Innhold

[1. Innledning 2](#_Toc385238892)

[2. Forutsetninger 2](#_Toc385238893)

[3. Gjennomføring av leveransen 4](#_Toc385238894)

[3.1 Organisering 4](#_Toc385238895)

[3.2. Etablering av Utviklings- og Testmiljøer 5](#_Toc385238896)

[3.3 Kravspesifisering 6](#_Toc385238897)

[3.3.1 Funksjonelle krav 6](#_Toc385238898)

[3.3.2 Ikke-funksjonelle krav 7](#_Toc385238899)

[3.3.3 Øvrige leveranseelementer 7](#_Toc385238900)

[3.4 Leveranseplan og Teststrategi 7](#_Toc385238901)

[3.5 Delleveranser 8](#_Toc385238902)

[3.5.1 Programvareutviklingen 8](#_Toc385238903)

[3.6 Test 10](#_Toc385238904)

[3.7 Utprøving 10](#_Toc385238905)

[3.8 Overlevering og Produksjonssetting 11](#_Toc385238906)

[4 Endringshåndtering 11](#_Toc385238907)

[4.1 Endringer relatert til Øvrige leveranseelementer 11](#_Toc385238908)

[4.2 Endringer relatert til den Smidige programvareutviklingen 11](#_Toc385238909)

[5. Vederlag og bonus 12](#_Toc385238910)

[5.1 Vederlag for programvareutvikling og test 12](#_Toc385238911)

[5.2 Bonusordningen 13](#_Toc385238912)

[5.2.2 Ulike bonustyper 13](#_Toc385238913)

[5.2.3 Bonus som utbetales underveis 14](#_Toc385238914)

[5.2.5 Sluttbonus 14](#_Toc385238915)

[5.2.6 Eksempel på bruk av bonus 14](#_Toc385238916)

[6. Vedlikeholdsavtale 16](#_Toc385238917)

[7. Dagbot 16](#_Toc385238918)

[7.1 Dagbot ved forsinkelse i programvareutviklingsleveransen 16](#_Toc385238919)

[7.2 Dagbot ved forsinkelse i Øvrige leveranselementer 16](#_Toc385238920)

[8. Begrepsforklaringer 17](#_Toc385238921)

# Innledning

Smidigavtalen (SSA-S) er beregnet på større utviklingsanskaffelser hvor det skal benyttes smidige programvareutviklingsmetoder. Det er en omfattende avtale som krever mye av Kunden både når det gjelder kompetanse og ressursinnsats. Avtalen skiller seg fra de andre programvareutviklings-avtalene i SSA-porteføljen (SSA-U og SSA-T), bl.a. ved at kontrakten inngås tidligere i systemutviklingsprosessen, se figuren nedenfor:

**SSA-S**



**SSA-T**

**SSA-U**

Mike Cohn: Agile Estimating and Planning, Prentice Hall 2006

***Figuren illustrerer når i systemutviklingsprosessen det inngås kontrakt for henholdsvis SSA-S, SSA-U og SSA-T***

# Forutsetninger

Smidigavtalen er en omfattende avtale som krever mye av Kunden både når det gjelder kompetanse og ressursinnsats. Det ligger et større ansvar på Kunden for at leveransen kommer i havn enn i de tradisjonelle utviklingsavtalene i SSA-porteføljen. Kunden har ansvaret for det funksjonelle omfanget (scope) mens Leverandøren har ansvaret for kvaliteten (Ikke-funksjonelle krav). Dersom Kunden øker omfanget, øker normalt også prisen. Leverandøren har altså ikke ansvar for omfang. Han skal i korte trekk levere, og få betalt for, det Kunden ber om. Leverandøren har imidlertid blant annet et ansvar for å varsle Kunden dersom Kundens ønsker vil gå ut over avtalte krav til kvalitet og resultere i at andre Ikke-funksjonelle krav ikke vil kunne bli oppfylt.

**Ressurser:** Programvare-
utvikling på løpende timer
(Kan ha andre vederlagsmodeller for Øvrige leveranseelementer)

**Leverandør:** Ansvar for
Ikke-funksjonelle krav/kvalitet

**Kunde:** Ansvar for omfang (scope)

***Figuren illustrerer ansvar for kvalitet og omfang i smidigavtalen. Dersom omfang øker, øker også prisen. Dette er i så fall Kundens ansvar.***

Det er generelt ønskelig med stor grad av fleksibilitet i anskaffelse av smidig programvareutvikling. Samtidig er det et mål å ha mest mulig forutsigbarhet. Det er derfor viktig at Kunden spesifiser i konkurransegrunnlaget alt Kunden har mulighet for å avklare på forhånd, spesielt innenfor de områdene hvor Leverandøren har ansvar, ref. over. Det gjelder for eksempel teknisk plattform og andre tekniske rammebetingelser for leveransen, krav til arkitektur, sikkerhet, ytelse osv. Kunden må også ha kompetanse til å kunne verifisere at disse kravene er oppfylt i leveransen. Dette betyr at Kunden enten må ha denne kompetansen internt, eller leie den inn i hele avtaleperioden. Det samme gjelder testkompetanse.

Avtalen har separate regimer for henholdsvis Funksjonelle og Ikke-funksjonelle krav.

* **Funksjonelle krav** skal være beskrevet på et overordnet nivå i form av mål for hva programvaren skal gjøre og beskrivelser av hvilke behov den skal oppfylle. Det er Kundens ansvar hvis omfanget øker underveis i avtaleperioden. Viser det seg at noen av behovene (Funksjonelle krav) er mer krevende å oppfylle enn opprinnelig antatt i én Delleveranse, må Kunden som hovedregel ta ut et annet Funksjonelt krav. Det er derfor viktig at det som er viktigst for Kunden blir utviklet først.

Kunden bør ha en Behovshaver/Produkteier som har myndighet til å ta beslutninger på vegne av Kunden når det gjelder Funksjonelle krav. Dette er ofte krevende i offentlige virksomheter. Et eksempel kan være tilfeller hvor Programvaren som utvikles inneholder tolking av regelverk. Spesielt utfordrende er det da hvis regelverket ikke er implementert enda, eller er under revisjon. I et slikt tilfelle bør Kunden sørge for god kontakt med de som utvikler regelverket for å unngå for mye forsinkelser i utviklingsprosjektet på grunn av utsatte beslutninger knyttet til selve regelverksarbeidet.

* **Ikke-funksjonelle krav** skal være spesifisert som krav i konkurransegrunnlaget. De Ikke-funksjonelle kravene er som nevnt Leverandørens ansvar. Leverandøren skal varsle Kunden dersom han tar beslutninger i utviklingen av brukerfunksjonalitet som vil gjøre at ikke-funksjonelle krav ikke kan oppfylles.

Hvis Kunden ikke har kompetanse internt til å spesifisere de Ikke-funksjonelle kravene, bør Kunden leie inn ekstern bistand fra tredjepart (Avtalens punkt 6.2). Eksempler på slike kompetanseområder er arkitektur, sikkerhet og test. Slik kompetanse er nødvendig gjennom hele kontraktsperioden. Det er fordelaktig at de som leies inn bistår Kunden helt fra utarbeidelse av konkurransegrunnlaget og frem til Programvaren er levert og akseptert.

# Gjennomføring av leveransen

Etter kontraktsinngåelse skal Kunde og Leverandør samarbeide om å omdanne Behovsbeskrivelsen til krav, som så skal detaljeres og være utgangspunkt for den smidige programvareutviklingen. Programvareutviklingen skal etter denne avtalens oppbygning være delt opp i Delleveranser, og ellers foregå etter den Smidige programvareutviklingsmetoden som er beskrevet i bilag 6 og. Det er således viktig at Leverandøren beskriver Programvareutviklingsmetoden i tilbudet, eventuelt ut fra føringer som Kunden har gitt i bilag 1.

I forberedelsesfasen skal prosjektorganisasjonen etableres, de fysiske rammene rundt prosjektet skal på plass (lokaler, felles verktøy for metodestøtte, Utviklings- og Testmiljø osv) og de første versjonene av planer og andre styrende dokumenter skal utarbeides.

Avtalen skiller grunnleggende mellom programvareutvikling og Øvrige leveranseelementer. Øvrige leveranseelementer kan omfatte opplæring, etablering og levering av Utviklings- og Testmiljøer, rutineutvikling m.m. Disse skal leveres i henhold til prosjekt- og milepælsplanen etter avtalt vederlag på samme måte som i andre SSA-er.

## 3.1 Organisering

Det eneste som er regulert i selve avtaleteksten om organisering er at Kunde og Leverandør skal ha hver sin prosjektleder som organisatorisk er ansvarlig for at partenes forpliktelser oppfylles i henhold til avtalen. Øvrige deler av leverandørens prosjektorganisasjon skal være beskrevet i bilag 4.

Hovedoppgaven til Kundens og Leverandørens prosjektleder vil være å koordinere de ulike prosessene i avtalen og å følge opp fremdrift og økonomi.

Programvareutvikling vil utgjøre en stor del av avtalen. Kunden bør derfor ha en dedikert person til å følge programvareutviklingen, koordinere prioriteringer, sørge for avklaringer og til å ta beslutninger når det gjelder krav. I prosjekter basert på Scrum-metodikk, vil dette være Produkteier .

Leverandørens prosjektorganisasjon skal fremgå av tilbudet. Smidig programvareutvikling kjennetegnes ved at utviklingen gjøres av selvstyrte, tverrfunksjonelle (”crossfunctional”) team. Grunnen til at teamene bør være tverrfunksjonelle er at det gir større fleksibilitet, enn hvis teamet er avhengig av å hente inn spisskompetanse utenfor teamet, eller hvis noen av teammedlemmene kun har én spisskompetanse og ikke kan brukes til andre oppgaver. Dersom utviklingen er basert på Scrum, vil teamet ha en Scrum-master som legger til rette for teamets arbeid og fasiliteter kommunikasjonen mellom produkteier og utviklingsteamet eller teamene.

Testing er viktig i smidig Programvareutvikling. Både Kunde og Leverandør bør ha dedikerte testledere. Leverandørens testleder skal legge til rette for Leverandørens testing og samarbeide med Kundens testleder i forbindelse med Kundens Akseptansetest, oppfølging av feil med mer.

Kundens testleder skal utarbeide tester og testbeskrivelser parallelt med programvareutviklingen, administrere Akseptansetester og følge opp feil. Se mer om test i pkt. 3.4 og 3.6 nedenfor.

For å øke utviklingskapasiteten kan Leverandøren tilby flere utviklingsteam. Kunden bør i så fall vurdere om han har kapasitet til å ivareta sitt ansvar dersom utviklingskapasiteten økes. I mange smidigprosjekter har tilgang på nøkkelkompetanse hos Kunden vært et problem for fremdriften i prosjektet. I smidigavtalen kan Leverandøren kreve at Kunden stanser prosjektet dersom Kunden ikke følger opp sine forpliktelser for eksempel når det gjelder avklaringer og beslutninger (Avtalens punkt 6.1).

## 3.2. Etablering av Utviklings- og Testmiljøer

Det skal fremgå av konkurransegrunnlaget hvem som har ansvar for etablering og drift av Utviklings- og Testmiljøer. Frist for etablering av Utviklingsmiljø skal fremgå av prosjekt- og milepælsplan. Frist for etablering av Testmiljøer skal fremgå av Leveransplanen. Med mindre både Utvikling og Test foregår på Leverandørens utstyr, vil det være et grensesnitt mellom Leverandør og Kunde og eventuelt også Kundens driftsleverandør. Dette er et område som erfaringsmessig skaper en del utfordringer i gjennomføringen av anskaffelsesprosjektet:

I PERFORM-prosjektet i NAV - det største smidig-prosjektet som har vært gjennomført i norsk offentlig sektor - var det å få satt opp effektive og behovstilpassede utviklings- og testmiljøer, samt koordinering mellom linje, eksterne parter og prosjektet noe som ble fremhevet som vanskelig og tidkrevende og det ble nevnt som to av ti problemområder som hemmet produktiviteten i prosjektet. **(Ref. Hannay & Benestad 2010: Perceived productivity threats in large agile software development projects, ESEM2010☺)**

Det er derfor viktig at rammebetingelser, herunder rutiner for samarbeid mellom prosjekt, linje og eventuelle eksterne parter (driftsleverandør), blir godt og entydig definert/beskrevet i konkurransegrunnlaget.

## 3.3 Kravspesifisering

Avtalen skiller som nevnt grunnleggende mellom Programvareutvikling (grønt område) og Øvrige leveranseelementer (blå og lilla). Programvareutvikling er igjen delt inn i Funksjonelle krav (tilsvarer Behovsbeskrivelsen, se øverste grønne felt) og Ikke-funksjonelle krav (se svart felt under grønt).

Figuren nedenfor illustrerer de ulike elementene i leveransen:



***Figuren illustrerer de ulike leveranseelementene i smidigavtalen.***

### 3.3.1 Funksjonelle krav

En av forskjellene i denne avtalen sammenlignet med SSA-U og SSA-T er at Kunden og Leverandøren skal samarbeide om å spesifisere Funksjonelle krav ut fra formåls- og behovsbeskrivelsen i konkurransegrunnlaget. Behovsbeskrivelsen kan for eksempel være gitt i form av Brukerhistorier og Epos. Forskjellen på en Brukerhistorie og et Epos er størrelsen. Et Epos er en stor Brukerhistorie. Både Epos og Brukerhistorier har formen:

*Som en <rolle>, ønsker jeg å <behov>,
(slik at <hensikt/fordel>)”.*

***Eksempler på Brukerhistorier:***

*”****Som*** *saksbehandler* ***ønsker jeg å*** *legge inn informasjon om en sak* ***slik at*** *jeg kan slippe å registrere denne informasjonen på nytt hver gang det kommer inn et nytt dokument på saken.”*

*”****Som*** *linjeleder* ***ønsker jeg*** *å kunne ta ut rapporter over hvilke saker mine medarbeidere jobber med* ***slik at*** *jeg kan holde oversikt over hvem som jobber med hva”*

*”****Som*** *prosjektleder* ***ønsker jeg*** *å få oversikt over alle timelister de innleide konsulentene har sendt siste måned* ***slik at*** *jeg kan holde oversikt over hvor mange timer som har påløpt”*

Alle roller som inngår i Brukerhistoriene må være definerte:

***Eksempler på rolledefinisjoner****:*

***En saksbehandler*** *er en bruker internt i virksomheten som …*

***En linjeleder*** *er en leder som det fremgår av organisasjonsstrukturen at har personalansvar for et gitt antall ansatte. En linjeleder kan ha følgende tittel: Direktør, assisterende direktør, avdelingsdirektør, seksjonssjef, underdirektør, Fagdirektør regnes ikke som linjeleder fordi de ikke har personalansvar*

***En Prosjektleder*** *er ….*

Dersom programvareutviklingen er basert på scrum, vil de Funksjonelle kravene ligge i en produktkø (product backlog) som Kundens produkteier senere skal prioritere før oppstart av hver sprint.

### 3.3.2 Ikke-funksjonelle krav

Konkurransegrunnlaget skal også inneholde krav til Programvaren som skal utvikles, såkalte Ikke-funksjonelle krav. Disse kravene går på egenskaper ved den programvaren, som for eksempel Pålitelighet, Effektivitet eller Vedlikeholdbarhet. I SSA-S er det Leverandørens ansvar å oppfylle disse kravene til den Programvaren som blir utviklet. Leverandøren har for øvrig plikt til å varsle Kunden dersom måten Kunden detaljerer og konkretiserer de Funksjonelle kravene på vil kunne føre til at noen av de Ikke-funksjonelle kravene ikke kan eller vil bli oppfylt som avtalt.

### 3.3.3 Øvrige leveranseelementer

Utover dette vil det også være krav i konkurransegrunnlaget knyttet til Øvrige leveranseelementer. Eksempler på slike krav kan være krav til opplæring, rutineutvikling, Testmiljøer, bistand ved Prodiksjonssetting og lignende

## 3.4 Leveranseplan og Teststrategi

Kunden skal lage en overordnet prosjekt- og milepælsplan som del av konkurransegrunnlaget. Leveransen skal deles opp i Delleveranser (såkalte «releaser»). Rammene for oppdeling av leveransen i Delleveranser kan delvis være gitt av prosjekt- og milepælsplanen, men bør sannsynligvis justeres når Kunde og Leverandør etter hvert – gjennom spesifiseringsarbeidet som nevnt over - har fått en felles forståelse av hva de funksjonelle kravene innebærer. Les mer om Delleveranser i punkt 3.5 nedenfor.

I forberedelsesfasen skal det utarbeides en Leveranseplan som omfatter alle deler av leveransen, herunder både programvareutvikling og Øvrige leveranseelementer. Leveranseplanen skal være detaljert for den aller første delleveransen og mer overordnet for de kommende Delleveransene. Leveranseplanen skal videre vise sammenhengen mellom de ulike leveranseelementene (ref. kakediagrammet over) i hver enkelt Delleveranse, og revideres i forbindelse med detaljplanlegging av hver Delleveranse.

Leverandørens Teststrategi skal være en del av tilbudet. Den skal beskrive hvordan Leverandøren skal teste programvaren han skal utvikle. Kunden kan eventuelt stille krav til hvordan Teststrategien skal utformes i konkurransegrunnlaget, beskrive sin testing, og stille krav til Leverandøren sin testing.

Kunde og Leverandør skal etter signering, samarbeide om å lage en omforent Teststrategi som beskriver både Kundens og Leverandørens testing, bruken av Testmiljøene, Testdata og Testbrukere, samt rutiner for gjennomføring av testene, feilrapportering, Retesting med mer. Teststrategien skal revideres ved behov.

## 3.5 Delleveranser

Programvareleveransen består av én eller flere Delleveranser, se figuren nedenfor:

###

***Figuren illustrerer Innholdet i Delleveransene og sammenhengen mellom Delleveransene og Øvrige leveranser i smidigavtalen.***

Delleveranser skal bestå av én eller flere iterasjoner, også kalt utviklingssykluser eller sprinter. Når iterasjonen planlegges skal Kundens behovseier/produkteier være tilgjengelig for å gjøre avklaringer og å fatte beslutninger relatert til kravene. Kunden skal definere Godkjenningskriterier som definerer hva som skal til for at kravene skal være oppfylt.

Dersom Kunden ikke har definert Godkjenningskriterier for oppfyllelse av kravene til en funksjon, så kan Kunden heller ikke senere underkjenne funksjonen, dersom behovsbeskrivelsen i henhold til konkurransegrunnlaget er oppfylt.

Dersom Kunden ikke evner, rekker eller av andre grunner ikke har gjort nødvendige avklaringer og fattet beslutninger fort nok, vil det kunne føre til forsinkelser i programvareutviklingen. Dette vil i utgangspunktet være Kundens ansvar, og vil kunne bli betraktet som mislighold fra Kundens side

### 3.5.1 Programvareutviklingen

Utviklingen av programvaren skal foregå i utviklingssykluser eller sprinter. Nedenfor har vi beskrevet hvordan en slik utviklingssyklus/sprint kan foregå dersom Programvareutviklingsmetoden (som skal være beskrevet i bilag 6) er basert på Scrum. Resultatet av kravspesifiseringen beskrevet i kapittel 3.3 vil da være en initiell Produktkø hvor elementene er utformet som Brukerhistorier. Programvareutviklingen vil foregå i utviklingssykluser som kalles Sprinter i Scrum. En Sprint skal ha en fastsatt lengde (gjerne 3-4 uker).

Det skal gjennomføres planleggingsmøte ved oppstart av hver Sprint. Leverandørens Scrum-master leder møtet.

Før planleggingsmøtet skal Produkteier ha prioritert Produktkøen og utviklingsteamet skal ha estimert omtrentlig hvor mye av Produktkøen som teamet vil kunne utvikle i neste utviklingsperiode. På planleggingsmøtet skal Produkteier presentere mer funksjonalitet (flere brukerhistorier) enn det utviklingsteamet har kapasitet til, slik at teamet kan velge ut de elementene som gir dem mest mulig effektiv utvikling.

Produkteier skal bistå utviklingsteamet med utdypende forklaringer av elementene i Produktkøen når utviklingsteamet velger ut og estimerer det de ønsker å ha med i Sprintkøen (liste over det som skal utvikles i Sprinten).

Når Sprintkøen er besluttet skal utviklingsteamet lage en oversikt over i hvilke perioder det er behov for at Produkteier deltar på morgenmøtene og i eventuelle andre aktiviteter

Det skal også avtales tid for avslutningsmøte med demonstrasjon av det som er blitt utviklet og oppsummering.

Under gjennomføringen av Sprinten skal utviklingsteamet skal ha morgenmøte (Stand up) hver dag. Produkteier skal delta på de morgenmøter og andre aktiviteter som er avtalt på oppstartsmøte. Produkteier skal i tillegg være tilgjengelig for avklaringer gjennom hele Sprinten og kunne delta på morgenmøtene dersom det skulle vise seg likevel å være behov for det.

Utviklingsteamet er ansvarlig for helheten i det som utvikles. Dersom Produkteier ønsker å få utviklet funksjonalitet som går på tvers av hensynet til helheten i løsningen, har utviklingsteamet ansvaret for å gjøres oppmerksom på konsekvensene. I hht Avtalen har Leverandøren (Utviklingsteamet) ansvar for at det som utvikles tilfredsstiller de Ikke-funksjonelle kravene i Bilag 1. Dersom Produkteier opprettholder ønsket, skal dette håndteres som en endring etter Avtalens kapittel 3.

Kunden (Produkteier) skal definere Godkjenningskriterier for all funksjonalitet i Brukerhistoriene. De skal brukes i forbindelse med planlegging og gjennomføring av Akseptansetesten. Kunden kan ikke introdusere nye Godkjenningskriterier etter at Brukerhistorien er utviklet.

Når Sprintperioden er ferdig skal det gjennomføres et avslutningsmøte som ledes av Scrummaster. Møtet omfatter en demonstrasjon av det som er blitt utviklet i løpet av Sprinten. Dersom ikke alt som var planlagt er blitt ferdigstilt, skal det inngå i grunnlaget for planleggingen av neste Sprint sammen med det som ligger i Produktkøen.

Produkteier skal kvittere ut eller forkaste det som blir demonstrert. Produkteier kan kun forkaste funksjonalitet som ikke tilfredsstiller oppsatte Godkjenningskriterier. Dersom Produkteier likevel ønsker at løsningen skal endres, skal dette håndteres som en endringsanmodning i hht kapittel 3 i Avtalen.

Som del av avslutningsmøtet bør det også gjennomføres Retrospektiv, som er en strukturert prosess for å samle gode og dårlige erfaringer fra Sprinten.

## 3.6 Test

Avtalen punkt 2.3.2 stiller krav til at Leverandøren skal utføre Komponent-, Integrasjons- og Systemtest av Programvaren før den leveres til Kunden for Akseptansetesting. Leverandøren skal videre godtgjøre at Programvaren tilfredsstiller Startkriteriene for Akseptansetesten.

Mange Leverandører vil kunne oppleve disse kravene som strenge i en konkurransesituasjon. Det vil imidlertid lette gjennomføringen av leveransen dersom feil blir luket ut så tidlig som mulig. Dette punktet om at Leverandøren skal utføre Komponent-, Integrasjons- og Systemtest bør derfor vurderes nøye før man eventuelt beslutter å endre Avtalen på dette punktet til bruk i forbindelse med en konkret anskaffelse.

Leverandøren skal etter avtalen teste Delleveransen før den overleveres til Kundens Akseptansetest, samt godtgjøre at den ikke inneholder flere feil enn det som er beskrevet i Startkriteriene for Akseptansetesten. Kunden skal få Testmateriale fra Leverandørens testing slik at det kan gjenbrukes i Akseptansetesten. Leverandøren skal også gi Kunden beskjed om hvilke Ikke-funksjonelle krav som kan testes.

Kundens testleder skal utarbeide en plan for hva som skal testes i akseptansetesten samt testbeskrivelser/testscenarier. Dersom det er avtalt Ikke-funksjonelle krav som krever kompetanse som Kunden ikke har, kan Kunden benytte tredjepart til å gjennomføre testen.

## 3.7 Utprøving

Hensikten med Utprøvingen er at brukere (gjerne fremtidige brukere av programvare) kan bruke Programvaren og komme med tilbakemeldinger. Utprøvingen kan foregå som en utvidet test, som en pilot eller i ordinær drift. Dersom det kommer forslag til endringer, så skal de endringshåndteres og konsekvensene ved å implementere dem skal utredes. Kunden kan velge om endringene skal utføres istedenfor å utvikle annen funksjonalitet, eller om de skal komme som tillegg, se kapittel 5.2 nedenfor.

Det kan være behov for Øvrige leveranseelementer, for eksempel opplæring i forbindelse med Utprøvingen. Det skal i så fall fremgå av Leveranseplanen.

## 3.8 Overlevering og Produksjonssetting

Når Kunden har akseptert en Delleveranse skal Leverandøren Overlevere Programvaren og eventuelle Øvrige leveranseelementer til Kunden på den måten som er avtalt i bilag 4.

Delleveransene skal Akseptansetestes, Utprøves og Produksjonsettes etterhvert, hvis ikke forhold hos Kunden gjør at dette ikke er hensiktsmessig (f.eks. at systemet skal støtte et regelverk som trår i kraft en bestemt dato). I så fall aksepteres Delleveransene etterhvert, men Produksjonsettes samlet.

Fristen for Produksjonssetting skal fremgå av prosjekt og milepælsplanen. Godkjenningsperioden løper i 3 måneder etter (hver) Produksjonssetting.

# Endringshåndtering

Endringshåndtering i SSA-S håndteres forskjellig avhengig av om det gjelder endringer i den Smidige programvareutviklingen (5.2) eller i Øvrige leveranseelementer (5.1):

## 4.1 Endringer relatert til Øvrige leveranseelementer

Endringer relatert til Øvrige leveranseelementer skal alltid håndteres som endringer iht. Avtalens kap. 3, og vil ikke avvike fra endringshåndtering i de andre SSA-ene. Eksempler på slike endringer er:

* Kunden ønsker å kjøpe flere kurs
* Kunden ønsker å få satt opp ekstra testmiljøer
* Kunden ønsker at Leverandøren skal gjennomføre en installasjon av programvarens som Kunden opprinnelig hadde sagt at de ville gjøre selv

## 4.2 Endringer relatert til den Smidige programvareutviklingen

I Smidig programvareutviklingsmetode håndteres endringer (knyttet til utviklingen av Programvare) ved at de prioriteres på lik linje med andre krav og behov. Det er derfor ikke relevant å snakke om endringshåndtering i den delen av avtalen som dreier seg om utvikling av Programvaren.

I smidigavtalen er det imidlertid lagt inn en formalisert aksept av Delleveransene, slik at Kundens eventuelle endringsønsker (omgjøringer, tillegg etc.) etter at Kunden i Delleveransen har testet og akseptert Programvaren, skal håndteres etter bestemmelsene i kapittel 3. Leverandøren skal utrede konsekvensene av endringen og lage et endringsoverslag og får vederlag for dettesom beskrevet i Avtalens punkt 3.3. tredje avsnitt.

Gjennomføringen av endringen vil som hovedregel gjøres i den Smidige programvareutviklings-prosessen, og Leverandøren skal derfor ikke ha særskilt vederlag for dette. Kunden kan imidlertid velge å få utført endringen som et tilleggsarbeid, og da skal Leverandøren ha vederlag for utførelse av endingen. Hvis Kunden har bestemt at endringen skal gjøres som et tillegg, kan Kunden, dersom det er bonus som ikke er blitt utbetalt, velge å utbetale bonus for leveranse av programvare med få feil (se kapittel 6.2.3 nedenfor) til endringen.

Alle programvareendringer skal Akseptansetestes på lik linje med annen kode som utvikles. Dersom endringene skjer etter siste delleveranse i avslutningsfasen (Avtalens kapittel 2.6 Avslutning) vil det ikke være noen Utprøving av endringen.

# Vederlag og bonus

Det er lagt opp til fleksibilitet i avtalen ved at det kan avtales forskjellige vederlagsmodeller for ulike leveranseelementer. Utgangspunktet er at alle priser i Avtalen oppgis i norske kroner og inklusiv merverdiavgift. Annet må spesifiseres særskilt. Det må oppgis hvorvidt prisen er per enhet eller for eksempel per måned/år/avtaleperioden.

Leveranseelementene i avtalen kan grovt deles inn i følgende aktiviteter:

* Forberedelsesaktiviteter,
* Programvareutvikling, samt
* Øvrige leveranseelementer (oppsett av miljøer, systemintegrasjon, konvertering, opplæring, dokumentasjon og eventuelt annet).

Vederlag for leveranseelementer i bilag 1 og 2 kan baseres på en eller flere av følgende vederlagsmodeller:

* Løpende timer for prosjektleder, testleder og timebasert konsulentbistand, evt. med tak
* Løpende teampris (timepris for hele utviklingsteamet inkl. eventuell Scrummaster)
* Enhetspris, fastpris, målpris eller løpende timer, evt. med tak, for spesifiserte leveranseelementer

## 5.1 Vederlag for programvareutvikling og test

Avtalen legger opp til å benytte løpende teampris for Leverandørens aktiviteter knyttet til:

* programvareutvikling, inklusive Leverandørens testing (Komponent- Integrasjons og Systemtest),
* konfigurasjonsstyring,
* planleggings- og avslutningsmøter samt
* utvikling og vedlikehold av løsningsspesifikasjon med Kundens Godkjenningskriterier.

Testleder kan være en del av teamprisen, eller betales på løpende timer uavhengig av teamet.

Begrunnelsen for å prise teamet som et hele er at det bygger opp under betydningen av at teamene er tverrfunksjonelle eller «crossfunctional», se kapittel 3.2 Organisering, ovenfor, som vektlegges tungt i smidige utviklingsmetodikker. Teamprisen kan gjerne være angitt som teamukeverk (vederlag for hele teamet per uke). Det må være definert hva et teamukeverk omfatter og hva slags kortere fravær blant teammedlemmer som kan kompenseres ved merarbeid. Det bør videre være beskrevet mekanismer for å håndtere de økonomiske sidene ved Leverandørens ansvar for uplanlagt fravær som sykdom eller liknende, utskifting av teammedlemmer ol.

Kunden skal ikke betale full teampris, dersom det er fravær i teamet som ikke kompenseres fullt ut i form av merarbeid eller supplerende ressurser. Det bør innarbeides mekanismer som hindrer eller begrenser Leverandørens mulighet til å ta ut medarbeidere av teamet for å bruke dem på prosjekter som er viktigere for Leverandøren.

Avtalen legger ikke opp til at feilretting av programvareutviklingen håndteres som en egen prosess, uten vederlag, som i de andre SSAene. Men, dersom feilene ikke blir rettet etter tre runder med retesting, sier ordlyden i Avtalens punkt 2.3.4 Stans ved vedvarende feil, at utviklingsteamet ikke får vederlag for sitt arbeid frem til feilene er rettet.

##

## 5.2 Bonusordningen

Formålet med bonusordningen er å motvirke utfordringer ved prismodellen. En prismodell basert på løpende timer, enten det er teampris eller vanlig timepris har den bi-effekt at det på mange måter ”lønner seg” for Leverandøren å oppmuntre Kunden til å øke omfanget på kontrakten. Bonusen skal virke som motvekt til dette og være et insentiv for Leverandøren til å forventningsstyre Kunden mot ”godt nok”.

Bonusordningen må således ikke sammenblandes med mislighold av kontrakt. Fravær av retten til bonus i et gitt tilfelle er ikke det samme som sanksjonering av et mislighold. Dersom Leverandøren for eksempel blir ansvarlig for en mangel eller forsinkelse og samtidig ikke har gjort seg fortjent til bonus, så må ikke dette betraktes som dobbeltstraff eller -sanksjonering.

Bonus kommer som et tillegg til ordinært vederlag og beregnes ut fra Estimert totalkost. Det innebærer at størrelsen på bonus avhenger av størrelsen på Estimert totalkost. Jo høyere Estimert totalkost, dess høyere bonusgrunnlag. Fordi bonusgrunnlaget er basert på Estimert totalkost, reduserer bonus risikoen for taktisk underestimering.

### 5.2.2 Ulike bonustyper

Det er to hovedtyper bonus; én sluttbonus og én bonus som utbetales underveis. Det bør være oppgitt i konkurransegrunnlaget hvor stor del av bonusgrunnlaget som utbetales underveis og hvor mye som benyttes til sluttbonus.

Som grunnlag for fordelingen bør Kunden ha gjort en risikoanalyse og på den måten ha en oversikt over hva som er kritiske suksessfaktorer for leveransen. Dersom leveransen gjelder et system som skal støtte et regelverk som trer i kraft en bestemt dato og derfor ikke kan settes i produksjon før hele systemet er ferdig, kan det være fornuftig å holde igjen en stor andel av bonusgrunnlaget til sluttbonusen. Dersom det er mulig å ta i bruk systemet etter hvert, bør en større del settes av til bonus som utbetales underveis.

### 5.2.3 Bonus som utbetales underveis

Det kan være fornuftig å dele opp bonusgrunnlaget for den bonus som utbetales underveis i bonuspoeng. På den måten er det enkelt å variere størrelsen på bonusen ut fra omfanget på den enkelte Delleveranse og kanskje også ut fra hvor kritisk den er.

Bonusutbetalingen basert på opptjening av bonuspoeng kan fungere som en fleksibel ordning, hvor ikke alt er forhåndsbestemt. Kunden kan for eksempel velge å la bonuspoeng som ikke ble utdelt fordi Leverandøren ikke tilfredsstilte kriteriene, gå inn igjen i potten og brukes som et ekstra insentiv mot slutten av leveransen.

Bonusene som betales ut underveis er knyttet til opptjening av bonuspoeng for god kvalitet på programvareutviklingen. Leverandøren får full bonuspoengopptjening dersom det som er utviklet tilfredsstiller akseptansekriteriene uten feilretting og halv bonusopptjening dersom akseptansekriteriene er oppfylt etter én feilrettingsrunde. Bonus tilsvarende opptjente bonuspoeng bør utbetales ved aksept av hver enkelt Delleveranse.

Dersom det er bonuspoeng «til overs», dvs bonuspoeng som ikke er blitt utbetalt fordi Leverandøren ikke har kvalifisert seg for det i tidligere Delleveranser, kan Kunden velge å dele ut bonuspoeng for programvareendringer etter bestemmelsene i kapittel 3I så fall bør dette fremgår av endringsordren. Bonuspoengene bør som hovedregel utbetales når den enkelte Delleveranse eller endring er Testet og Akseptert.

### 5.2.5 Sluttbonus

Sluttbonusen utbetales dersom Leverandøren leverer på estimert tid og/eller tidligere. Bonusutbetalingen kan reguleres slik at den blir høyere jo tidligere leveransen kommer. Sluttbonus kan gjerne utbetales på Leveransedag, men partene kan avtale andre utbetalingstidspunkter.

### 5.2.6 Eksempel på bruk av bonus

*I konkurransegrunnlaget er det oppgitt at 10% av estimert totalkost skal benyttes til bonus.*

*Hva som er den beste måten å fordele bonus på, vil avhenge fra anskaffelse til anskaffelse. Dersom det er mange involvert i testing og utprøving, vil det være lurt å legge mesteparten av bonusen der. Omfattende tidsbruk til retesting og forsinkede utprøvingsperioder som involverer store deler av organisasjonen, vil gi store kostnader internt gjør dette til en fornuftig måte å fordele bonusen på.*

*Dersom tidspunkt for leveranse er kritisk, f.eks. fordi programvaren skal støtte et regelverk som trer i kraft på en bestemt dato, vil det være fornuftig å bruke en større andel av bonus på sluttbonus for leveranse før eller på tid.*

*I dette eksemplet har Kunden bestemt at 6% skal benyttes til bonus underveis, 4 % skal benyttes til sluttbonus. Bonus underveis skal gis i form av bonuspoeng. I forbindelse med Leveranseplanleggingen blir partene enige om at programvareutviklingen skal deles opp i 4 Delleveranser i tillegg til Avslutning. Bonus underveis blir delt opp i 120 bonuspoeng og de blir fordelt som følger: Første Delleveranse: 30 bonuspoeng, de tre påfølgende Delleveransene 20 bonuspoeng hver, Avslutning 30 bonuspoeng.*

*Leverandøren som fikk kontrakt har angitt Estimert totalkost for prosjektet til 20 millioner kroner. 10% av dette skal utgjøre bonusgrunnlaget. Det gir et samlet bonusgrunnlag på 2 millioner kroner. Dette deles opp slik at 1,2 millioner skal brukes til bonus som deles ut underveis og 800 000 kr skal benytte til sluttbonusene.*

***Fordeling av bonusen som ble delt ut underveis:****De 1,2 millionene som utgjør bonusgrunnlaget for denne type bonus, deles i 120 bonuspoeng. Hvert bonuspoeng blir da verd 10 000 kr. Ved første Delleveranse vil det da bli delt ut 300 000 kr i bonus dersom Kunden aksepterer uten feilretting. Dersom det blir nødvendig med én feilrettingsrunde med påfølgende retesting, vil bonusen bli 150 000 kr.*

*I eksempelet nedenfor forestiller vi oss at programvareutviklingsleveransene foregikk på følgende måte:*

* *I første Delleveranse ble det nødvendig med én feilrettingsrunde. Leverandøren fikk dermed utbetalt halv bonus. Det betyr at 15 bonuspoeng ikke ble betalt ut.*
* *Neste Delleveranse kunne aksepteres uten feilretting og Leverandøren fikk full bonusutbetaling. Påfølgende Delleveranse var imidlertid full av feil og det ble ingen bonusutbetaling.*
* *I forbindelse med utprøvingen av denne Delleveransen kom det opp mange endringsønsker. Kunden ønsket å få utført dem i tillegg til den funksjonaliteten som var planlagt i siste Delleveranse. Som insentiv for å få Leverandøren til å levere med få feil, ønsker Kunden å gi 25 av de bonuspoengene som ikke var betalt ut som bonus.*
* *Leverandøren klarte å levere endringene med så god kvalitet at det ble full bonus. Den siste Delleveransen ble krevde én feilrettingsrunde og ga følgelig halv bonus. Avslutningen kunne aksepteres uten feilretting og ga 30 bonuspoeng. Det betyr at 100 av de 120 bonuspoengene ble utbetalt, dvs 1 million kroner.*

***Graderingen av sluttbonusen****Bonus for å levere til estimert tid eller tidligere, er gradert som følger:*

*Leveranse 1 måned før estimert: 100 %, dvs 800 000 kr, 3 uker før 90%, dvs 720 000 kr, 2uker 80% , dvs 640 000 kr én uke før 70% , dvs 560 000 kr og på tid 60% av bonusgrunnlaget, dvs 480 000 kr i bonus.*

*I dette eksemplet leverte Leverandøren på estimert tid og fikk utbetalt 480 000 kr i sluttbonus.*

# Vedlikeholdsavtale

Smidigavtalen inneholder ikke bestemmelser om vedlikeholdsavtale.

Rettighetsbestemmelsene i kapittel 10 i avtalen legger til rette for at Kunden kan inngå vedlikeholdsavtale med en annen Leverandør enn den som har utviklet Programvaren. Dette kan være krevende, men sikrer leverandøruavhengighet for det som er utviklet. Dessuten gir bytte av Leverandør en fin mulighet til å få testet systemdokumentasjon og Ikke-funksjonelle krav som for eksempel Vedlikeholdbarhet i forbindelse med leverandørbytte.

# Dagbot

Dagbøter hører i utgangspunktet ikke hjemme i en smidig programvareutviklingsprosess. Grunnen er at denne utviklingsprosessen forutsetter tett samarbeid mellom Kunde og Leverandør, og det skal - og vil - derfor ikke være mulig å avgjøre hvem som har ansvar for en eventuell forsinkelse.

Avtalen har likevel et dagbotregime både for programvareutviklingsleveransene og for Øvrige leveranseelementer. Dagbøter i smidigavtalen er vesentlig mindre enn dagbøtene i de andre SSA-ene, fordi de blir beregnet kun ut fra verdien av det leveranseelementet som er forsinket (se Avtalens punkt 11.5.2).

7.1 Dagbot ved forsinkelse i programvareutviklingsleveransen
Dagbot bør kun benyttes i forbindelse med Avslutning (Avtalens kapittel 2.6).

Frister hvor det kan være aktuelt med dagbot:

* Frist for oppfyllelse av startkriteriene for kundens avsluttende testing:
* Dagbot dersom Leverandøren blir forsinket fordi han ikke klarer å godtgjøre at Startkriteriene for Kundens avsluttende testing (Avtalens punkt 2.6.1)
* Frist for Overlevering etter Kundens avsluttende testing:
* Dagbot dersom frist for Overlevering (Avtalens punkt 2.6.2) etter Kundens avsluttende testing ikke overholdes.
* Leveransedag etter avsluttende godkjenningsperiode
Dagbot dersom Leverandøren ikke overholder frist for Leveransedag etter avsluttende godkjenningsperiode.

7.2 Dagbot ved forsinkelse i Øvrige leveranselementer
Øvrige leveranser avviker ikke i sitt vesen fra Leveranser i mer tradisjonelle SSA-er, så her kan det være relevant med dagbot ved forsinket levering av noen av leveranseelementene. Et eksempel kan være frist for etablering av Utviklingsmiljø eller Testmiljøer, dersom dette ikke skyldes forhold som Kunden er ansvarlig for.

# Begrepsforklaringer

Begrepsforklaringene i listen nedenfor er i det vesentlige hentet fra den norske versjonen av terminologilisten til International Software Testing Qualification Board ISTQB 2.2N, datert 20.3.2013. Listen er mer omfattende enn den tilsvarende listen i kapittel 17 i avtalen og inneholder begreper brukt i Veiledning til bilag og Veiledning til avtaletekst i tillegg til begrepene som er brukt i Avtalen.

|  |  |
| --- | --- |
| **Begrep** | **Forklaring** |
| AkseptansekriterierISTQB 2.2.N | Sluttkriterier som en komponent eller et system må oppfylle for å bli godkjent av en kunde, bruker eller en annen autorisert enhet [IEEE 610] |
| AkseptansetestingISTQB 2.2.N | Formell testing med hensyn til brukerbehov, brukerkrav, myndighetskrav og kundens arbeidsprosesser som utføres for å avklare om et system oppfyller akseptansekriterier eller ikke. [IEEE 610] |
| AvbruddskriterierISTQB 2.2.N | Kriteriene som brukes for (midlertidig) å stoppe hele testen eller en del av den på testobjektet. [IEEE 829] |
| Avtalen | Generell avtaletekst med bilag |
| Behovsbeskrivelse | Verbal beskrivelse av behov, uten bruk av spesielle notasjoner eller formater |
| Brukerhistorie | Behovsbeskrivelse beskrevet på følgende format:*Som en <rolle>, ønsker jeg å <behov>, (slik at <hensikt/fordel>)”.*  |
| Delleveranse | En av flere programvareleveranser i avtalen (release) |
| Epos | Stor/omfattende Brukerhistorie |
| Estimert Totalkost(nad) | Estimert totalpris for kontrakten. Tilsvarer kontraktspris i øvrige SSA-er |
| Exit | Avslutning av avtalen før Akseptansetest av første Delleveranse etter initiativ fra én av partene |
| FeilhåndteringISTQB 2.2.N | Prosessen med å oppdage, undersøke, behandle og disponere feil. Dette inkluderer å beskrive feil, klassifisere de og identifisere deres alvor. [IEEE 1044] |
| Funksjonell testingISTQB 2.2.N | Testing basert på en analyse av spesifikasjonen av funksjonaliteten av en komponent eller system. |
| Funksjonelt kravISTQB 2.2.N | Et krav som spesifiserer en funksjon som et system eller en komponent må utføre. [IEEE 610] |
| GodkjenningskriterierISTQB 2.2.N | Beslutningsregler som brukes for å bestemme om et testobjekt eller en feature har klart eller ikke klart en test (skal betraktes som godkjent). [IEEE 829]  |
| Ikke-funksjonelt kravISTQB 2.2.N | Et krav som ikke går på funksjonalitet, men på egenskaper som for eksempel pålitelighet, effektivitet, brukbarhet, vedlikeholdbarhet, portabilitet etc. |
| IntegrasjonstestISTQB 2.2N | Test som utføres for å finne feil i grensesnittene og samspillet mellom integrerte komponenter eller systemer |
| KomponenttestingISTQB 2.2N | Test av individuelle programvarekomponenter [IEEE 610] |
| Leveranseplan | Plan for programvareleveransene i Avtalen |
| Overlevering | Formell overlevering av programvare fra Leverandør til Kunde i henhold til avtale  |
| Produksjonssetting | Sette i ordinær drift |
| Produkteier | Product owner i Scrum. Den som skal ha programvaren som utvikles, behovshaver, for eksempel lederen for de brukerne som skal benytte programvaren som utvikles. |
| Produktkø | Product backlog i Scrum. Lise over de elementene (krav/løsningsspesifikasjoner) som skal utvikles. Kan gjerne være formulert som Brukerhistorier |
| ProgramvareISTQB 2.2.N | Dataprogrammer, prosedyrer og eventuell tilknyttet dokumentasjon og data relatert til drift av et datasystem[IEEE 610] – oversatt fra engelsk |
| RetestISTQB 2.2N | Testing som kjører det testtilfelle som feilet siste gang testen ble kjørt, for å kontrollere at en feil er rettet. |
| ScrumISTQB 2.2.N | Et iterativt inkrementelt rammeverk for å lede og styre prosjekter. Brukes vanligvis ved smidig softwareutvikling. |
| Scrummaster | Koordinator for et utviklingsteam som benytter Scrum for å utvikle programvare |
| SluttkriterierISTQB 2.2.N | Generiske og spesifikke betingelser for å tillate at en prosess blir offisielt avsluttet og som de ulike interessentene er blitt enige om. Målet med sluttkriterier er å forhindre at en oppgave blir ansett som avsluttet når det fortsatt finnes utestående deler av oppgaven som ikke er avsluttet. Sluttkriterier blir brukt i testingen for å rapportere mot og for å planlegge når en skal stoppe testingen. [Gilb og Graham] |
| Smidig programvare-utvikling ISTQB 2.2.N | En samling programvareutviklingsmetoder basert på iterativ og inkrementell utvikling, hvor krav og løsninger utvikles gjennom samarbeid mellom selvstyrte team bestående av folk med forskjellige arbeidsfunksjoner (”crossfunctional”). |
| Sprint | Brukes i Scrum om en utviklingsperiode eller utviklingssyklus.En sprint bør vare én til fire uker |
| Sprintkø | Sprint backlog i Scrum . Liste over det Utviklingsteamet har prikket ut blant den høyest prioriterte funksjonaliteten i Produktkøen, som det de skal utvikle i løpet av en Sprint. |
| Standardprogramvare | Programvare som er laget for levering til flere brukere, hvor lisens (disposisjonsrett) kan erverves uavhengig av tjenester fra programvareprodusenten. |
| StartkriterierISTQB 2.2.N | Mengden generiske og spesifikke betingelser for å tillate en prosess til å starte en definert oppgave, for eksempel testutførelsen. Formålet med startkriteriene er å forhindre at en oppgave blir startet når dette fører til mer bortkastet arbeid, sammenlignet med arbeidet med å gjøre noe med de feilede startkriteriene. [Gilb og Graham] |
| SystemtestISTQB 2.2N | Prosessen med å teste et integrert system for å verifisere at det oppfyller spesifiserte krav. [Hetzel] |
| Testbruker | Brukeridentitet som benyttes i testøyemed. En testbruker er ikke en reell bruker |
| Testdata | Data spesielt konstruert på forhånd for å gjøre det mulig å gjennomføre handlingene i testtilfelle osv. Testdata kan kopieres fra produksjon eller andre steder, og eventuelt manipuleres, eller de kan konstrueres (tidl. DND terminologiliste). |
| TestmateriellISTQB 2.2N | Alt som trenges for å planlegge, designe eller gjennomføre tester. Kan innebære dokumentasjon, skripter, input, forventede resultater, oppsett og oppryddingsprosedyrer, filer, databaser, miljø og all ytterlige programvare og verktøy som brukes under en testprosess. [Fewster and Graham] |
| TestmiljøISTQB 2.2.N | Et miljø som består av hardware, instrumentering, simulatorer, programvareverktøy og andre støtteelementer som behøves for å utføre en test. [IEEE 610] |
| TestplanISTQB 2.2.N | Et dokument som beskriver omfang (hva som skal testes), tilnærming til test, ressurser og tidsplan for planlagte testaktiviteter. Planen identifiserer testobjekter, hva som skal testes, testoppgavene og hvem som skal utføre disse, testernes grad av uavhengighet, testmiljøet, testdesignteknikker og testmåleteknikker som skal brukes og begrunnelsen for deres valg, og beskriver risikoene og planene for deres inntreden. Testplanen er en dokumentasjon av testplanleggingen.[IEEE 829] |
| TeststrategiISTQB 2.2.N | Et høynivådokument som definerer testnivåene som skal utføres og testingen innenfor disse nivåene for et eller flere prosjekter  |
| Utprøving | Brukere utenfor prosjektorganisasjonen får tilgang til programvaren som er utviklet i en Delleveranse for å prøve den ut og komme med tilbakemeldinger.  |
| Utviklingsmiljø | Et miljø bestående av hardware og programvare, som brukes av utviklerne for å utvikle programvaren.  |
| VedlikeholdbarhetISTQB 2.2.N | Egenskapen som uttrykker hvor lett det er for et programvareprodukt å bli endret for å rette feil, møte nye krav, gjøre framtidig vedlikehold enklere, eller bli tilpasset en endret omgivelse. [ISO 9126] |
| Virkedager | De dager som ikke er lørdager, søndager og offentlige høytids- og helligdager, og heller ikke jule- og nyttårsaften. |
| Øvrige leveranseelementer | Leveranselementer som er relatert til programvareutviklings-leveransen, men som ikke er en del av den, for eksempel opplæring, konvertering, systemintegrasjon, rutineutvikling. |