TUNTIMUISTIINPANOT

Perinteinen vesiputousmalli

On vaiheellinen tuotanto prosessi jossa suunnittelu ja toteutus eteen vaiheellisesti alaspäin.

Ongelmat:

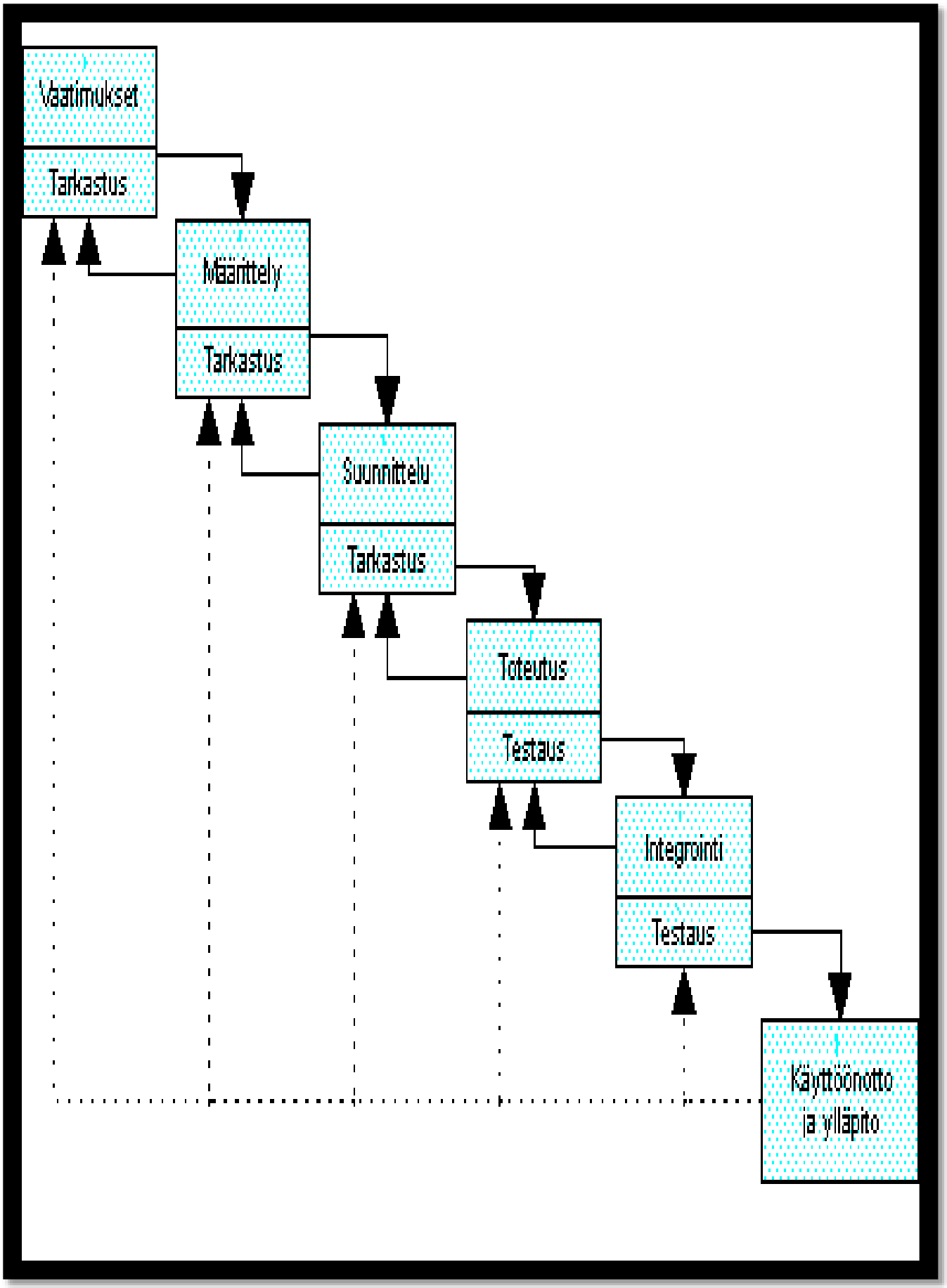
- Suurempien ohjelmistojen jokaista vaihetta
on mahdotonta suunnitella etukäteen
- Joustamaton
- Vaikea soveltaa
- Edetään vaihe kerrallaan
- Vaikea korjata virheet
- Viivästyvät tulokset

Hyödyt:

- Ohjelmiston alussa kattava suunnitelma säästää
myöhemmissä vaiheissa paljon rahaa ja aikaa.
- Helppo hallita
- Toimii hyvin pienissä projekteissa

Soveltaminen koulussa:

Kouluprojekteissa
Kurssisuunnitelmissa


Vaatimukset
Tarkastus
Määrittely
Tarkastus
Suunnittelu
Tarkastus
Toteutus
Testaus
Integrointi
Testaus
Käyttöönotto
ja ylläpito

Ketterän kehityksen yleiset piirteet

Ketterän kehityksen maailmassa lähdetään siitä, että ei ole olemassa yhtä oikeaa tapaa saavuttaa haluttu lopputulos. Senpä vuoksi harvat ketterät menetelmät ovat tarkoin määriteltyjä ohjeistoja siitä, miten missäkin tilanteessa pitää toimia. Pakollisten käytäntöjen sijaan manifestin taustalla on kokoelma periaatteita, joita noudattamalla uskotaan päästävän hyvään lopputulokseen.

4 tyypillistä arvoa:

1. Yksilöitä ja vuorovaikutusta
2. Toimivaa sovellusta
3. Asiakasyhteistyötä
4. Muutokseen reagoimista

12 periaatetta

- Tärkein tavoitteemme on **tyydyttää asiakas** toimittamalla tämän tarpeet täyttäviä versioita ohjelmistosta aikaisessa vaiheessa ja säännöllisesti.
- **Otamme vastaan muuttuvat vaatimukset** myös kehityksen myöhäisessä vaiheessa. Ketterät menetelmät hyödyntävät muutosta asiakkaan kilpailukyvyn edistämiseksi.
- **Toimitamme versioita toimivasta ohjelmistosta säännöllisesti**, parin viikon tai kuukauden välein, ja suosimme lyhyempää aikaväliä.
- **Liiketoiminnan edustajien ja ohjelmistokehittäjien tulee työskennellä yhdessä** päivittäin koko projektin ajan.
- Rakennamme projektit **motivoituneiden yksilöiden** ympärille. Annamme heille puitteet ja tuen, jonka he tarvitsevat ja **luotamme** siihen, että he saavat työn tehtyä.
- Tehokkain ja toimivin tapa tiedon välittämiseksi kehitystiimille ja tiimin jäsenten kesken on **kasvokkain käytävä keskustelu**.
- **Toimiva ohjelmisto** on edistymisen ensisijainen **mittari**.
- Ketterät menetelmät kannustavat **kestävään toimintatapaan**. Hankkeen omistajien, kehittäjien ja ohjelmiston käyttäjien tulisi pystyä ylläpitämään työtahtinsa hamaan tulevaisuuteen.
- **Teknisen laadun** ja ohjelmiston hyvän rakenteen jatkuva huomiointi edesauttaa ketteryyttä.
- **Yksinkertaisuus** - tekemättä jätettävän työn maksimointi - on oleellista.
- Parhaat arkkitehtuurit, vaatimukset ja suunnitelmat syntyvät **itseorganisoituvissa tiimeissä**.
- Tiimi **tarkastelee säännöllisesti, kuinka parantaa tehokkuuttaan**, ja mukauttaa toimintaansa sen mukaisesti.

KETTERÄN KEHITTÄMISEN MALLI

XP

Työskentelytapa yhdistää molempien kokemukset ja yleensä tämä johtaa parempaa tuottavuuteen ja parantuneeseen laatuun.

- Testivetoinen kehitys
- Jatkuva integraatio
- Refaktorointi
- Koodauskonventiot
- Koodin yhteisomistajuus
- KISS-periaate

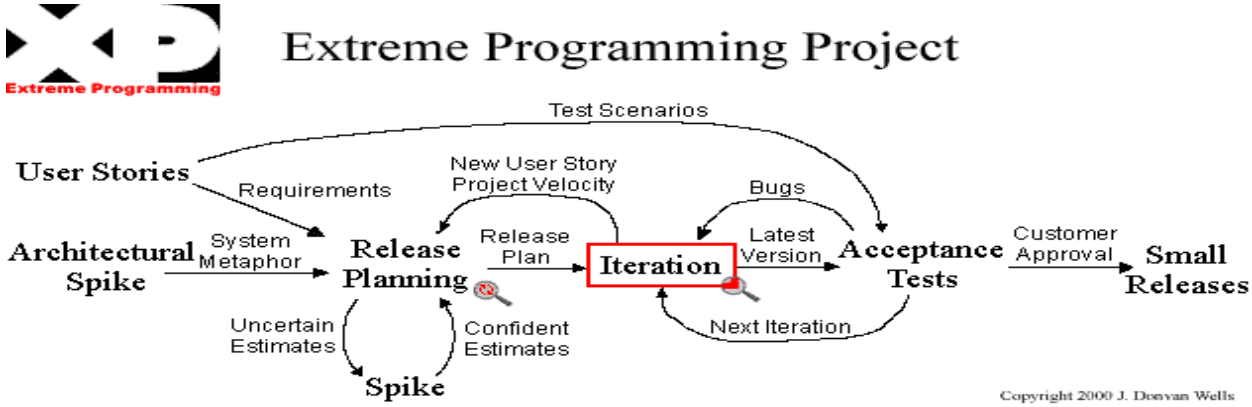
- Jokainen version täytyy olla testattu ja toimiva ennen kuin se julkistetaan
- On myös tärkeää saada asiakkaalta ja käyttäjältä palautetta ajoissa sillä mitä nopeammin tiedät ongelman sitä enemmän aikaa on korjata se.

2 henkilöä työskentelee.

Koodin yhteisomistus

Tarkoittaa sitä, että kaikki voi muokata mitä vaan,
Kuka vaan voi parantaa mitä vaan.
Ei toimi ilman hyvää kommunikointia

Tarvitsee kommunikointia
Saada kaikki ymmärtämään miten systeemi toimii.



LEAN

Koodamisessa ”jätteitä” ovat esimerkiksi turhat koodinpätkät, koodi joka joudutaan kirjoittamaan uudestaan lisäyksien takia tai poistamaan kokonaan.

LEAN; in perustana on jätteen minimointi ja prosessin tuotannon maksimointi.

PERIAATTEET

Leanin periaatteena on etsiä kaikki mahdolliset jätteet ja poistaa mahdollisimman tehokkaasti. Eliminoida turhat asiat.

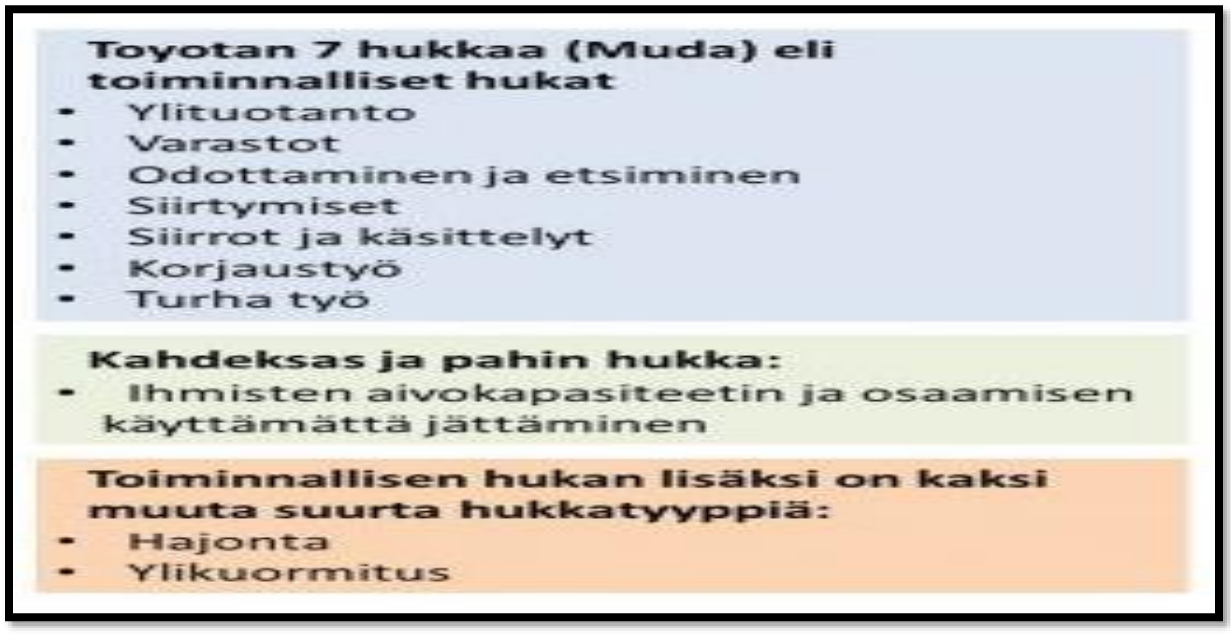
MVP

Tarkoittaa prototyyppiversiota, jossa on kaikki ominaisuudet, joita asiakas tarvitsee tuotteelta.

KANBAN

On visuaalinen työkalu jolla osoitetaan milloin tuotannon tulisi alkaa ja loppua. Varmistaa myös, että tuotannossa on riittävästi tarvikkeita ja muita työtehtäviä.

Projektiryhmä voi esimerkiksi hyödyntää Kanban:ia pysyäkseen projektissa ajan tasalla. Mm. koulu projekteissa.



TDD eli test-driven development

PARIAATTEET

Ensin luodaan uusi testitapaus, vasta sen jälkeen muokataan kehitettävää ohjelmaan niin, että se läpäisee uuden testin.

Eli yksilötestit kirjoitetaan pienissä osissa ennen varsinaista tuotantokoodilla.

Tätä pyritään parempaan rajapintasuunnitteluun sekä myös varmentumaan ohjelmiston oikeasta toiminnasta.

HYÖDYT

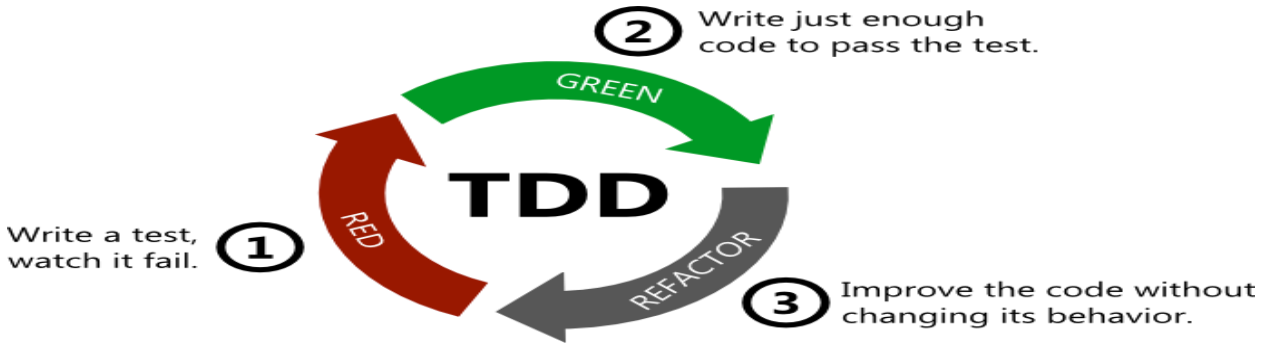
Kun testi koodi kirjoitetaan etukäteen saadaan jatkuvasti kehittyvä testiverkosto.

Sen varassa uusien toimintojen kehittäminen sekä virheiden korjaaminen on huomattavasti turvallisempaa.

Jos olemassa olevia testejä suorittamalla huomataan, jos virheitä korjatessa tulee virheitä.

MOCKING

On ykköstestausilmiö joka auttaa testaamaan objekteja erikseen toisistaan, korvaamalla riippuvaiset objektit monimutkaisella käyttäytymisellä, testiobjekteilla ja ennalta määritetyillä / simuloiduilla käytöksellä.



RUP

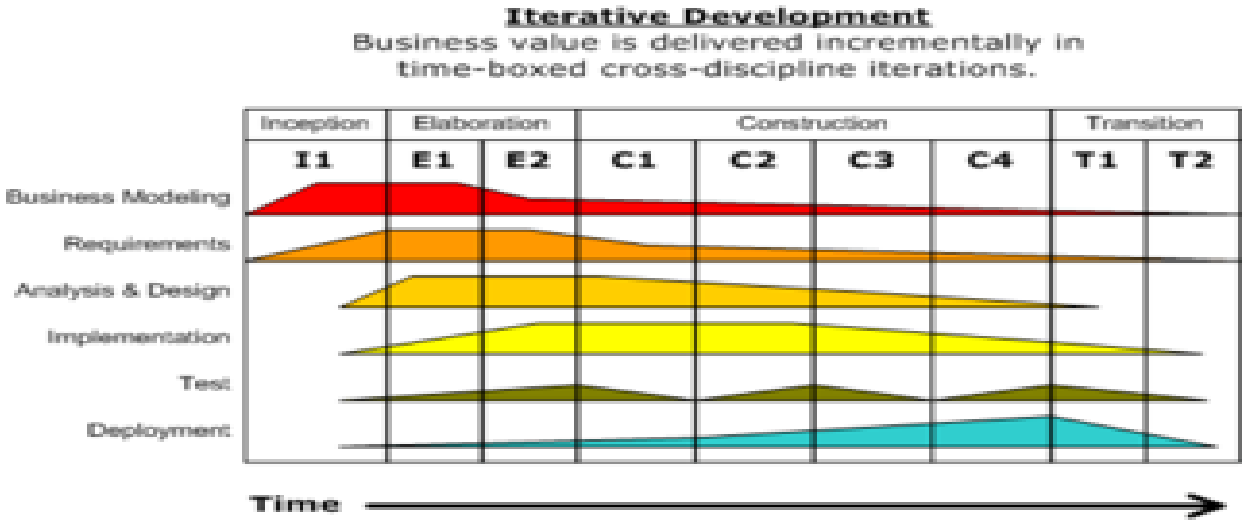
RUP on ohjelmistokehityksen prosessikehys.
Se ei itsenäinen prosessi vaan jaajennttava kehys, jota muokataan vastaamaan yrityksen tai projektin tarpeita.
RUP käytetään usein korvaavaa nimeä Unified Process, kun puhutaan yleisesti prosessista.

4 ELINKAARTA:

- voimaantulovaihe: onko järjestelmää kannattava toteuttaa
- kehittelyvaihe: pyritään vähentää riskit.
- rakennusvaihe: tavoitteena rakentaa ohjelmisto.
- muutosvaihe: tavoite saada järjestelmä tuotantoon ja tuoda se loppukäyttäjien saataville.

Selitys siitä miten ohjelmoinnin vaiheet tulee tapahtumaan.
Rakennusosat ovat:

- roolit: työntekijöille jaetaan vastuu alueet
- tuotteet: tulos, johon sisältyy kaikki prototyypit ja dokumentit
- tehtävät: jaetaan roolien mukaan.



Rup parhaat käytännöt perustuu kuuteen ideaan, joilla pyritään parantamaan laatua ja minimoimaan vikoja.

- kehitä interaktiivisesti: mieti kaikki vaatimukset etukäteen
- hallinnoi vaatimuksia: pidä käyttäjän vaatimukset aina mielessä
- käytä komponentteja.
- suunnittele visuaalisemmin: käytä kaaviota aina kun voit ja käytä suunnittelun apuna kuvia.
- valvo laatua: tee testauksista suuri osa projektia
- hallinnoin muutoksia: pidä kirjaa muutoksista

- HYÖDYT:
  Käyttää vesiputousmallista parhaimmat osat ja hyödyntää niitä
  Keskittyy dokumentoinnin tärkeyteen

- ONGELMAT;
  Raskas prosessinen.
  Hidas tiettyihin projekteihin.
  Riippuu liikaa osakkaiden palautteesta.
  Monimutkainen ymmärtää

ASD

Rakennettin kolmen ”cyclin” periaatteelle: Spekulaatio. Yhteistyö. Oppiminen.

Spekulaatiossa; koitetaan selvittää mitä asiakas haluaa ja tarvitsee.
Oletuksena se, että asiakas ei tiedä mitä hän haluaa tai tarvitsee.
Kaiken tämän informaation perusteella rakennetaan koko projektin aikataulu.

Yhteistyössä: Työmäärän tasapainoituksen haasteet ja työn suunnittelua.

Oppiminen tavoitellaan virheiden korjausta.
ASD yksi kulmakivi on: “Do it wrong the first time”.
Tämä muodostaa oppimisen.
Virheitä ei tarvitse pelätä.

HYÖDYNTÄMINEN KOULUYMPÄRISTÖSSÄ

On erittäin taipuva, sen vaatimukset on vähäiset, joten se soveltuu oppimisympäristöön. Oppiminen vaiheena antaa tutkia ongelmia ja oppia virheistä.