Krav Specificering og Vedligeholdelsesstyring.

Kravsspecificering for et produkt skal ofte foregå på et meget detaljeret plan, så det er tydeligt hvad det enkelte krav indeholder. Krav kan både danne ramme for kontrakter og er på samme tid rettesnoren for designfasen i et projekt. Krav gøres op i funktionelle og ikke-funktionelle krav. Ofte er det de ikke-funktionelle krav, der er mest kritisk at få tydeliggjorte, da de i værste fald kan gøre systemet ubrugeligt, hvis de fejler. Endeligt kan de ikke-funktionelle krav deles op i følgende tre grupper: Produkt og kvalitetskrav, organisatoriske krav og eksterne krav. Organisatoriske krav vil være bundet til, hvordan omstændighederne er i organisationen. Eksterne krav kan være eksempelvis lovgivning eller integration med andre it-produkter.

I praksis kan kravspecifikationen være et konkret dokument, men når man arbejder agilt, vil de funktionelle krav typisk være repræsenteret, via product backloggen, hvor de øverste krav vil være dem der er mest detaljeret beskrevet. Da agile metoder er modstander af formel dokumentation, kan man sige at product backloggen fungerer, som en slags inkrementel dokumentation, hvilket passer fint med, at man hen gennem den agile process, løbende kommer til at stå med nye versioner af product backloggen. At specificere krav er både komplekst og tidskrævende. Der kan spille mange forhold ind, der komplicerer specificeringen af krav. Disse forhold kan blandt andet være lovgivning, ændringer i kundebehov, underleverandørers ændringer i underprodukter og endelig kan det være, at kunderne ikke er sikre på hvad de præcist vil have, eller at de ikke præcist formår at give udtryk for det, i relevante termer der er it-faglige eller it-relaterede.

Agile metoder er baseret på løbende kravændringer, og derfor skelnes der ikke mellem nyudvikling og videreudvikling, som også kaldes vedligehold. Praktisk idriftsættelse af ny funktionalitet er det kunden oplever som en ny version af produktet. Typisk sker videreudvikling ved, at der enten tilføjes komponenter, eller laves ændringer i eksisterende komponenter. Grunden til vedligeholdelse er vigtigt er, at det oftest vil være for dyrt og risikabelt, at implementere nye systemer til at erstatte de gamle, frem for at udvikle på det der allerede er integreret. I forbindelse med vedligehold er der ofte en række forhold, der har indflydelse på omkostningerne for vedligeholdelse. Det er afgørende om der er stabilitet omkring teamet forstået på den måde, at hvis vedligeholdelsen overgår til nye udviklere, vil der gå viden og effektivitet tabt. Udvikleres erfaringsniveau har også betydning, da det kan afgøre, hvor meget domæneviden de har indenfor systemets område. Endvidere kan det være, at systemet ikke fra starten er designet optimalt til at blive ændret på. Endeligt kan der være tale om, at systemet er skrevet i legacy-sprog, eller der kan mangle dokumentation, eller der kan være manglende konsistens i koden.

Når man videreudvikler på et system, vil man typisk foretrække re-engineering, da det er billigere end at udvikle et helt nyt system. Et eksempel på en moderne metode der faciliterer re-engineering, er extreme programming, som netop taler om refactoring, som en af sine 12 praktikker. Dette kunne være et argument for at integrere extreme programming i sin udviklingspraksis.