X-treme programming

12 principer of agile practice (rörlig)

- Ge nöjd kund genom tidig och kontinuerliga leveranser
  - Den viktigaste punkten som betyder att min vill ha kontinuerlig feedback
- Välkomna sena ändringar av krav.
  - Är ett måste för många företag och också ett måste för programmen

- Kontinuerligt leverera mjukvara som är under utveckling
  - Intervaller från några veckor till några månader, helst det kortare intervallet
- Verksamhets personer och utvecklare ska jobba tillsammans dagligen
  - Konstant interraktion och kommunikation är väsentlig i rörlig utveckling

- Bygg projektet kring engagerade individer, ge dem miljön och förtroendet
  - I rörlig utveckling är individerna viktigare.
- Den effektivaste metoden att få tag på information är att prata ansikte mot ansikte.
  - Är det rörligt så måste man prata med varandra!!!
  - Dokument kan skapas, men framförallt pratar man med varandra

- Fungerande mjukvara är det viktigaste måttet på hur långt man kommit
  - Man mäter hur lång man kommit med hur mycket av mjukvaran som kunde är nöjd med.
- Rörlig utveckling ska innebära en hållbar utveckling
  - Man ska kunna hålla på i samma tempo för evigt!
  - Det handlar inte om ett 100m lopp i max tempo
  - För snabbt ger fel och genvägar.

- Att kontinuerligt se till god design och bra tekniska lösningar.
  - Hög kvalitet är nyckeln till hög hastighet.
  - Att koda med hög kvalitet är ett måste för att nå fram snabbt
- Enkelhet, konsten att maximer det jobba som inte behöver göras.
  - Ta den enklaste vägen till målet, inte bygga den optimala lösningen.
  - Saker ändras så snabbt så ödsla inte tid

- Bästa arkitekturen, kraven och designen kommer från självorganiserande team
  - Alla medlemmar jobbar med alla aspekter och finner gemensamt de bästa lösningarna via en typ av själv organisering
- Med jämna intervaller reflekterar gruppen på hur man ska jobba bättre och effektivare
  - Ett rörligt team är rörligt och ska förbättras

Practices of extreme programming

- Customer Team members
  - Ha med folk från verksamheten i utvecklings teamet
  - De måste vara med, eller åtminstone NÄRA.
- User Stories
  - I XP så skriver man användar berättelser för att beskriva kraven
  - De är inte detaljerade, utan mest till för att man ska komma ihåg vad man kommit överrens om

- Korta cykler
  - Man levererar normalt mjukvara varrannan vecka
  - Varje cykel producerar exekverbar kod
- Acceptans tester
  - User stories används för att fånga acceptans tester som kunden definierar
  - Ger tester som kontinuerligt och automatiskt kan köras

- Pair programming
  - All programmering görs i par, två personer på samma dator.
  - En skriver kod, den andra övervakar och föreslår förbättringar.
  - Man byter kontinuerligt par som jobbar tillsammans
- Test-driven development
  - All kod som skrivs har sin unit test som den ska passera
  - Testet skrivs oftast först och sedan inom snar framtid skriver man själva koden.

- Collective ownership
  - Alla får check ut och ändar i alla moduler!
  - Alla jobbar med alla delar, även om man har sina specialliteter
- Continous integration
  - Kod integreras kontinuerligt varje gång man checkar in kod.
  - Man utvecklar test, sedan kod som hyfsat fungerar och checkar in innan man är klar och integrerar innan det är klart.

- Sustainable pace
  - Det är inte sprint, utan maraton som vi springer!
  - En regel är att medlemmarna INTE får jobba övertid
- Open Worksapce
  - Alla jobbar tillsammans i ett öppet rum.
  - Öppet lanskap där man hör de olika parens konversation och kan ingripa om det är problem
  - Olika UML diagram och modeller hänger runt väggarna.

- The planning game
  - Bygger på att abvgöra asnvar mellan verksamheten och utvecklings teamet.
  - Verksamheten avgör vikten av en funktion och utvecklings teamet hur mycket det kostar.
  - Inför varje iteration så ger utvecklarna en budget baserat på hur mycket de lyckades göra i förra iterationen och kunden väljer jobb som matchar detta
- Simple design
  - Gör en så enkel design som möjligt och basera den endast på de user stories som är planerade för iterationen.
  - Inför tunga saker när de behövs.

- Refactoring
  - Innebär att man ska kontinuerligt ändra och förbättra kod
  - Men detta utan att förändra övrig kod
- Metaphor
  - Använd metaforer eller bilder för att förstå och realisera problem
  - Kanske inte ett måste för XP