe 2

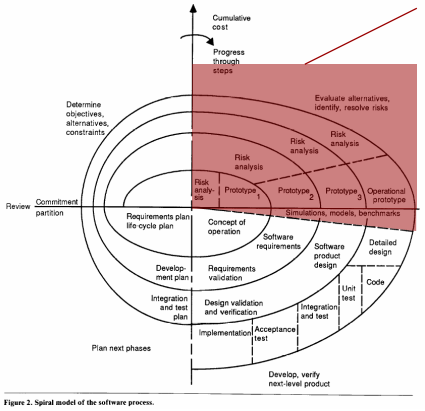


Figure 2. Spiral model of the software process.

Eräänä keskeisenä näkökohtana käyttöliittymäriskit

- Ratkaisuvaihtoehtojen arviointi
  - Riskien paikantaminen
  - Panostaminen epävarmoihin, *riskialttiisiin* kohtiin:
    - Kokematon kehittäjä keskittyy jo ennestään hyvin tuntemiensa kohtien määrittelyyn (huono tulos)
    - Kokenut kehittäjä osaa keskittyä **selvittämään epävarmoja kohtia** mahdollisimman pitkälle
  - Epävarmat kohdat saadaan ratkaistua *laatimalla prototyyppejä, simuloimalla, mallintamalla* jne.
  - Jäljellä olevat riskit määräävät, mitä seuraavassa vaiheessa (ja seuraavilla kierroksilla) tehdään, ks. seuraavan sivun taulukko.

Spiraalimalli
Jatkotoimenpiteet riskien ohjaamana

Käli- ja suorituskykyriskit dominoivat	**Budjetti- ja aikatauluriskit dominoivat**
Spiraalimalli painottuu kohti **evoluutiomallia**.	Jos käli- ja suorituskykyriskit on jo saatu hallintaan esim. aiempien prototyyppien avulla, spiraalimalli muuttuu kohti **vesiputousmallia**.
Suurimmista riskikohdista tehdään yksityiskohtaisia **prototyyppejä**; konkreettisilla kokeiluilla saadaan selville, miten kannattaa jatkaa. (Määrittelyyn ja speksaukseen minimaalisesti resursseja.)	Jokaista määrittelyä seuraa validointivaihe ja seuraavan syklin suunnitteluvaihe.
Spiraalimalliin merkityt speksien laatimisvaiheet tiedostetaan, mutta näitä vaiheita ei suoriteta.	Prototyyppien laatiminen, simulointi, mallinnus ym. riskinvähentämis-menettelyt tiedostetaan, mutta näitä vaiheita ei suoriteta (ei tarpeen).

Spiraalimalli
Vaihe 3

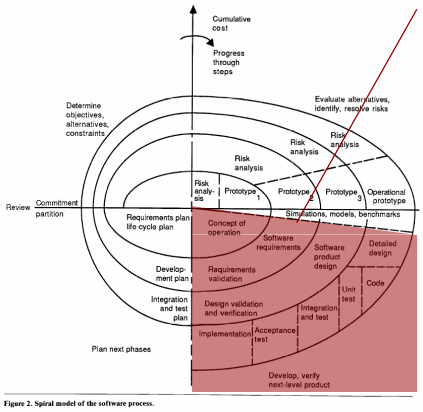


Figure 2. Spiral model of the software process.

- Jos riskejä jäi vielä jäljelle, voidaan hypätä tämän vaiheen (esim. software requirements) yli toistaiseksi ja jatkaa vaiheesta 4. Tarkoituksena on tällöin tehdä uusi riskiarviointi ja esim. prototyyppiselvitys.
- Jos tämä vaihe 3 suoritetaan, tuloksena on yhden vesiputousmallin vaiheen tuotos, kuten vaatimusmäärittely, karkea toteutussuunnitelma tms.

Spiraalimalli
Vaihe 4

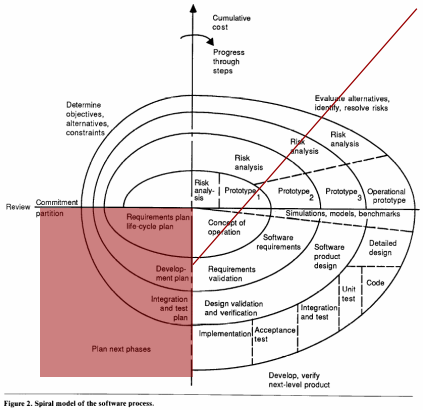


Figure 2. Spiral model of the software process.

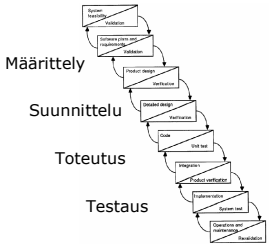
- Seuraavaa kierrosta varten
  - toimintasuunnitelma
  - resurssiarvio

Prosessimallit
Kälisuunnittelun lisääminen

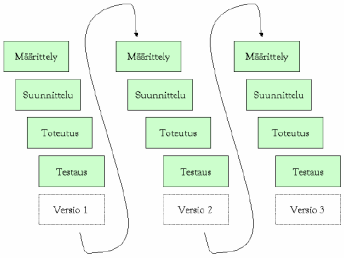
- Mitä prosessimalleille tapahtuu, jos nyt lisäämme niihin kuhunkin käyttäjien työhön perustuvan systemaattisen käyttöliittymäsuunnittelun esimerkiksi käyttämällä
  - GUIDe-mallin tavoitepohjaisia käyttötapauksia ja simulointipohjaista GDD-suunnitteluprosessia tai
  - Lauesenin tehtäväkuvauksia (tasks) ja Virtual Windows -suunnitteluprosessia.
- Selvää on, että yksi keskeinen riskivyyhti (väärä käyttöliittymä, väärä toiminnallisuus) putoaa pois.

Selvitä:
- Mihin käyttötapausselvitykset ja kälisuunnittelu sijoittuvat?
- Mitä prosessimallin muille osille tapahtuu?

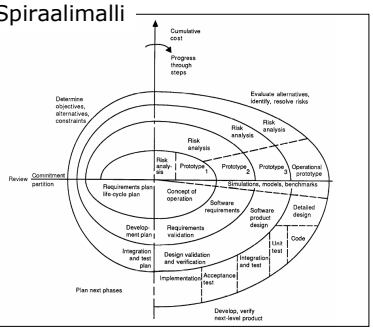
Vesiputousmalli



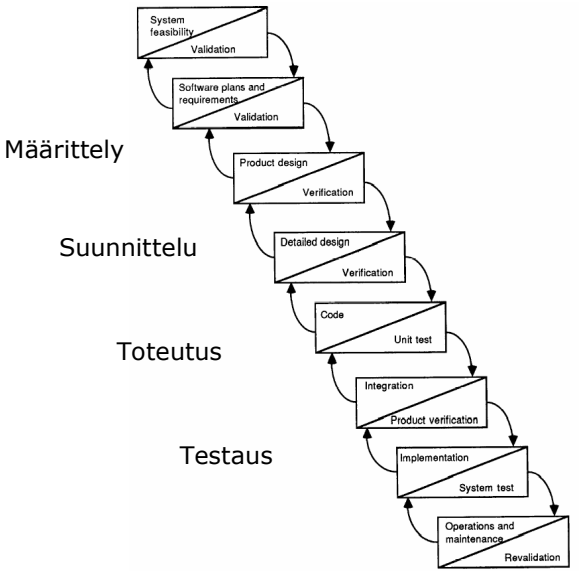
Evoluutiomalli



Spiraalimalli



Vesiputousmalli



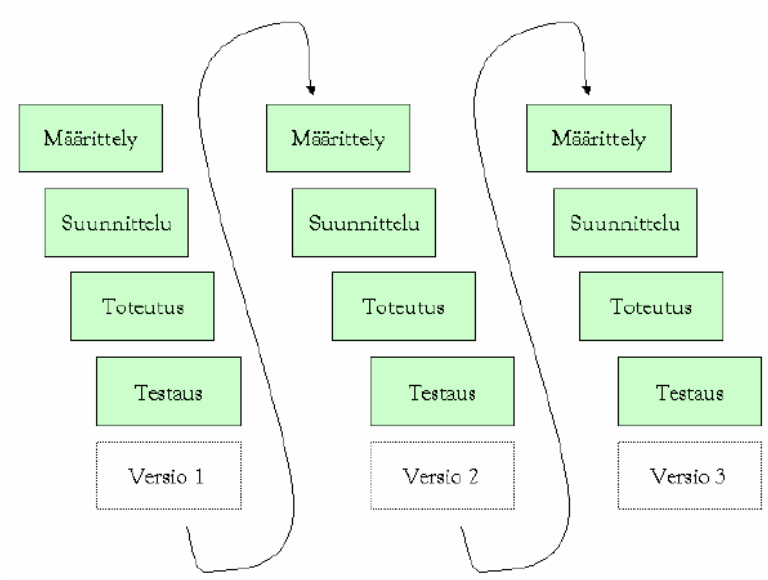
Määrittely

Suunnittelu

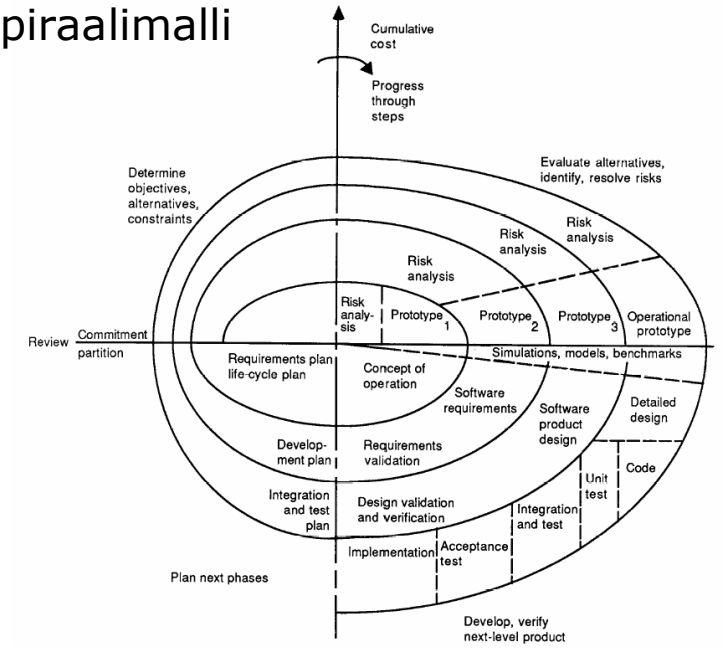
Toteutus

Testaus

Evoluutiomalli



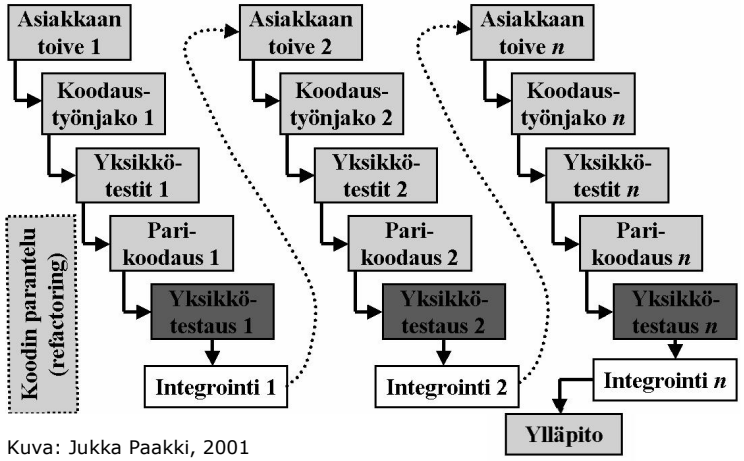
Spiraalimalli



Vielä yksi prosessimalli:
Extreme Programming (XP)

- XP:n lähtöoletus: Koska asiakkailta tulee aina paljon muutospyyntöjä projektin kuluessa, projektissa on varauduttava jatkuvaan vaatimusten muuttumiseen.
- Ratkaisu: Nopeatempoiset koodaussyklit, jotka tuottavat kokonaisia testattuja ohjelman versioita.
- Parin viikon pituiset koodaussyklit:
  - Asiakas kirjoittaa käyttäjätarinoita, jotka ovat sekä
    - toteutuksen lähtökohtana toimivia vaatimuksia että
    - hyväksymistestauksen testitapauksia.
  - Ohjelmoijat pilkkovat käyttäjätarinan ohjelmointitehtäviksi, jakavat työt ja kirjoittavat testit; koodaavat ja testaavat.
  - Koodattu ja testattu versio annetaan asiakkaalle, jolta koodaajat saavat palautetta.

Extreme Programming (XP)
Vaiheet



Kuva: Jukka Paakki, 2001

XP:n käyttäjätarinat (user stories)
Esimerkkejä 1/2

Esimerkkejä käyttäjätarinoista. Sovelluksena on asiakasprojekteihin liittyviä projektitietoja ja yhteystietoja käsittelevä järjestelmä.

- User opens "My Companies" window and sees all the companies that he is assigned to. Then user clicks a company and sees projects of his that are for that company. This would replace the "My Projects" window.
- When a user creates a project, he selects a project "type" or template that creates a default set of action items for the project.
- User sets an action item's priority.
- Only people who can see a project can see a project's action items.

http://c2.com/cgi/wiki?AtsUserStories

XP:n käyttäjätarinat (user stories)
Esimerkkejä 2/2

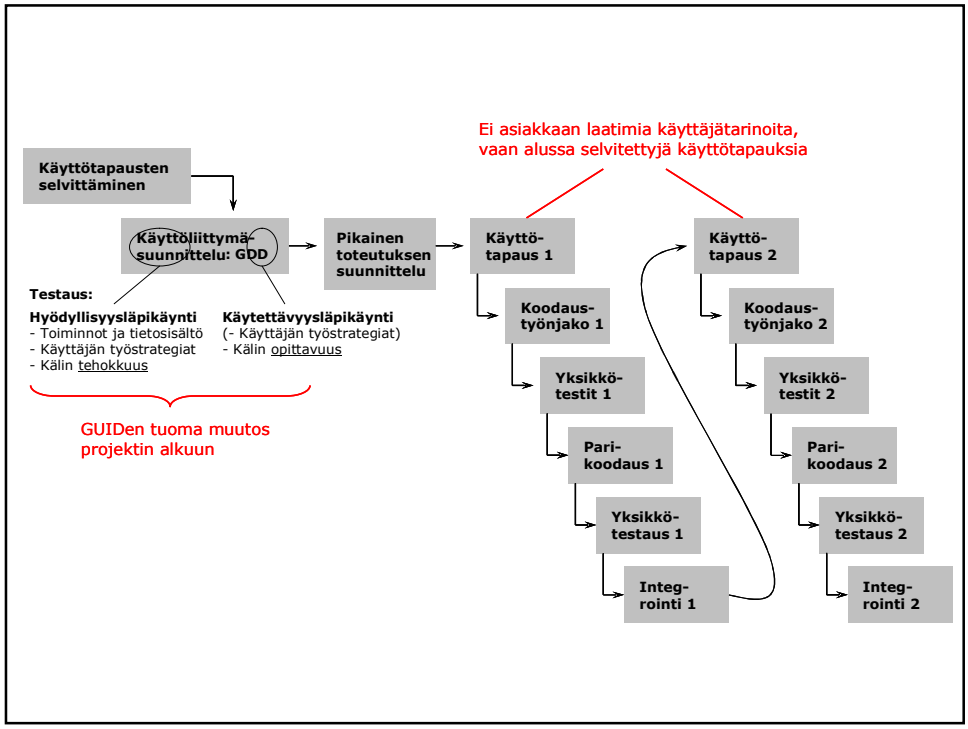
- A user marks a project as being owned by his group. (Question: can a user ever belong to more than one group?)
- Provide a box that will display the application name and version number for ATS, as well as tech. support contact info.
- A user wants access to the system, so he finds an ATS system administrator, who enters in the user's first name, last name, middle initial, email address, username (unique), and phone number. Users may be marked as "deleted." Users may also be marked as administrators.

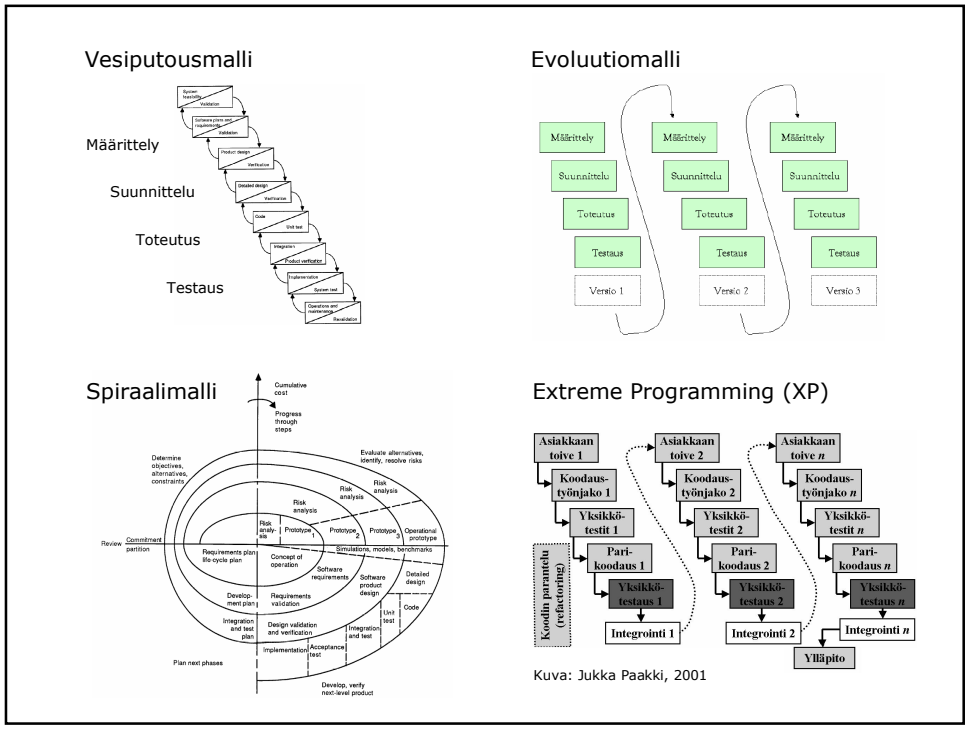
http://c2.com/cgi/wiki?AtsUserStories

Extreme Programming (XP)
Kälisuunnittelun lisäämisen vaikutus?

- Mitkä ovat XP:n keskeisimmät ongelmat?
- Toimisiko XP, jos
  - korvaisimme asiakkaiden kirjoittamat käyttäjätarinat käyttöliittymäsuunnittelijoiden laatimilla **käyttötilanteilla** tai **tavoitepohjaisilla käyttötapauksilla** ja
  - käyttöliittymäsuunnittelija laatisi toteutettavalle käyttötapaukselle **käliratkaisun**, ennen kuin koodaajat jakaisivat käyttötapauksen osiin, laatisivat testit ja koodaisivat ko. toiminnallisuuden?

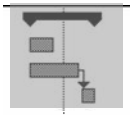
  Ts. voisiko kälisuunnittelun jälkeen pelkkä toteutusvaihe edetä XP-syklien mukaisesti?





Kuva: Jukka Paakki, 2001

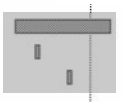
GUIDe-malli
Suurimmat jäljellä olevat riskit?



Tavoitepohjaisten **käyttötapausten selvittäminen** kenttätutkimuksilla

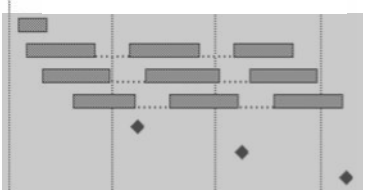
Toimintojen, tietosisällön ja interaktioratkaisujen generointi

Käyttöliittymän piirtäminen
GDD-simulointisuunnittelulla:
Käyttöliittymän piirtäminen näkyviin
=> käyttötapausten aukot esille
Paperiproton läpikäyntitestaukset



Hyödyllisyysläpikäynti
- Toiminnot ja tietosisältö
- Käyttäjän työstrategiat
- Kälin tehokkuus

Käytettävyysläpikäynti
(- Käyttäjän työstrategiat)
- Kälin opittavuus



Toteutus
- Toteutuksen suunnittelu
- Toteutus
- Testaus

Suurimmat riskit alkavat olla täällä...?

GUIDe-malli
Dokumentaatio