Systeemityö menetelmät

1. Perinteinen vesiputousmalli

- Etenee vaihe vaiheelta
- Sopii paremmin pieniin projekteihin

2. Ketterän kehityksen yleiset piirteet

- Agile Manifesto julistusta pidetään ketterän kehityksen perusmääritelmänä
- Ketterä ohjelmistokehitys on joukko ohjelmistotuotantoprojekteissa käytettäviä menetelmistöjä
- 12 periaatetta
    - asiakas on mukana aikaisessa vaiheessa ja säännöllisesti
    - http://agilemanifesto.org/iso/fi/principles.html
- 4 tyypillistä arvoa
    - Yksilöitä ja vuorovaikutusta
    - toimivaa sovellusta
    - asiakasyhteistyötä
    - muutokseen reagoimista

3. Scrum

- projektinhallinnan viitekehys
- käytetään ketterässä ohjelmistokehityksessä
- koko ryhmä pyrkii etenemään yksikkönä ja toimimaan tiiviissä yhteistyössä
- erilaisiin tilanteisiin sopeutuva, nopea ja itseohjautuva
- palaverit
    - suunnittelupalaveri
        - kestää enintään 8 tuntia
    - päiväpalaveri
        - varattu enintään 15 minuuttia
        - jäsenet kertovat mitä on tehty viime palaverin jälkeen, mitä on tarkoituksena tehdä ennen seuraavaa palaveria, työn edistyminen
- Scrumin vaiheet
    - sprintti
    - sprintin suunnittelupalaveri
    - päiväpalaveri
    - tuotteen kehitysjonon työstö
    - sprinttikatselmus
    - sprintin retrospektiivi
- roolit
    - tuoteomistaja
    - scrummaster
        - poistaa mahdolliset esteet
        - ryhmän valmentaminen
        - päivittäinen työ on tuottavaa
        - pelisääntöjen noudattaminen

        - suojaa ryhmää uusilta vaatimuksilta ja antaa työtauhan sprintin ajaksi
    - kehitystiimi
        - vastaavat sprintin valitusta tuotteen kehityspolusta ja julkaisukelvollisesta tuoteversioista
        - analyysi
        - suunnittelu
        - kehittäminen
        - testaus
        - dokumentointi

4. XP (Extreme Programming)

- painottaa muiden ketterien menetelmien tapaan mukavuutta enemmän kuin ennustettavuutta
- perustuu viiteen ydinarvoon
- suunniteltu siten, että muutoksen kustannus pysyy projektin ajan suunnilleen samana
- pienet julistukset
    - usein julkaistuja ohjelman versioita, joita julkaistaan asiakkaalle
    - jokaisen version täytyy olla testattu ja toimivaennen kuin ne julkaistaan
    - asiakkaiden ja käyttäjien palaute on tärkeä saada ajoissa, että olisi enemmän aikaa korjata ongelmat
- pariohjelmointi
    - 2 henkilöä työskentelee yhdellä koneella
    - roolit ovat ”navigoija” ja ”ohjaaja, navigoija auttaa ohjajaa antamalla hänelle vinkkejä ja sanomalla jos huomaa virheen ja ohjaaja tekee koodia ja pistää navigoijan ideat toteutukseen
- koodin yhteisomistus
    - kaikki ohjelmoijista voivat muokata mitä vaan koodilohkoista
    - ei toimi ilman hyvää kommunikointia
    - on varmistettava, että muutokset eivät aiheuta ristiriitoja muissa ohjelmiston osissa
- vertauskuva
    - tarkoitus on saada asiakas, ohjelmoijat ja esimiehet ymmärtämään, miten systeemi toimii toiminnan nimen kautta

5. LEAN

- perustana on jätteen (turhat koodinpätkät, koodi joka joudutaan kirjoittamaan uudestaan tai poistamaan kokonaan) minimointi ja prosessin tuotannon maksimointi
- periaatteena on etsiä kaikki mahdolliset jätteet ja poistaa ne mahdollisimman tehokkaasti
- MVP
    - Minimum Viable Product
    - 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
    - auttaa pysymään projektissa ajan tasalla

6. TDD (Test-driven development)

- periaate
    - ensin luodaan uusi testitapaus
    - sen jälkeen muokataan kehitettävää ohjelmaa niin, että se läpäisee uuden testin
    - yksikkötestit kirjoitetaan pienissä osissa

    - tällä pyritään parempaan rajapintasuunnitteluun sekä myös varmistumaan ohjelmiston oikeasta toiminnasta
- hyödyt
    - kun testikoodi kirjoitetaan etukäteen, saadaan jatkuvasti kehittyvä testiverkosto
    - sen varassa uusien toimintojen kehittäminen sekä virheiden korjaaminen on huomattavasti turvallisempaa
    - jo olemassa olevia testejä suorittamalla huomataan, jos virheitä korjatessa tulee tehneeksi uusia virheitä
    - TDD:tä voi hyödyntää myös kouluprojekteissa
- Mocking
    - se on yksikkötestausilmiö, joka auttaa testaamaan objekteja erikseen toisistaan korvaamalla riippuvaiset objektit monimutkaisella käyttäytymisellä, testiobjekteilla ja ennalta määritetyllä/simuloidulla käytöksellä. Näitä testiobjekteja kutsutaan Mock objekteiksi.

7. RUP

- ohjelmistokehityksen prosessikehys
- käytetään usein korvaavaa nimeä Unified Process
- ei ole itsenäinen prosessi vaan laajennettava kehys
- elinkaaret
    - voimaantulovaihe
        - Päätavoite tutkia järjestelmää ja katsotaan, onko järjestelmä kannattava toteuttaa
    - kehittelyvaihe
        - päätavoite vähentää pahimmat havaitut riskit. Projekti rupeaa ottamaan muotoaan.
    - rakennusvaihe
        - Päätavoite rakentaa ohjelmisto. Suurin osa ohjelmoinnista tapahtuu tässä vaiheessa.
    - muutosvaihe
        - Päätavoite saada järjestelmä tuotantoon ja tuoda se loppukäyttäjän saataville.
- Building Blocks
    - Roolit
        - Työntekijöille jaetaan vastuualueet osaamisen mukaan.
    - Tuotteet
        - Projektin tulos, johon sisältää kaikki prototyypit ja dokumentit
    - Tehtävät
        - Tehtävät, jotka jaetaan roolien mukaan.
- Parhaat käytännöt
    - Kehitä iteratiivisesti
        - Mieti kaikki vaatimukset etukäteen. Tähtää kustannusten minimointiin kehitysmallien avulla.
    - Hallinnoi vaatimuksia
        - Pidä käyttäjän vaatimukset aina mielessä.
    - Käytä komponentteja
        - Projektin ositus pakollista ja koodin uudelleenkäyttö on plussaa.
    - Suunnittele visuaalisemmin
        - Käytä kaavioita aina kun voi ja käytä suunnittelun apuna kuvia.
    - Valvo laatua
        - Tee testauksista suuri osa projektia.
    - Hallinnoi muutoksia
        - Pidä kirjaa tehdyistä muutoksista. Versionhallinta tärkeä osa.
- Hyödyt

    - Käyttää vesiputousmallin parhaimmat osat ja hyödyntää niitä.
    - Keskittyy dokumentoinnin tärkeyteen
- Ongelmat
    - Raskasprosessinen
    - Hidas tiettyihin projekteihin
    - Riippuu liikaa osakkaiden palautteesta.
    - Monimutkainen ymmärtää.

8. Adaptive Software Development (ASD)

- luotiin Rapid Application developmentin ja Complex Adaptive Systemsin pohjalta
- ideana oli luoda systeemi, jonka pohjana olisi jatkuva tilanteisiin ja olosuhteisiin sopeutuminen
- elinkaari
    - Spekulaatio
        - selvitetään mitä asiakas haluaa ja tarvitsee
        - sen perusteella rakennetaan koko projektin aikataulutus
    - Yhteistyö
        - Työmäärän tasapainotuksen haasteet ja työn suunnittelua.
        - Projektin adaptoituminen erilaisiin tilanteisiin ja olosuhteisiin
        - Delegointi
        - Projektin edistäminen ja työstäminen.
    - Oppiminen
        - oppimisessa tavoitellaan virheien korjausta, designing rakentamista ja testausta
        - ASD:n yksi kulmakivi on ”Do it wrong the first time”, tästä muodostuu oppiminen.
        - Virheitä eri ole tarve pelätä
    - Pohdinta
        - ASD on erittäin taipuva, sen vaatimukset ovat vähäiset, joten se soveltuu opiskeluympäristöön.
        - Oppiminen vaiheena, antaa aikaa tutkia ongelmia ja oppia virheistä, jälleen vahvistaen ASD:n potentiaalia oppimisympäristössä.