NTNU
Det skapende universitet

Metodikker for testmiljø

Unn Aursøy
Wenche Tollevsen

Master i datateknikk
Oppgaven levert: Juni 2007
Hovedveileder: Tor Stålhane, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Test er et område i sterk utvikling. Dette har resultert i at modenhet og utbredelsen av testmetodikker er økende. Innenfor testmiljø har det imidlertid vært mindre fokus på metode og guidelines.

Det finnes en "Best Practices" om krav og oppbygging av testmiljø i test-rammeverk for automatisert testing, og testmetodikker, men den er generell og vanskelig å bruke direkte i en testmiljømetodikk.

Dagens testmiljøer kan i store trekk deles inn i to hovedretninger; utviklingsmiljø og testmiljø. Utviklingsmiljø har i EDB lite referanser mot et endelig produksjonsmiljø, mens testmiljø har krav om likhet i forhold til produksjonsmiljø og dokumentering av avvik i forhold til dette. Kostnader til test og testmiljø er avgjørende i de fleste IT-prosjekter.

Sett i lys av den utviklingen som finner sted innen virtualisering gir dette noen teoretiske økonomiske besparelser både innen timer og investeringer.

1. Utredning
- Hvilke testmiljømetodikker finnes i markedet i dag? Hvor gode er de? Hvem benytter disse?
- Gjør en vurdering av risiko ved å benytte et testmiljø som ikke er produksjonslikt i en Leveranse-/Akseptansetest
- Hvilke programvare for testing/testmiljø og virtualisering finnes på markedet?
- Hvordan arbeider norske bedrifter med testmiljø i forhold til EDB?

2. Skriv en guideline for opprettelse av testmiljø for EDB Business Partner (testmiljømetodikk):
- Hva skiller test og produksjon
- Hvilke risikoer tar man ved å avvike fra produksjon (nedskalert/OS-type/OS-nivå/3. partsprodukter/testdata)
- Hvilke type verktøy kan man bruke for å få mest mulig kostnadseffektiv utnyttelse av testmiljø (virtualisering/simulatorer).
- I hvilken grad trenger man adskilte testmiljøer for de ulike testfasene (Integrasjonstest, Systemtest og Leveranse-/Akseptansetest, Ytelsestesting).

Oppgaven gitt: 22. januar 2007
Hovedveileder: Tor Stålhane, IDI

Sammendrag

Oppgaven omhandler testmiljø, og utreder hvilke metodikker for testmiljø som finnes i markedet og hvor oppdragsgiver for oppgaven, EDB Business Partner, står i arbeid med testmiljø i forhold til andre norske bedrifter.

Oppgaven beskriver et gjennomført litteraturstudium rundt testmiljø, og konkluderer med at det finnes lite dokumentert på området. Av det som finnes, er Testing Maturity Model (TMM) og dets krav for testmiljø det mest relevante. Vi ser så på hva som finnes på markedet av testverktøy. Videre gjengir oppgaven resultater fra seks gjennomførte samtaler i norske bedrifter, og diskuterer hvordan de involverte bedriftene jobber med testmiljø. Hovedinntrykket er at det finnes lite dokumentert, offisiell metodikk for testmiljø, men at arbeidet med testmiljø i bedrifter er basert på interne erfaringer. Dette viser at EDB er kommet langt i arbeid med testmiljø i forhold til andre bedrifter. Oppgaven konkluderer med at bedrifter som arbeider med testing bør utarbeide og følge en offisiell, dokumentert metodikk for testmiljø. Som et grunnlag for dette, er resultater fra samtaler og litteraturstudium samlet som et sett med retningslinjer beregnet for bruk ved arbeid med testmiljø.

Forord

Denne oppgaven utgjør den avsluttende masteroppgaven ved sivilingeniørstudiet i datateknikk ved Institutt for datateknikk og informasjonsvitenskap (IDI) ved Norges teknisk-naturvitenskapelige Universitet (NTNU).

Vi ønsker å takke hovedveiler Tor Stålhane for god rettledning og verdifull hjelp i arbeidet med denne oppgaven. Vi har satt pris på innspill og gode råd, og at vi alltid har blitt møtt med åpen dør.

IT-konsernet EDB Business Partner har vært initiativtaker til oppgaven, og har i tillegg stilt med medveileder Trygve Lauritzen. Vi har også fått stor hjelp av leder for testavdelingen, Vivi Horn. Vi ønsker å rette en stor takk til disse personene for faglig oppfølging og hjelp til den praktiske gjennomføringen av oppgaven. Vi takker også EDBs testavdeling i Oslo som stilte opp på samtale med oss.

Mange andre personer har bidratt til å gjøre oppgaven mulig. Vi ønsker å takke Hans Schaefer som har gitt oss nyttige innspill, og for at han satt oss i kontakt med Hans Erik Ballangrud i IBM. Ballangrud har, i tillegg til selv å stille opp på samtale, formidlet kontakt med ytterligere tre bedrifter som kunne være aktuelle intervjuobjekter. Disse bedriftene er HEMIT ved Vidar Myklebust, Domstoladministrasjonen ved Ståle Risem-Johansen og Abeo ved Jan-Erik Hagerud. Vi er svært takknemlige for at de nevnte personer tok seg tid til å stille opp på samtaler med oss, og med dette gitt et uvurderlig bidrag til oppgaven vår.

Trondheim, 7. juni 2007

Unn Aursøy Wenche Tollevsen

Innhold

Figurer

Tabeller

Kapittel 1
Introduksjon

Det er blitt stadig større fokus på at innføring av nye datasystemer skal gå problemfritt, og at uforutsette problem ikke skal oppstå. Oppstår det problemer kan dette føre til store ekstrakostnader, forsinkelser og fare for sikkerheten til brukerne. Skifter man for eksempel kommunikasjonssystem ved et sykehus, er man avhengig av at systemet har gjennomgått grundig testing, da feil i systemet kan føre til nedetid som skaper fare for liv og helse hos pasientene. Ikke alle systemer er like kritiske med hensyn på faren for liv, men man er fortsatt avhengig av at systemene man leverer er så feilfrie som mulig.

Dette fører til at testing blir en vital del av bedriften, og flere bedrifter ender opp med å etablere egne testavdelinger i bedriftene sine. Når man utvikler programvare, skal ikke bare hver enkelt modul i programmet fungere etter spesifikasjonene, men modulene skal også integrere med hverandre som spesifisert. Programvaren som utvikles er igjen avhengig av å fungere i produksjonsmiljøet. Produksjonsmiljø er en samlebetegnelse på det miljøet systemet benyttes i, og omfatter blant annet programvare, maskinvare og eventuelle nettverk. Lager man for eksempel et transaksjonssystem for en bank, er man avhengig av at systemet samarbeider problemfritt med andre banksystem, blant annet Bankenes betalingssentral (BBS).

For å kunne teste systemene utenfor produksjon, opprettes det testmiljø der programmene testes. Kostnadene ved å bygge og vedlikeholde et testmiljø kan beløpe seg til store summer. Maskinvare er kostbart, men også programvare, som lisenser på databaser, er dyrt. Når man bygger et testmiljø, ønsker man å kopiere det faktiske miljøet produktet skal brukes i. Dette er vanskelig, og blant annet en økonomisk avveing. Det er derfor interessant å vurdere hvor likt et akseptansetestmiljø må være det faktiske produksjonsnivået for at det skal være bra nok. Det er også viktig å tenke på er at testmiljøet lett skal kunne tilbakestilles til utgangspunktet for å kunne kjøre nye tester.

Vår samarbeidsbedrift EDB er en av de bedriftene i Norge abeider mye med testmiljø. Selv om bedriften har god kunnskap innen testmiljø, er de interessert i å øke sin kunnskap på nettopp dette området.

1.1 Forskningsspørsmål

I samarbeid med EDB sin testavdeling i Trondheim er det blitt utviklet fire forskningsspørsmål, som er tema for denne rapporten. Spørsmålene er svært

relevante for EDB, men vi tror også dette er interessant for andre bedrifter som benytter seg av testmiljø. På bakgrunn av forskningsspørsmålene har vi utviklet egne retningslinjer for testmiljø, som er uavhengig av størrelsen på bedriften eller arbeidsområde.

1. Hva finnes av dokumentasjon på metodikker for testmiljø?
2. Hvor likt produksjonsmiljøet må et akseptansetestmiljøet være?
3. Trenger man et eget testmiljø for hvert testnivå?
4. Hvor står EDB i forhold til andre norske bedrifter innenfor testmiljø?

1.2 Framgangsmåte

Denne oppgaven presenterer et grundig litteraturstudium av tilgjengelige metodikker for testmiljø og presentasjon av viktige teoretiske konsept. Deretter presenterer vi resultatene fra intervjuer og risikoanalyser gjennomført i norske bedrifter. Dette analyseres og oppsummeres i et sett av retningslinjer for bruk ved etablering, bygging, bruk og vedlikehold av testmiljø.

En av utfordringene med oppgaven har vært å finne metodikker, da det viser seg at dette er et område der det er publisert svært lite. Et grundig litteraturstudium, som beskrevet i kapittel 4, er blitt gjennomført, og litteraturstudiumet bygger opp under tesen om at det ikke er mye som er publisert om testmiljø. Dette samsvarte også med det inntrykket medveilederne hos EDB hadde. Det som finnes av informasjon er likevel blitt samlet, og i teoridelen av denne oppgaven samles den publiserte informasjonen som er funnet om bygging og bruk av testmiljø.

EDB hadde også et ønske om at vi skulle komme med anbefalinger innen verktøy for testmiljø, og dette har resultert i en oppsummering av hvilke verktøy som finnes på markedet. I tillegg til å se på en rekke verktøy sine spesifikasjoner, har vi vurdert resultat fra undersøkelser og tester av verktøy.

På tross av at det er lite litteratur på området, bygges og brukes testmiljø i flere norske bedrifter. Trolig sitter disse bedriftene på stor kunnskap om sitt eget testmiljø, uten at vi har kjennskap til at erfaringene er blitt publisert. Vi har derfor valgt å prøve å få innblikk i bedrifters testmiljø ved å gjennomføre intervju og risikoanalyse, med testmiljø som tema.

Gjennomføringen av intervjuene er beskrevet i kapittel 11, mens de utskrevne intervjuene ligger i vedlegg B. Det ble gjennomført samtale hos fem bedrifter, noe som ga et bredt bilde av norske bedrifters forhold til testmiljø. Bedriftenes arbeidsområder var spredt og omfattet bank, helsevesen, domstol, telekommunikasjon og et konsulentselskap. Målet vårt var å finne fellestrekk og forskjeller, for så å kunne sette opp retningslinjer. Da dette ikke tidligere er gjennomført, er håpet at nettopp en slik undersøkelse vil være til hjelp både for de deltakende bedriftene og for andre bedrifter.

Tanken bak utviklingen av retningslinjer er at man vil gjøre bruken av testmiljø så god som mulig. Hvilke tiltak er man villig til å iverksette for å unngå et problem? Er det verdt å unngå et problem, om sjansen for at det skal oppstå er svært liten og tiltakene for å unngå det er svært dyre? Dette har vært grunnen for at vi i tillegg til intervjuene, som gir mye generell informasjon om testmiljøet i bedriften, også har gjennomført en risikoanalyse. Skjema for risikoanalyse ble fylt ut i forbindelse med intervjuene og disse ligger i vedlegg C. Om man ser bort fra interne kartlegginger, har vi har ingen kjennskap til at dette har blitt gjennomført tidligere.

1.3 Oppdeling av oppgaven

Forstudium Forstudium er første del i rapporten og omfatter teori innen testing og testmiljø. Delen starter med et kapittel som beskriver vårt litteratursøk, før vi gir en generell forklaring av testing og ulike testbegrep i kapittel 5, og testmiljø i kapittel 6. Vi ser så på de fire tilnærmingene til testmiljømetodikk som vi fant i litteraturstudiumet før vi avslutter delen med å beskrive hva som finnes på markedet av testverktøy.

Case Case-delen skildrer hvordan vi har gjennomført intervju og risikoanalyse, hvem vi har gjennomført de hos, og hvilke svar vi fikk fra bedriftene. Delen består av seks kapittel der vi etter å ha introdusert delen skildrer metoden vi har brukt for gjennomføringa. Vi presenterer så EDB som har vært vår samarbeidsbedrift og de andre bedriftene vi har gjennomført samtaler hos, før vi avslutter med to kapittel som oppsummerer intervju og risikoanalyse.

Analyse Denne delen diskuterer resultatene fra samtalene opp mot teoridelen. Delen omfatter både diskusjon og konklusjon, i tillegg til retningslinjene for testmiljø som vi presenterer i kapittel 18. Retningslinjene er våre råd til hvordan man bør bygge og bruke testmiljø og er basert på resultatene fra forstudiumet, case-delen og den påfølgende diskusjonen. Resultatene blir oppsummert i konkrete retningslinjer. Deretter følger konklusjon, før vi avrunder analysen med å komme med forslag til videre arbeid på området.

Kapittel 2
Eget bidrag

Målet med denne oppgaven er todelt:

1. Å se på hvilke metodikker som finnes for bygging og bruk av testmiljø.
2. På bakgrunn av disse metodikkene lage et sett med retningslinjer for hvordan dette bør gjøres.

Det finnes mye publisert litteratur om utviklingsmetodikk og om testing, mens det innenfor området testmiljø er mangelfullt. I denne rapporten har vi kartlagt bruken av testmiljø i fem norske bedrifter, vist risikofaktorer de opplever i forbindelse med testmiljø og presentert våre retningslinjer for bygging og bruk av testmiljø.

Gjennom litteraturstudiumet har vi kartlagt hva som finnes av publisert materiale om testmiljø, og resultatene av dette presenteres i kapittel 4. Vi kan på bakgrunn av dette konkludere om at det finnes svært lite litteratur om bygging og bruk av testmiljø. Våre samtaler om testmiljø, kartlegger holdninger til testmiljø i flere norske bedrifter. Samtalene har bestått av både intervju og risikoanalyse, og våre resultater diskuteres i kapittel 17. Resultatene viser blant annet at:

- Testmiljø er et område der lite er dokumentert.
- Mye kunnskap finnes som erfaring hos de ansatte.
- Det er uvanlig for en norsk bedrift å ha en egen nedskrevet testmiljømetodikk.
- Hva som legger føringer for bruk av testmiljø varierer.
- Bedriftene utvikler egne metoder for bygging og bruk av testmiljø, ettersom det finnes lite litteratur på området.
- Hvert testmiljø er forskjellig, men det finnes likevel sterke fellestrekk. For eksempel er utfordringene rundt sensitive testdata gjengående, selv om man befinner seg innenfor forskjellige fagområder som helse, bank og domstol.

Ved bygging og bruk av testmiljø møter bedriftene flere utfordringer. Vi har kartlagt risikoer i forbindelse med testmiljøbruk, gjennom risikoanalysen i undersøkelsen. Disse risikoene er oppdelt i tre hovedgrupper:

1. Organisatoriske risikoer
2. Samarbeidsrisikoer

3. Tekniske risikoer

En beskrivelse av risikoene finnes i kapittel 15.

Retningslinjene som oppsummerer rapporten er veiledende for bygging og bruk av testmiljø, og gir råd for hvordan dette bør gjennomføres på en best mulig måte. Disse har vi skrevet på bakgrunn av det som finnes av litteratur på området og undersøkelsene våre i norske bedrifter. Retningslinjene er et utgangspunkt når man skaper en egen testmiljømetodikk og er uavhengig av størrelsen på bedriften eller arbeidsområde. Retningslinjene finnes i kapittel 18, og omfatter tips innen disse områdene:

- Dokumentasjon
- Testleders ansvarsområder
- Verktøy
- Gjenoppretting av testmiljø
- Produksjonslikhet
- Dynamikk i testmiljøet
- Testing og kvalitetssikring av testmiljøet

Arbeidet med denne rapporten har altså ført til at vi har fått samlet informasjon innen testmiljø, både fra litteraturen og undersøkelser i norske bedrifter. Dette har gitt et innblikk i hvordan man bør bygge og bruke testmiljø, og vi håper at våre retningslinjer vil være til nytte for EDB, og andre norske bedrifter, i deres arbeid med testmiljø.

Del I

Forstudium

Kapittel 3
Introduksjon til forstudium

Denne første delen av oppgaven tar for seg teori rundt testing, testmiljø og risikoanalyse. Vi ønsker å etablere et teoretisk fundament, som skal ligge til grunn for retningslinjer for testmiljø i Del III.

3.1 Delens oppbygging

Litteratursøk Dette kapittelet tar for seg litteratursøket som ble gjennomgått i forkant og underveis i arbeidet med teorien. Kapittelet presenterer framgangsmåten ved søk etter artikler, bøker, konferanser og generelle søk på internett.

Testing Testing-kapittelet omhandler generell teori rundt testing. Vi gjennomgår definisjon på test, dekningsgrad, tilnærminger til test, testnivå og automatiske tester.

Testmiljø Dette kapittelet omhandler teori rundt testmiljø. Her ønsker vi å gi en innføring til testmiljø, undergrupper av testmiljø og hva som kan være viktig ved bygging av testmiljø, basert på teori fra området.

Metodikker for testmiljø Dette kapittelet vurderer fire tilnærminger til metodikk for testmiljø. De fire som presenteres er en artikkel av Richard Caesar, Certified Software Tester (CSTE), en mal fra Hans Schaefer og Testing Maturity Model (TMM).

Hva finnes på markedet Dette kapittelet ser på hva som finnes av løsninger for testmiljø på markedet og omfatter blant annet programvare for visualisering.

Riskoanalyse Dette kapittelet gir en innføring til risikoanalyse. Tema som omhandles inkluderer hvorfor man kan benytte risikoanalyse i programvareutvikling og i arbeidet med testmiljø.

Kapittel 4
Litteratursøk

For å skrive en oppgave av dette omfanget, kreves mye kildemateriale, noe som fordrer et systematisk og omfattende litteraturstudium. Dette kapittelet beskriver framgangsmåten vår ved søk etter kilder. Aktuelle kilder danner så grunnlaget for resten av kapitlene i del I.

4.1 Artikler

Vitenskapelige artikler er en viktig kilde og NTNU abonnerer på en rekke vitenskapelige tidskrift og fagdatabaser. Vi har søkt i tilgjengelige databaser for fagområdet datateknikk, i tillegg til Google Scholar, en søkemotor for akademisk litteratur. Tabell 4.1 viser en oversikt over databaser, hva vi har søkt på og hvor mange av treffene vi sjekket.

4.1.1 Utvelgelse av artikler

For å avgjøre hvilke av treffene som kunne være aktuelle, leste vi raskt gjennom tittelen, sammendraget og eventuell stikkordliste for hver artikkel. I tvilstilfeller leste vi raskt gjennom selve artikkelen også. Evalueringen av treffene gikk raskere etterhvert, fordi mange av søkemotorene returnerte de samme artiklene. Som vi ser i Tabell 4.1 er det stor forskjell på hvor stort antall av treffene vi sjekket fra hver søkemotor. Dette skyldes at enkelte motorer returnerte mer relevante treff enn andre, noen returnerte bare noen få treff hvor vi sjekket alle. Disse er merket "Alle" i tabellen. Andre returnerte over tusen treff. For søkemotorer der treffene var lite relevante, for eksempel relatert til kybernetikk, leste vi gjennom de 20 første treffene. Ellers er antall treff som er lest gjennom basert på hvor gode treffene var. For eksempel returnerte ACM interessante treff relatert til programvareutvikling, og her sjekket vi de 110 første.

Noen søkemotorer returnerte samme treffliste ved søk på "Testing environment" som ved "Test environment", mens andre returnerte ulike treff. Dette har med søkemotorens oppbygging og rangering å gjøre.

Til tross for denne grundige gjennomgangen av aktuelle kilder, fant vi bare 16 artikler vi ville skrive ut for å lese ordentlig gjennom. Etter gjennomlesing satt vi igjen med tre aktuelle artikler, derav en blir omtalt i kapittel 7. De øvrige

Database	Søkeord	Sjekket
Google Scholar	Test environment	14
	Testing environment	13
	Testing environment software	16
	Test environment software	40
	Building test environment software	20
	Building reusable test environment	20
	Building reusable testing environment	20
	Test environment production environment software	20
	Best-practice test environment	20
ACM	Test environment	110
	Testing environment	Samme som over
Blackwell Synergi	Test environment	20
	Testing environment	20
	"Test environment"	20
Engineering Village	Test environment	75
Scirius	Test environment	10
Dekker Encyclopedia	Test environment	10
Electronic Journal Service	"Test environment"	20
IEEE	"Test environment"	41
Ingenta Connect	"Test environment" software	7
JStor	"Test environment"	27 (Alle)
	"Testing environment"	22 (Alle)
Oxford Journals	"Test environment" software	4 (Alle)
	"Testing environment" software	10
Science Direct Elsevier	"Test environment" software	16 (Alle)
	"Testing environment" software	8 (Alle)
Springer Link	"Test environment" software	19 (Alle)
	"Testing environment" software	9 (Alle)
Wiley InterScience	"Test environment"	75
	"Testing environment"	Samme som over

Tabell 4.1: Litteratursøk artikler

to hadde en generell innledning vi kan ha nytte av. Felles for flere av de andre artiklene er at de evaluerer et spesifikt system det er vanskelig å overføre til vår oppgave. I tillegg var flere av artiklene fra slutten av åtti-tallet og tidlig nitti-tall, noe som indikerer at systemene de evaluerer kan være utdaterte. Ingen av artiklene definerte begrepet testmiljø, og det virket som om begrepet var brukt som en betegnelse på et verktøy for testing.

På bakgrunn av dette konkluderer vi med at det finnes få artikler som tar for seg testmiljø, og dette er i overenstemmelse med hva vi ble fortalt av EDB.

4.2 Bøker

En annen aktuell kilde til informasjon er bøker om testing. Dette inkluderer både lærebøker innenfor området, beste-praksis-bøker for testere og bøker som tar sikte på å forbedre en eksisterende testprosess.

For å finne aktuelle bøker, brukte vi et nasjonalt biblioteksystem for norske fag- og forskningsbibliotek, BIBSYS, og Google Book Search. Bøkene ble bestilt gjennom hovedbiblioteket ved NTNU.

Vi fant 4 bøker som inneholdt kapitler eller deler om testmiljø, deriblant en bok vi ble anbefalt av EDB (Pol, Teunissen & van Veenendaal 2002). Disse danner fundamentet for kapittel 6. I tillegg lånte vi 9 bøker som grunnlag for innføringskapittelet i testing kapittel 5.

4.3 Konferanser

Vi besøkte hjemmesidene til flere konferanser innenfor testing for å se om noen av de hadde presentert tema som omhandlet testmiljø. Vi leste raskt gjennom artiker og presentasjoner fra konferanser, men fant lite av interesse. Dette er også i overenstemmelse med hva vi ble fortalt av EDB, nemlig at det finnes få konferanser med testmiljø som tema.

VMware arrangerte et "Roadshow" i mars, det vil si en presentasjon av produktene sine, og dette deltok vi på. Fokuset på dette seminaret var mer rettet mot drift, og hvordan virtualisering kunne benyttes i driftsammenheng. Vi fikk likevel en del bakgrunnsinformasjon om VMware som blir brukt i kapittel 8.

4.4 Internett

I tillegg til søkene som er beskrevet ovenfor, gjennomførte vi generelle søk på internett, hovedsakelig ved hjelp av søkemotoren Google. Vi fant en del nettsider som tok for seg testing, noen artikler fra IT-nettaviser og interessegrupper for testing. Imidlertid må man utvise stor grad av kildekritikk

ved bruk av nettsider, ettersom dette er stoff som ikke er kvalitetssikret av et forlag eller tidsskrift.

En viktig kilde til stoff om programvare i kapittel 8 er internett. For å finne informasjon om programmer for testmiljø og virtualisering har vi benyttet oss av produsentenes hjemmesider. Denne typen kilder er det grunn til å behandle med forsiktighet, ettersom produsentene har en klar agenda, nemlig å markedsføre produktet sitt. Vi har prøvd å ta hensyn til dette, og i stor grad benyttet produktspesifikasjoner, som vi anser å være mer nøytrale, enn generell, positiv omtale av produktet som gjerne finnes på forsiden til produsenten. Vi er imidlertid klar over problemet med at produsentene kan inkludere funksjonalitet som ikke er implementert i en produktspesifikasjon, eller de kan ha forskjellig oppfatning av hva "støtter" betyr.

4.5 Sammendrag

I dette kapittelet har vi presentert framgangsmåten og resultatene under litteratursøket. Kapittelet har vist at det finnes lite informasjon om testmiljø, resultatet av litteratursøket ble tre artikler og fire bøker med kapitler om testmiljø, tillegg til artikler fra IT-aviser og interessegrupper for testing. Dette er stoff som benyttes i kapittel 6 og kapittel 7. Generell litteratur om testing ble også samlet inn, i form av 9 bøker til bruk i kapittel 5.

Kapittel 5
Testing

Dette kapittelet ser på generell teori om testing, og presenterer begreper, modeller og temaer rundt området programvaretesting. Dette danner så grunnlag for kapittel 6 som fokuserer mer spesifikt på testmiljø, som er en eget tema under testing.

5.1 Definisjon på test

Vi vil i denne oppgaven benytte begrepet test om programvaretesting. Programvaretesting kan gjennomføres på flere måter, men det vil alltid være noe som testes, et system, ved at man gir inn data og evaluerer data som man får ut av systemet. Dette er enkelt demonstrert i figur 5.1 som beskriver hva vi legger i begrepet test. Hvilken tilstand systemet befinner seg i før og etter testen er også relevant og refereres til som start- og sluttilstand.

Figur 5.1: System under test

5.2 Dekningsgrad

Dekningsgrad er relasjonen mellom hva som faktisk er testet av et sett med tester, og hva som kan bli testet (Pol et al. 2002). Konseptet blir ofte brukt i forbindelse med programkode. Har man et sett med tester, kan man regne ut hvor mange prosent av kodelinjene som er dekket. Har man forgreininger i koden, for eksempel ved if-setninger, må man ha en test for hver valgmulighet. Dette prosenttallet er nyttig for å avgjøre hvor langt man er fra å ha testet ferdig (Kaner, Bach & Pettichord 2002). Dersom en tester har testet 10% av kodelinjene bør man ikke ha mye tillit til programvarens pålitelighet. Men som (Kaner et al. 2002) påpeker, nærmer man seg 100% dekning av kodelinjene

betyr ikke dette at produktet er nært fra å være ferdig. Det betyr bare at det ikke lenger er opplagt at produktet er langt fra ferdig.

For å avgjøre hvor stor andel av kildekoden som er gjennomgått under testing, kan spesielle verktøy brukes. Verktøyene rapporter blant annet hvilke linjer som ble gjennomgått, om både *true* og *false* i en forgreining ble utført eller om unntak ble fanget korrekt (Doar 2005).

I tillegg til dekningsgrad av kode, kan begrepet også brukes i forbindelse med funksjonelle spesifikasjoner eller krav (Pol et al. 2002). Utvikler man for eksempel et produkt for massemarkedet er det viktig å teste systemet under flere forskjellige operativsystem, nettlesere, internetthastigheter og så videre, og kombinasjoner av disse.

Det er problematisk å avgjøre hvor god dekningsgrad testing har. Framfor alt er dette et spørsmål om ressurser, og en viktig del ved planleggingen av en teststrategi er å finne den optimale balansen mellom arbeid nedlagt i testing og dekningsgraden som kreves for å minske risikoen for feil (Pol et al. 2002). Høyere dekningsgrad betyr vanligvis mer ressurser brukt på testing.

5.3 Tilnærminger til test

Ved tilnærming til testing av programvare er det black- og white box-testing som er mest aktuelle. En kombinasjon av disse to teknikkene har fått navnet grey box-testing.

5.3.1 Black box

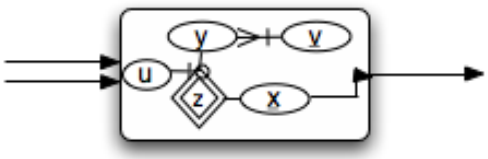
Black box-testing baserer seg på at man ikke vet noe om hvordan systemet er laget, og systemet blir betraktet som en svart boks der hvordan det er bygd opp innvendig er irrelevant under testing, som vist i figur 5.2. Det man velger å teste er hvilket resultat man får med gitte inndata til systemet, uavhengig av den indre strukturen. Black box-testteknikker baserer seg dermed på funksjonelle krav og kvalitetskrav og tester hvordan disse blir oppfylt (Pol et al. 2002), og ønsker å sikre at et program møter sine spesifikasjoner både fra et utførelses- og funksjonsperspektiv (Schroeder & Korel 2000). På grunnlag av dette, kalles black box-testing også funksjonell testing.

Figur 5.2: Black box-testing

En av ulempene med denne typen testing er at man aldri kan vite hvor stor del av systemet som er testet (Copeland 2003), ettersom man ikke har noe kjennskap til den indre strukturen.

5.3.2 White box

Testspesifikasjonsteknikkene i white box-testing er vanligvis basert på kode, programbeskrivelse og teknisk design, det vil si eksplisitt bruk av kunnskap om systemets indre struktur (Pol et al. 2002). Et mål kan for eksempel være at alle programlinjer skal utføres. I motsetning til black box-testing baserer altså white box-testing seg på at man har kjennskap til systemet og hvordan det er bygd opp, som vist i figur 5.3. Dette er grunnen til at white box-testing også går under begrepet strukturell testing. Glassboks og klarboks er andre termer som har blitt brukt om white box-testing.



Figur 5.3: White box-testing

Ved white box-testing kan man teste alle delene av et system, men dette behøver ikke å bety at systemet er feilfritt.

5.3.3 Fordeler og ulemper med white box og black box

Black box og white box har begge sine fordeler og ulemper. Black box-testing har den fordelen at den kan simulere faktisk systembruk og gjør ingen antakelser om struktur. På en annen side kan dette føre til at man ikke finner logiske feil i programvaren, og at redundant testing kan forekomme. White box-testing lar testeren få se programvarelogikken, og gjør det mulig å teste strukturelle attributter, slik som kodens effektivitet. Ulemper kan være at man er ikke sikker på om brukerens krav er oppfylt og at det kan være vanskelig å etterlikne virkelige situasjoner (Perry 2006).

Ettersom de to metodene å teste på har hver sine fordeler og ulemper, brukes begge metodene ved testing. Begge metodene brukt sammen kan validere hele systemet (Perry 2006).

5.3.4 Grey box

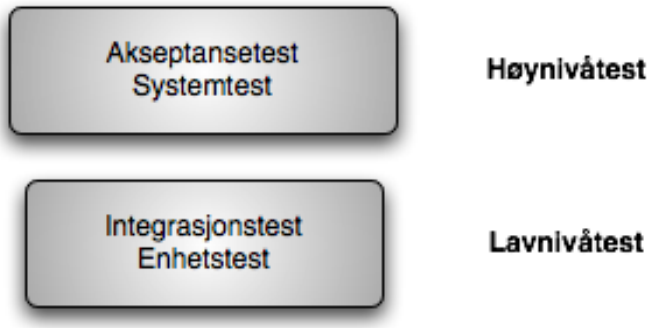
I praksis bruker black box-testing kunnskap om systemet. Testene lages på bakgrunn av kunnskap om designmetoden som er brukt og risikoen dette gir.

Slik oppstår begrepet grey box-testing (Pol et al. 2002). Et system blir sett på som grey box dersom brukerne har noe kunnskap om hvordan det fungerer under overflaten (Battle 2003). Som i black box-testing er det ikke nødvendig at de som utfører testene har kunnskap om systemets struktur, men som i white box-testing har kunnskap om systemet blitt brukt under planleggingsdelen av testene.

Et annet eksempel på grey box-testing kan være at det finnes kunnskap om en bestemt algoritme som er brukt under implementering av en tjeneste i systemet (Battle 2003).

5.4 Testnivå

En vanlig tilnærming til testing er å utføre ulike tester avhengig av hvor i systemutviklingsprosessen man befinner seg. De vanligste testnivå er enhetstest, integrasjonstest, systemtest og akseptansetest. Målgruppen, hvem som utfører testene og i hvilket miljø testen blir utført, varierer med hvert testnivå. For en utdyping av testmiljøene de ulike testnivåene utføres i, henviser vi til delkapittel 6.2. Tester som utføres på delsystemer og seperate komponenter kalles lavnivåtester, mens tester som involverer testing av hele det komplette produkt går under navnet høynivåtester (Koomen & Pol 1999).



Figur 5.4: Testnivå

Som vi ser av figur 5.4 klassifiseres enhetstest og integrasjonstest som lavnivåtest, og integrasjonstest og akseptansetest betegnes som høynivåtester. På hvert testnivå kan vi utføre flere testtyper. Dette kan for eksempel være røyktesting, forhåndsdefinerte testcase som dekker hovedfunksjonaliteten til en komponent eller system uten å gå inn på finere detaljer (Graham, Veenendaal, Evans & Black 2007), stresstesting, tester som utsetter systemet for last forbi grensene i kravspesifikasjonen for å finne ut under hvilke omstendigheter det kræsjer, eller regresjonstesting. Ettersom regresjonstesting er svært utbredt på flere testnivå vil vi beskrive den seperat i avsnitt 5.4.5.

5.4.1 Enhetstest

Enhetstest tester kodens klasser og metoder. En definisjon på programvareenhet er:

> "En enhet er den minste programvarekomponenten det er mulig å teste" (Burnstein 2003)

Man tester at koden fungerer som tiltenkt, for eksempel ved å trykke en tast på tastaturet for å sjekke om den utløser en bestemt handling (Perry 2006). Feil som oppdages under enhetstesting er forholdsvis enkle å lokalisere og rette, ettersom komponenten som testes er relativt liten (Burnstein 2003). Enhetstesting blir som oftest utført av utviklerne i et utviklingsmiljø. Denne type testing krever god kunnskap til den interne strukturen til programmet, derfor benyttes hovedsakelig white box-testing (Koomen & Pol 1999).

5.4.2 Integrasjonstest

Integrasjonstest setter sammen programmoduler og tester at delene fungerer sammen som en gruppe. Et eksempel kan være å teste en database sammen med et program som bruker databasen. Som oftest er det utviklerne som står for utførelsen i et utviklingsmiljø. I likhet med enhetstest, er white box-teknikker mest vanlig (Koomen & Pol 1999), men det blir anbefalt å teste med både black box og white box for å teste systemet best mulig (Burnstein 2003). Integrasjonstest og enhetstest kalles ofte for lavnivåtester som vist på figur 5.4.

5.4.3 Systemtest

Systemtest tester hele systemet sammen, og sjekker at det kjører korrekt og feilfritt. Denne typen tester blir kjørt etter at utviklingen av systemet er ferdigstilt, eller ved slutten av en iterasjon, og blir utført av testere, eventuelt i samarbeid med kunden eller bruker (Perry 2006). I en systemtest ønsker man å finne svake punkter i programvaren, derfor er det best om utviklerne ikke er direkte involvert (Burnstein 2003). Testen blir som oftest utført i et systemtestmiljø (Koomen & Pol 1999), av dedikerte testere.

En systemtest består av flere testtyper, blant annet funksjonelle, brukbarhetstester, ytelsestester og sikkerhetstester. Fra et testmiljøperspektiv er funksjonelle tester og ytelsestester tester som stiller høye krav til testmiljøet.

Funksjonelle tester

Funksjonelle tester er en demonstrasjon av systemets funksjonalitet, og overlapper i stor grad med akseptansetester (Burnstein 2003). Målsetningen med testen er å forsikre seg om at alle funksjonelle krav fra kravspesifikasjonen

er dekket av systemet, og på denne måten forsikre seg om at systemet gjør hva det er forventet å gjøre.

Funksjonelle tester er black box av natur, og dette illustreres av at black box-tester også kalles funksjonelle tester, som beskrevet i avsnitt 5.3.1.

Ytelsestest

Hovedmålet med en ytelsestest er å sjekke om programvaren møter ytelseskravene som er definert i kravspesifikasjonen. Andre nyttige resultater av ytelsestesting kan være at testerne ser om det er program- eller maskinvarefaktorer som har innflytelse på systemets ytelse, og de har mulighet til å finjustere systemet for å optimalisere tildelingen av systemressurser (Burnstein 2003).

Et eksempel på et ytelseskrav kan være at bruker skal få svar fra systemet innen 0.5 sekunder når han/hun gjør en bestemt operasjon. Ytelseskravene må være formulert på en kvantifiserbar måte for å muliggjøre testing (Burnstein 2003).

5.4.4 Akseptansetest

For en bedrift som utvikler programvare er kanskje akseptansetest den viktigste testen. Denne testen kjøres i samarbeid med kunde, og er en endelig sjekk på at kunden er fornøyd med systemet i henhold til kravspesifikasjonen og deres forventninger og mål. Pol deler akseptansetestene i to deler, funksjonell akseptansetest og produksjonsbasert akseptansetest, der sistenevnte tester om systemet oppfyller krav til kontroll- og produksjonsstandard i henhold til last- og ytelsesbehov (Pol et al. 2002).

Akseptansetester ligner på systemtester, og gjenbruk av case fra disse testene er mulig. Det er viktig at testene som lages er representativ for en typisk arbeidsdag. Vanlige og ulovlige inndata og alle hovedfunksjoner bør bli testet. Er systemet beregnet å kjøre kontinuerlig, bør man teste minst gjennom en 25-timers sykel (Burnstein 2003).

For å teste på en realistisk måte, kjøres akseptansetest i et miljø som ligner produksjonsmiljøet mest mulig (Koomen & Pol 1999, Burnstein 2003).

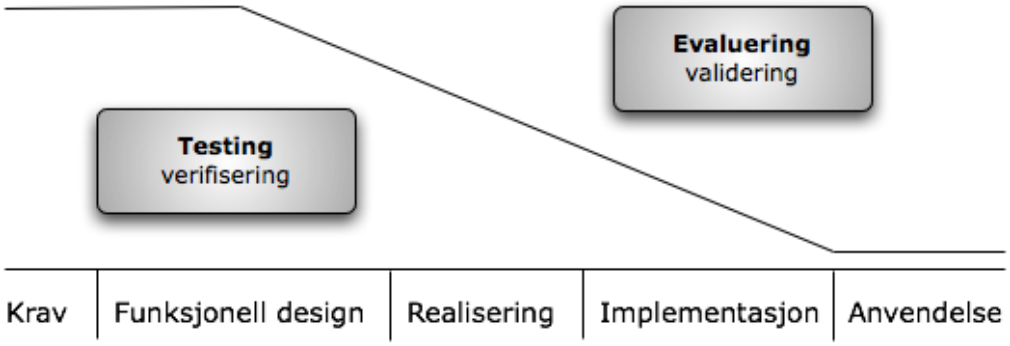
5.4.5 Regresjonstesting

Regresjonstesting er ikke et bestemt testnivå og vil brukes under flere av de testnivåene som er nevnt ovenfor. Regresjonstesting er testing av tidligere testet kode, der det er utført endringer i programmet (Graham et al. 2007). Denne testingen gjøres for å kontrollere om endringene har resultert i nye feil eller at skjulte feil dukker opp, i andre deler av koden. Vanlige endringer ved programvareutvikling er enring i koden eller miljøet programmet kjører i, og det er i nettopp disse tilfellene at regresjonstesting er mest vanlig (Graham et al. 2007)

5.5 Verifisering og validering

Når man tester et system er det viktig at alle brukerens krav til funksjonalitet i systemet er oppfylt. Testing av dette blir en validering (Tian 2005). Samtidig er det også viktig at systemet utfører funksjonen sin riktig med tanke på korrekte utregninger og pålitelighet. Denne formen for testing kalles verifisering.

Koomen og Pol bruker uttrykkene testing og evaluering om henholdsvis verifisering og validering, og hevder at testvariantene utføres i forskjellige faser i utviklingsprosessen (Koomen & Pol 1999). Dette illustreres i figur 5.5, der man ser at verifisering benyttes i større del i de tidligere fasene, mens validering taes i bruk i de senere faser.



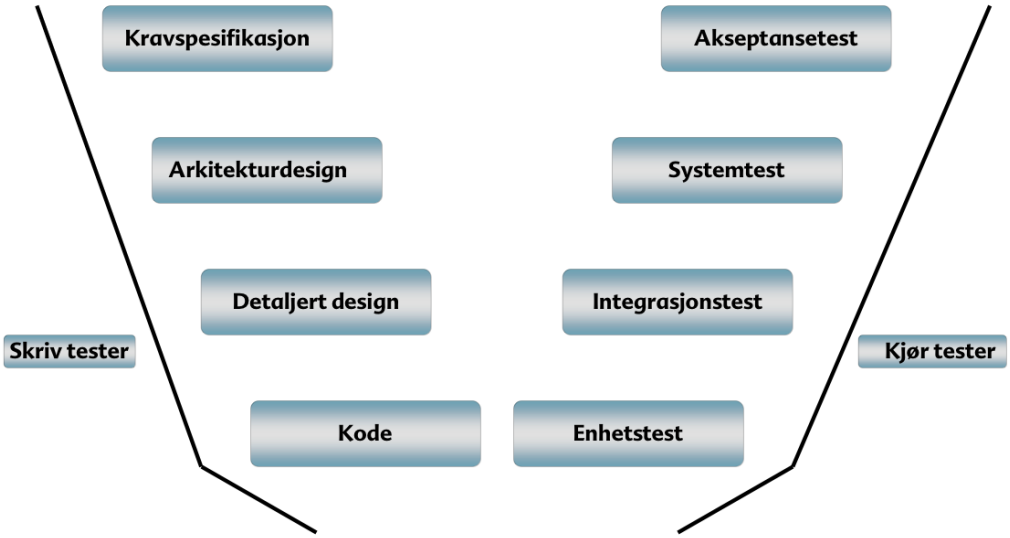
Figur 5.5: Verifisering og validering (Koomen & Pol 1999)

5.6 V-modell for testing

V-modellen for testing er gjengitt i figur 5.6, og er mye brukt til å beskrive når aktiviteter relatert til testing bør finne sted, i tillegg til å vise i hvilken rekkefølge testene finner sted.

Et viktig aspekt ved modellen er at testing ikke bare innebærer å kjøre tester, men også å skrive tester. Dette er vist i figuren som en V, der hver utviklingsaktivitet på venstre side har en korresponderende testaktivitet på høyre side. Modellen er en generell modell og organisasjoner kan ha ulikt navn på aktivitetene, eller gjøre modellen mer detaljert. Det viktigste er at hver utviklingsaktivitet har en tilsvarende testaktivitet på høyre side.

Enhetstest er på det laveste nivået i figuren, hvilket vil si at de som oftest blir skrevet sist, men kjørt først. Øverst finner vi akseptansetest, dette er testen som blir skrevet først, men kjørt sist.



Figur 5.6: V-modell for testing

5.7 Test Maturity Model (TMM)

TMM er et rammeverk som er blitt utviklet av Illinois Institute of Technology som hjelp for testprosessforbedring (Jacobs, van Moll & Stokes 2000). Opprinnelig var ikke testmiljø en del av rammeverket, og dette ble kommentert som en mangel i modellen.

> "Also missing in the TMM is explicit attention for test infrastructure. Test infrastructure referes to test equipment, test systems, test beds, etcetera (Jacobs et al. 2000)"

Her benyttet ikke begrepet testmiljø, men i henhold til definisjonen av testmiljø i kapittel 6, er infrastrukturbegrepet samfallende med testmiljø. I etterkant har flere forfattere lagt til testmiljø som den fjerde prosessen på TMM-nivå 2, blant andre Erik van Veenendaal som vist i figur 5.7.

5.7.1 Hvorfor benytte TMM

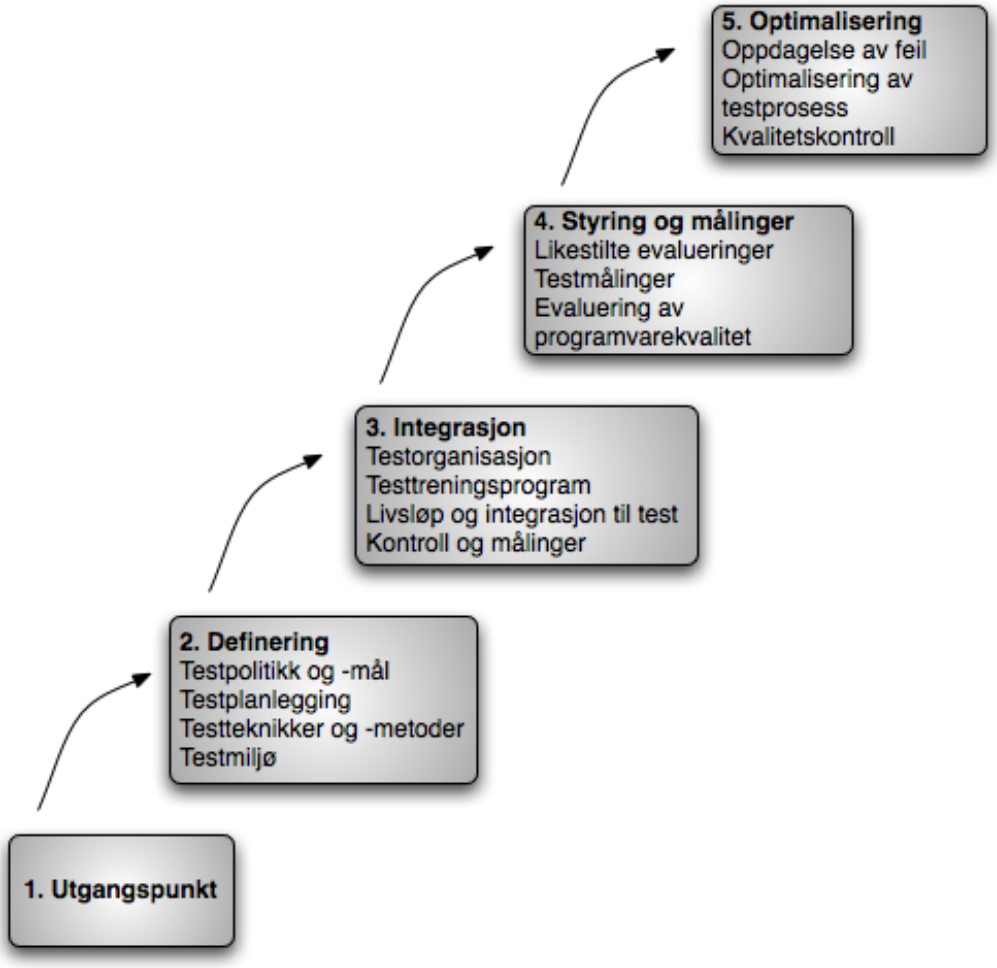
Målet med å benytte et rammeverk for testingen er å forbedre testprosessen (Burnstein 2003). Det finnes i dag flere rammeverk for utvikling av programvare, men flere av disse tar ikke hensyn til de utfordringene man finner i testprosessen. Noen av utfordringene man møter i testprosessen vil gjerne være de samme som i andre faser av utviklingsprosessen. For eksempel vil det å følge tidsplaner være viktig i de fleste faser. Det vil likevel være enkelte utfordringer som er spesielt viktig i testprosessen. (Burnstein 2003) nevner noen av utfordringene:

- Stadig behov for programvare av høy kvalitet med svært få feil. Å kunne bedømme kvaliteten på programvaren er ikke like enkelt.
- Testing er en viktig delprosess i utviklingen av programvare.
- Eksisterende modeller for programvareutvikling og andre forbedringsmodeller har ikke dekket utfordringer som gjelder test.

Det er altså på bakgrunn av slike utfordringer at man kan se nytten av TMM. Ved å bruke denne modellen hevder Burnstein at det er mulig å oppnå:

- Forbedret programvarekvalitet.
- Forbedret mulighet til å følge budsjett- og tidsplaner.
- Forbedret planlegging.
- Forbedret mulighet til å bedømme testmålene (Burnstein 2003).

Med TMM er tanken at man oppnår disse fordelene ved å få et mer bevisst og modent forhold til testprosessen i bedriften, og at man gjennom bruk av retningslinjene i TMM kan unngå problem og forbedre testprosessen.



Figur 5.7: Test Maturity Model (TMM)(van Veenendaal & Swinkels 2002)

5.7.2 Rammeverkets oppbygging

Som vist på figur 5.7 består rammeverket av fem modenhetsnivåer. Disse nivåene reflekterer modenheten på testprosessene (van Veenendaal & Swinkels 2002).

For hvert nivå er det definert et antall prosessområder, relaterte aktiviteter innenfor testprosessen, for eksempel testplanlegging på nivå to på figuren. Når disse aktivitetene blir utført på en god måte, vil de bidra til å forbedre testprosessen. De fem nivåene i rammeverket vil støtte en organisasjon i å bestemme modenheten til testprosessene sin, og å identifisere neste forbedringstiltak som er nødvendig for å oppnå en høyere grad av testmodenhet (van Veenendaal & Swinkels 2002).

Å bestemme modenheten til organisasjonen gjøres ved å gjennomføre en større undersøkelse av organisasjonen. Spørsmålene i undersøkelsen er definert av TMM, og danner grunnlaget når man så skal bestemme hvilket nivå organisasjonen befinner seg. Det er viktig at alle aktivitetene for et gitt nivå, i tillegg til nivåene under, er dekket. En organisasjon som mangler en testaktivitet på nivå tre, vil ikke kunne nå nivå tre eller høyere før aktiviteten er innført i organisasjonen (Burnstein 2003). Disse aktivitetene som skal tilføres er definert av TMM og vil danne grunnlaget for hvordan man utvikler testprosessen i organisasjonen.

5.8 Automatiserte tester

Flere og flere tester som tidligere ble kjørt manuelt er i dag automatisert. Mens manuelle tester utføres av testere, kan automatiserte tester kjøres ved å bruke testvare, en samling filer man trenger for å gjennomføre en automatisk test, og utvørelsen av tester kan dermed bli mindre arbeidskrevende for testerne (Perry 2006). En viktig del av testvaren er et testskript.

5.8.1 Testskript

Et testskript er data og instruksjoner med formell syntaks som blir brukt av et verktøy for automatisk testing. Testskriptet blir vanligvis lagret i en fil. Et testskript kan implementere ett eller flere testcase, navigering-, oppsett- eller nedriggingsprosedyrer og verifisering. Et testskript kan være på en form som også kan benyttes til manuell testing. Testinndata og forventet utdata kan enten være inkludert som en del av testskriptet, eller det kan være i en seperat fil eller database (Fewster & Graham 1999).

5.8.2 Fordeler ved automatisk testing

Automatiseringen har gitt store fordeler. Spesielt gjelder dette produktiviteten der testerne får frigitt tid som tidligere ble brukt til å utføre enkle men

tidkrevende oppgaver (Vogel 1993). Høyere arbeidstilfredshet er en fordel som følger av dette, ettersom arbeidsoppgavene endrer seg og det blir mer tid til å utvikle gode tester. Dette resulterer igjen i bedre produktkvalitet, da man får utført flere tester på kortere tid, og kortere produksjonstid, da testingen går fortere. Fordelene med automatiserte tester er dermed økt produktivitet, arbeidstilfredshet, produktkvalitet, og kortere produksjonstid (Vogel 1993).

Dette samsvarer med Fewster og Graham som mener at fordelen med testautomatisering hovedsaklig er at man kan utføre testoppgaver mer effektivt enn ved manuell testing (Fewster & Graham 1999). Andre fordeler som nevnes er bedre bruk av ressurser, både i form av tid som frigis, og at automatiske tester kan kjøres over natten, gjenbruk av tester og utførelse av tester som vanskelig eller umulig kan gjennomføres manuelt (Fewster & Graham 1999).

Eickelmann og Richardson påpeker at manuelle tester ofte har en høy feilrate, tidsbruk og kostnad i forhold til automatiserte tester (Eickelmann & Richardson 1996). Ved å automatisere testene, gjennom bruk av testmiljø, mener de at man vil redusere kostnader, forbedre testpresisjonen, forbedre programkvaliteten ved at flere tester oppdages og rettes, og øke gjenbruken av tester.

Fewster og Graham oppsummerer med at målet er at testautomatisering skal, ved å la grundigere tester utføres enklere, føre til en økning i kvalitet og produktivitet (Fewster & Graham 1999).

5.8.3 Ulemper ved automatisk testing

Selv om automatiserte tester har mange fordeler framfor manuelle tester er det likevel en del fallgruver. Ofte blir automatisering av tester sett på som den store løsningen på et testproblem, og det føre til uventede problem. For eksempel kan man ha urealistiske forventninger med hva som kan løses ved bruk av automatiske tester, automatisering kan gi en falsk trygghetsfølelse, og vedlikehold av testene kan ta mye tid (Fewster & Graham 1999). Ved å ta hensyn til ulempene mener Fewster og Graham at man kan unngå dem (Fewster & Graham 1999) .

Flere av ulempene går på organisatoriske faktorer under innføring av automatiserte tester. De tekniske problemene som nevnes er få i forhold og man får et inntrykk av at man vil lykkes dersom man bruker en skeptisk tilnærming og ikke blir for naivt positiv (Fewster & Graham 1999).

Black er også skeptisk til at automatisering blir gjort til svaret på alle problem (Black 2002) . Han understreker at man bruker en god del tid til manuelle prosesser innen oppsett og nedtaking av testprogram for å gjennomføre automatiske tester. Dette er tid som ellers kunne vært brukt til å gjennomføre tester. I tillegg tar automatiserte tester ofte lengre tid å lage enn manuelle tester (Black 2002), og man bør derfor kun lage automatiserte tester når denne tiden kan spares inn under testprosessen.

På bakgrunn av de ulempene automatiserte tester har, i følge dette avsnittet, kan man konkludere med at man bør ta forhaåndsregler og ikke se på automatisering av tester som løsningen på testproblemer, selv om automatiserte tester bringer mange fordeler. med seg

5.9 Sammendrag

Dette kapittelet har tatt for seg teori om programvaretesting. Det viktigste for resten av oppgaven er delkapittel 5.4 som omhandler testnivå, og presentasjonen av TMM i delkapittel 5.7. Testnivå er et viktig konsept i teorien om testmiljø og derfor også et viktig tema i intervjuene vi gjennomfører og videre i analysen. TMM beskrives ytterligere som en teoretisk modell for testnivå i kapittel 7.

De øvrige temaene som er diskutert i kapittelet er først og fremst bakgrunnsinformasjon på området testing, og danner grunnlaget for å forstå domenet oppgaven handler om. For eksempel representerer automatiserte tester en ny måte å arbeide med testing på, mens å avgjøre dekningsgrad er en tradisjonell og kjent utfordring innenfor testing.

Kapittel 6
Testmiljø

I dette kapittelet går vi nærmere inn på hovedtemaet for oppgaven, nemlig testmiljø. Kapittelet bygger på begreper og tema rundt testing som ble introdusert i kapittel 5, og danner det teoretiske fundamentet for resten av oppgaven.

6.1 Definisjoner

Denne oppgaven omhandler testmiljø og metodikker for testmiljø, og for å klargjøre hva vi legger i begrepene, defineres de i dette avsnittet.

6.1.1 Testmiljø

Det finnes flere definisjoner av begrepet testmiljø, og det varierer hvilke komponenter som er dekket av begrepet. I det følgende gjengir vi noen definisjoner på testmiljø og forslag til hvilke komponenter et testmiljø består av.

> "Test environment. The conditions that management has put into place that both enable and constrain how testing is performed. The test environment includes management support, resources, work processes, tools, motivation, and so forth." (Perry 2006)
>
> "Test environment. An environment containing hardware, instrumentation, simulation, software tools, and other support elements needed to conduct a test. [After IEEE 610]" (van Veenendaal 2005)
>
> "Test environment (BS 7925-1): A description of the hardware and software environment in which the tests will be run, and any other software with which the software under test interacts when under test including stubs and test drivers." (Schaefer 2005b)
>
> "The test environment is comprised of all the conditions, circumstances, and influences surrounding and affecting the testing of software. The environment includes the organization's policies, procedures, culture, attitudes, rewards, test processes, test tools, methods for developing and improving test processes, management's support of software testing, as well as any test

> labs developed for the purpose of testing software and multiple operating environments." (Certified Software Tester 2006a)
>
> "The platform tests are executed at." (Schaefer 2005b)
>
> "...facilities needed for the execution of tests. (...) This environment comprises the following key components:
>
> - hardware
> - software
> - means of communication
> - facilities for the construction and use of files
> - procedures" (Pol et al. 2002)
>
> "A test environment is made up of the tools that you use to run your tests and report the results, together with the processes and policies for how to use the tools" (Doar 2005)
>
> "Test environment specifications
>
> 1. Hardware
> 2. Software
> 3. Databases
> 4. Network / communication
> 5. Security
> 6. Tools
> 7. Publications, literature, background material
> 8. Procedures for set-up, initializing, post-processing, backup, restore, use, maintenance" (Schaefer 2005b)

Som vi ser, er definisjonene litt ulike med hensyn på hva et testmiljø består av. Definisjonene er enige om at både programvare og maskinvare er en del av testmiljøet. Noen av definisjonene inkluderer støtte fra ledelsen og organisasjonspolikk, mens andre mener dette ligger utenfor testmiljøet.

I denne oppgaven velger vi å følge Hans Schaefers gjengivelse av BS 7925-1, en ordliste over ord og uttrykk i testing, publisert av the British Standard Institute (BSI), og oversetter denne definisjonen til:

> "Testmiljø. (BS 7925-1): En beskrivelse av maskinvare- og programvaremiljøet testene kjøres i, og annen programvare som programvaren under test samhandler med når den testes, inkludert stubber og testdrivere. "

Et testmiljø varierer i størrelse og omfang alt ettersom hvilket system som skal testes, eller hvilket testnivå man befinner seg på. Et enkelt en-brukers-program trenger bare en enkelt maskin som testmiljø, mens store, omfattende systemer som for eksempel et hotellbestillingssystem på internett krever flere klienter og tjenere koblet sammen i nettverk, med lastgeneratorer og målesystemer.

6.1.2 Testmiljømetodikk

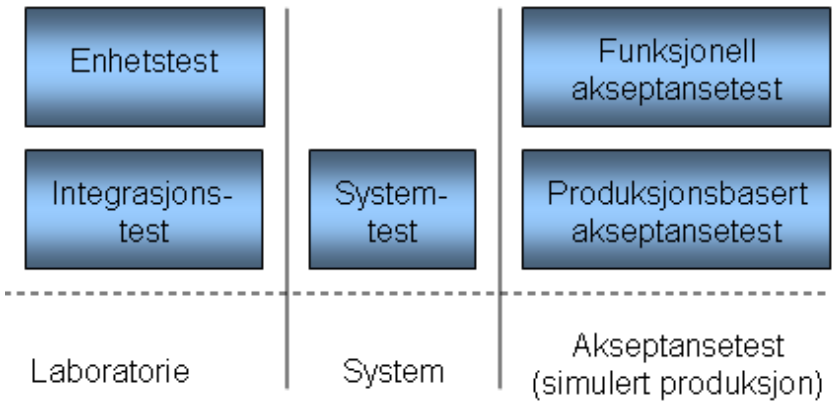
Ordet testmiljømetodikk er sammensatt av de to delordene testmiljø og metodikk. Testmiljø er definert i avsnittet ovenfor. Når det gjelder metodikk vil vi benytte definisjonen fra Kunnskapsforlagets ordbøker

> "Metodikk er læren om metoden for et visst arbeidsområde." (Kunnskapsforlaget 2007)

Her er metode definert som en planmessig framgangsmåte (Kunnskapsforlaget 2007). En testmiljømetodikk er altså læren om hvilken planmessing framgangsmåte man har for å håndtere testmiljø.

6.2 Undergrupper av testmiljø

Hvilket testmiljø man bruker er avhengig av hvilke tester man skal gjennomføre, og hvilket testnivå man er på, som beskrevet i delkapittel 5.4. Det skilles gjerne mellom tre testmiljø, nemlig laboratoriemiljø, systemtestmiljø og akseptansetestmiljø, som vist på figur 6.1 (Pol et al. 2002, Schaefer 2005b).



Figur 6.1: Oppdeling av testmiljø (Pol et al. 2002)

6.2.1 Laboratoriemiljø

Enhetstest og integrasjonstest blir gjerne sett på som lavnivåstester og utføres vanligvis på samme system som de blir utviklet. Utviklingsmiljøet tilbyr ofte standard funksjonalitet for testing, inkludert filer, testverktøy og prosedyrer for versjonskontroll (Pol et al. 2002). Disse verktøyene er ofte tilstrekkelige, og oppsett av testmiljøet og testaktivitetene blir en del av utviklingsprosessen.

I laboratoriemiljøet gjennomføres vanligvis testene av utvikleren som skriver koden, men det kan også være andre utviklere eller prosjektlederen som har ansvaret for dette (Pol et al. 2002).

6.2.2 Systemtestmiljø

Målet med systemtestmiljøet er å teste både de tekniske og funksjonelle aspektene ved systemet (Pol et al. 2002). Dette anbefales å foregå i et kontrollert miljø, ettersom det er nødvendig å kunne repetere testene og å la individuelle delsystem testes uavhengig av andre delsystem. Bruk av testdatabaser skaper ofte problemer når det gjelder nettopp denne uavhengige testingen fordi flere testere kan bruke dem samtidig, noe som kan få innvirkning på resultatet.

Systemtestmiljøet, som også går under navnet kontrollerbart laboratoriemiljø, gjør det mulig å bruke testverktøy for å bestemme testobjektets funksjonelle kvalitet.

Fasiliteter og prosedyrer

I følge Pol er et problem med systemtestmiljøet at det vokser ukontrollert (Pol et al. 2002). For å sikre kvaliteten på testobjektet vil mange utviklere prøve å oppnå simulerte produksjonsfasiliteter i systemtestmiljøet. Ved mangel på disse fasilitetene og prosedyrer for hvordan dette skal lages, vil resultatet være enorme mengder kopier og store filer, som lagres over en lengre periode. Å holde orden på store datamengder er kostbart og opptar mye lagringsplass. For å unngå dette er det viktig å ha gode fasiliteter og prosedyrer for systemtestmiljøet.

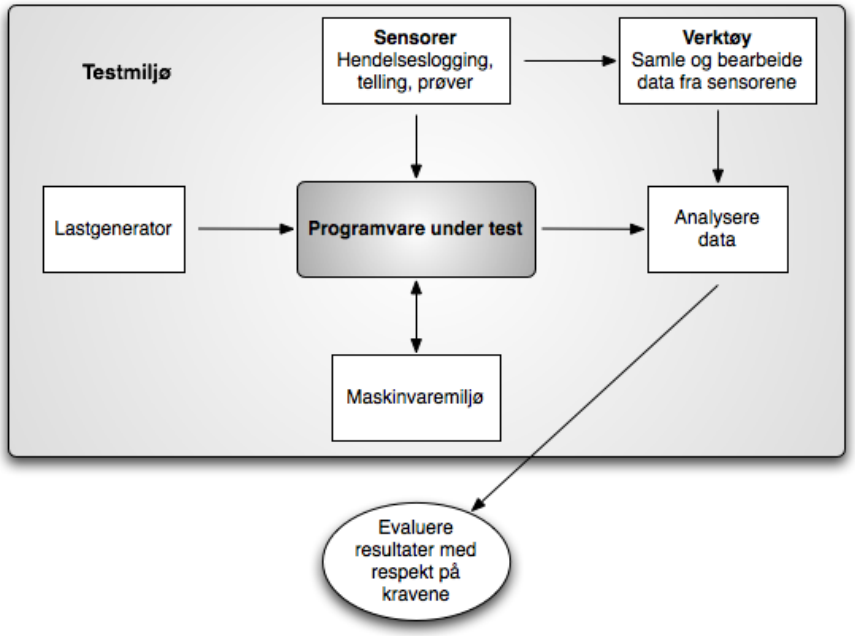
Eksempel på oppsett - ytelsestest

Hvor likt systemtestmiljøet må være det faktiske produksjonsmiljøet, er avhengig av hvilke former for systemtest man ønsker å gjennomføre. Burnstein gjengir et eksempel på ressurser som trengs for å utføre en ytelsestest. Dette er gjengitt i figur 6.2 (Burnstein 2003).

Som vi ser til venstre i figuren, trengs det en form for transaksjoner som inndata til systemet under testing. Dette vil typisk være en lastgenerator, som blir omtalt delkapittel 6.6. Testmiljøet må også omhandle maskinvare systemet samhandler med, også spesielt laboratorieutstyr. For å samle inn ytelsesdata, trenger man sensorer koblet til programmet. Disse sender data til dedikerte verktøy som bearbeider dataene og sender dem videre til et analyseverktøy. Ofte vil man samle inn store datamengder under ytelsestesting, og testerne vil ha vansker med å bearbeide og analysere disse uten verktøy (Burnstein 2003).

6.2.3 Akseptansetestmiljø

Akseptansetesten deles i en funksjonell og en produksjonsbasert del (Pol et al. 2002). I akseptansetestmiljøet foregår dermed både funksjonell akseptansetest og produksjonsbasert akseptansetest. Ved funksjonell akseptansetest



Figur 6.2: Testmiljøkrav til ytelsestest (Burnstein 2003)

blir testobjektet testet for å bestemme om systemet fyller kravene til funksjon og kvalitet ved bruk i simulerte produksjonsfasiliteter og -prosedyrer. (Pol et al. 2002). Ved produksjonsbasert akseptansetest testes det for å se om systemet fyller kravene til kontroll og produksjonsstandard i henhold til last- og ytelsesbehov (Pol et al. 2002). Målet med dette miljøet er å teste systemet, så nært det miljøet det skal brukes i, som mulig. I akseptansetestmiljøet gjennomføres tester på et høyt testnivå av framtidige brukere og ledelse.

Deling av akseptansemiljø

Funksjonell og produksjonsbasert akseptansetest utføres av og til i samme testmiljø, sekvensielt eller frittstående. I andre tilfeller er det behov for to forskjellige testmiljø. Dette gjør akseptansetestmiljø til et flertydig testmiljø. I et produksjonsbasert akseptansetestmiljø vil man kunne risikere at der akseptansetestmiljøet blir en flaskehals. Akseptansetesten gjennomføres på et høyt testnivå, og det er da gjerne et tidspress før levering (Pol et al. 2002). Svært mange produksjonsnære aspekt trenger testing på samme tid og det oppstår konflikter om hvem som skal ha tilgang til akseptansetestmiljøet. Pol mener at dette problemet kan unngåes ved å ha tilgang til et simulert produksjonsmiljø tidligere i utviklingsprosessen (Pol et al. 2002).

6.2.4 Variasjoner av testmiljø

I tillegg til de overnevnte testmiljøtypene er det vanlig med kombinasjoner av testmiljøer (Pol et al. 2002). Dette kan skje ved at man bygger testmiljøet

Testtype	Kvalitetstrekk	Testmiljø
Stresstest	Operasjonell pålitelighet, kontinuitet	Simulert produksjon
Test av ressursbruk	Effektivitet	Simulert produksjon
Produksjonstest	Korrekthet, operasjonell pålitelighet	Simulert produksjon
Gjenoppretting	Vedlikeholdbarhet	Systemtest, simulert produksjon
Sikkerhet	Sikkerhet	Simulert produksjon
Funksjonalitet	Korrekthet, kompletthet	Laboratorie, systemtest, simulert produksjon
Standarder	Sikkerhet, effektivitet	Systemtest, simulert produksjon
Kontroller	Korrekthet, datakontrolerbarhet	Simulert produksjon
Grensesnitt	Korrekthet, kompletthet	Laboratorie, systemtest, simulert produksjon

Tabell 6.1: Testtyper og testmiljø (Pol et al. 2002)

på bakgrunn av hvilke testtyper man bruker eller om man kombinerer testmiljøtyper for å utnytte fordelen med begge.

Oppsett på bakgrunn av testtype

Av og til kan det være ønskelig å fokusere mer på hvilken type kvalitet man vil teste systemet for, og det er da mulig å se på hvilke testtyper som dekker dette. Ved gjennomføring av disse testene vil det være testtypen som setter krav til hvilket testmiljø man trenger. Tabell 6.1 viser hvilke kvalitetstrekk som dekkes av de ulike testtypene og hvordan dette relateres til testmiljøene (Pol et al. 2002).

Kombinasjon av testmiljø

Et eksempel på et kombinert testmiljø er et integrert miljø som kombinerer systemtestmiljø og funksjonell akseptansetestmiljø. Både systemtesten og den funksjonelle akseptansetesten vil teste den funksjonelle kvaliteten på systemet. Forskjellene er testmiljøet testene blir kjørt i og hvor testene kommer fra.

6.3 Etablering, bygging, bruk og vedlikehold av testmiljø

Det er flere punkt som kan være viktig å tenke på når en organisasjon skal etablere, bygge, bruk og vedlikehold et testmiljø. I dette avsnittet oppsummeres de viktigste punktene basert på kilder i teorien.

Avveie hvor nært produksjonsomgivelsene testmiljøet skal være For å muliggjøre realistiske tester, skal et testmiljø være representativt for produksjonsomgivelsen (Pol et al. 2002). Dette medfører at testmiljøet skal bygges så nært produksjonsmiljøet som mulig. I praksis kan dette være dyrt eller umulig å gjennomføre, og en viktig avveing er derfor å avgjøre hvor nært produksjonsmiljøet som er bra nok for å muliggjøre realistisk testing. Dette er en av de vanskeligste temaene ved bygging av testmiljø og vil bli omhandlet i detalj senere.

Også testdata må være realistiske. Dette inkluderer å ha data som dekker hele spekteret av anvendelsen, ikke bare testdata som dekker det viktigste, eller til og med helt tilfeldige deler.

Etablere behovet for testmiljø tidlig I de fleste organisasjoner er det viktig å etablere behovet for et testmiljø tidlig for å ha det tilgjengelig når det trengs (Fewster & Graham 1999). Man må finne ut hvilke testmiljø man har behov for for å gjennomføre testene, og skape aksept fra ledelsen og utviklerne. Et testmiljø bør holdes atskilt fra andre miljø, som for eksempel miljø brukt for utvikling, produksjon og trening, og derfor kan et testmiljø planlegges og bygges parallelt med andre aktiviteter.

Skille testene fra mekanismene brukt til å kjøre dem En viktig egenskap ved et godt testmiljø, er at det skiller testene fra mekanismene som blir brukt til å kjøre dem (Doar 2005). På denne måten trenger man ikke å endre tester selv om man ønsker en annen måte å kjøre testene på. Dette er viktig for å tilrettelegge for gjenbrukbarhet, og dermed spare kostnader.

Rask tilpasning Det må være mulig å tilpasse oppsettet av et testmiljøet raskt (Pol et al. 2002).

Ha oppdaterte prosesser og prosedyrer Dokumentasjon og vedlikehold av prosesser og prosedyrer er vesentlig ved bygging og vedlikehold av testmiljø. Eksempler på prosedyrer og prosesser er:

- Oppdaterte driftsprosedyrer.
- Prosedyrer for sikkerhetskopier og skript.
- Prosedyrer for oppsetting og installasjon.
- Prosedyrer for overføring av systemet under test fra annet miljø til testmiljøet (Schaefer 2005b).

Denne dokumentasjonen må lagres lett tilgjengelig for testerne.

Skille manuelle og automatiske tester Bruker man mekanismer for automatisk testing, bør man skille testmiljøet brukt ved manuell testing fra dette, med mindre man ønsker å kombinere de to tilnærmingene. Årsaken er at man ikke ønsker at de manuelle testene skal bli begrenset av reglene for de automatiske testene (Fewster & Graham 1999).

Mulig å endre dato I noen tilfeller er det nødvendig å utføre tester med en annen dato enn systemets dato, og for å muliggjøre dette, må testeren kunne endre på denne. I praksis er dette ofte vanskelig, ettersom flere programmer kjører på maskinen samtidig. Derfor bør et testmiljø inneholde en individuell fasilitet for å endre dato i systemet (Pol et al. 2002). Et eksempel er det såkalte Y2K-problemet, hvor store datasystemer ble testet for overgangen fra '99 til '00 ved 2000-årsskiftet. I denne situasjonen var det viktig å kunne endre dato.

Lage basis testdata for hver tester Det kan være nyttig å ha et sett av basis testdata som er dokumentert for hver tester og kan brukes som testeren vil, men som blir endret til startverdi ved omlasting av database (Fewster & Graham 1999).

Mulighet for gjenoppretting Det er i følge Schaefer essensielt å tenke gjennom følgende spørsmål:

- Kan testmiljøet bli kopiert og gjenopprettet ved en senere anledning?
- Hvor enkelt og raskt kan dette skje?
- Hvor presist kan det bli gjenopprettet?

Tar man hensyn til og dokumenterer disse spørsmålene allerede ved utformingen av testmiljøet, etablerer man rutiner for å gjenopprette systemet ved et systemkræsj. Denne gjenopprettingen basert på sikkerhetskopier må også testes (Schaefer 2005b).

Dersom kontinuitet er veldig viktig for testing, og man ikke har anledning til å vente på at systemer skal gjenopprettes, må alternativer for testmiljøet bli arrangert (Pol et al. 2002), for eksempel duplikatservere.

Stabilt miljø For at sammenhengende tester skal kjøre under samme forhold, er det viktig at testmiljøet er stabilt. Endringer i testmiljøet, for eksempel endringer maskinvare eller programvare, testobjekter eller prosedyrer skal bare utføres etter at testledelsen har gitt klarsignal til dette(Pol et al. 2002).

6.4 Testing og kvalitetssikring av testmiljø

Etter at testmiljøet er satt opp, er det viktig å forsikre seg om at det fungerer, og at kvaliteten er tilfredsstillende. Hans Schaefer presenterer en liste på fire punkt på hvordan man kan teste et testmiljø (Schaefer 2005b):

1. Røyktest - Sjekker tilgjengeligheten til systemet under test.

2. Manuell sjekk av testmiljøet mot dokumentert konfigurasjonsbehov. Bekrefter at miljøet er riktig, men garanterer ikke at det vil fungere.
3. Automatiserte verktøy
   (a) Verktøy for konfigurasjonskontroll.
   (b) Verktøy for datastyring.
   (c) Verktøy eller skript for oppdagelser angående miljøet.
   (d) Verktøy for sikkerhetskopi og gjenoppretting.
4. Automatiske miljøtester - Bekrefter at miljøet er oppe og kjører.

Det er også viktig å måle kvaliteten på testmiljøet. For å kunne si noe om dette, må man beregne flere parametre. Schaefer foreslår tre ting som bør måles:

1. Hvor mye nedetid har testmiljøet?
2. Hvor mange feil og defekter skyldes problemer med testmiljøet?
3. Hvor mye arbeid kreves for å sette opp, gjenopprette og arbeide på miljøet? (Schaefer 2005b)

Disse parametrene må beregnes etter at testmiljøet er bygget. For å måle nedetid bør man registere nedetiden til testmiljøet og evaluere denne mot forventet oppetid. Deretter kan man for eksempel regne ut prosent av tiden testmiljøet er nede. Å måle hvor mange feil og defekter som skyldes problemer med testmiljøet er vanskelig å beregne, men om mulig bør dette estimeres. De gangene man identifiserer problemer med testmiljøet bør man registrere dette, men det kan likevel forekomme skjulte feil.

Ettersom testverktøy bruker ressurser, vil de ha innvirkning på testmiljøet. Testobjektet vil da ofte oppføre seg annerledes enn i virkeligheten. Dersom det er mulig, er det lurt å kartlegge omfanget av problemet.

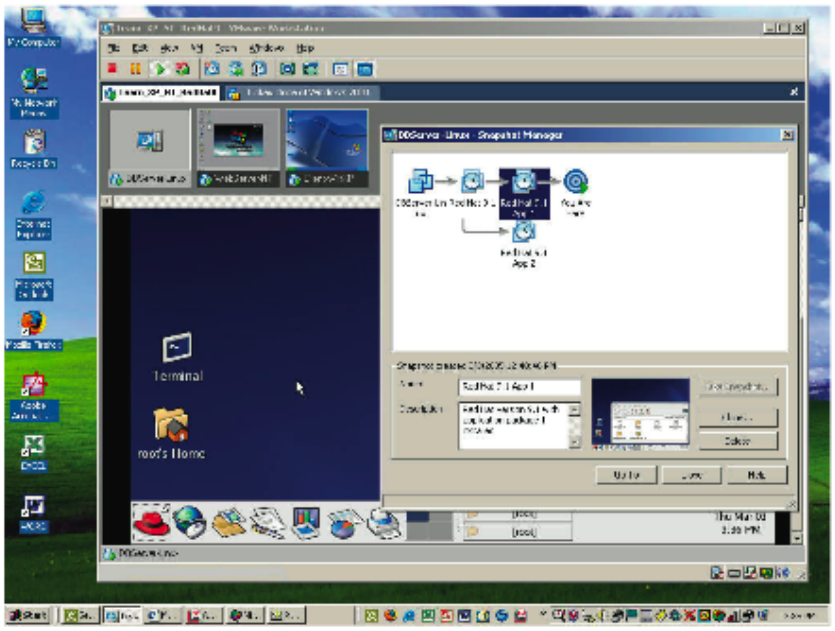
6.5 Virtuelle testmiljø

I mange situasjoner er det tilstrekkelig å simulere klienter, tjenere og hele nettverk på en eller et lite antall fysiske maskiner. Har man for eksempel et system som kjører på 10 servere i produksjon, kan det være tilstrekkelig å virtualisere disse på to fysiske servere. For å oppnå denne funksjonaliteten, benyttes virtualiseringsverktøy.

6.5.1 Hva er virtualisering

Virtualisering er et abstraksjonslag som skiller den fysiske maskinvare fra operativsystemet, og på denne måten gjør det mulig å kjøre flere virtuelle maskiner med ulike operativsystemer på samme fysiske maskin. De virtuelle maskinene er isolert fra hverandre og vertsmaskinen, slik at dersom en virtuell maskin kræsjer, påvirker ikke dette andre (Shankland 2006). Operativsystemet

som installeres først kalles gjerne vertsmaskin, mens de andre virtuelle maskinene kalles gjester.



Figur 6.3: Virtualisering av flere operativsystem på samme datamaskin (VMware 2007b)

Figur 6.3 viser VMwares verktøy for virtualisering, Workstation. Her kjører en Linux databasetjener, en Windows webtjener og en Windows klientmaskin sammen på en maskin, og ved hjelp av brukergrensesnittet til VMware er det enkelt å skifte mellom systemene. Vi vil si mer om Workstation i kapittel 8.

6.5.2 Fordeler ved virtualisering

Fordeler for testere er blant annet reduserte kostnader, ettersom man slipper å kjøpe maskinvare ved operativsystem man ønsker å teste produktet på. Det er også enklere å kjøre tester på samme maskin framfor å ha dedikerte maskiner for hvert operativsystem. Det er mulig å fjernstyre alle virtuelle tjenere fra en maskin (Schaefer 2005b).

Man kan også simulere at en fysisk maskin kjører flere tjenere og klienter og dermed slippe å sette opp flere maskiner for å teste distribuerte systemer. På denne måten kan man teste nettverk på en enkel måte.

6.5.3 Begrensninger ved virtualisering

Et testmiljø basert på virtualisering, trenger mye hurtigminne og stor lagringskapasitet (Schaefer 2005b). Prisen på minne har imidlertid falt mye, så dette er ikke lenger den største investeringenn ved kjøp av maskinvare. Virtuelle

testmiljø er tilstrekkelig for å teste funksjonalitet, men det er ikke mulig å teste ytelsen på denne måten (Schaefer 2005b).

6.6 Simulering og emulering

Verktøy for simulering og emulering kan være nyttige verktøy ved bygging av testmiljø. Disse verktøyene kan brukes til å erstatte programvare- eller maskinvarekomponenter som mangler, er utilgjengelig eller er for kostbare å anskaffe (Burnstein 2003). På denne måten kan systemer som baserer seg på dyre delsystemer testes på en billigere måte. Simulering kan også være nyttig tidlig i utviklingsprosessen, før man har utviklet fullstendig fungerende system. Her kan simulering hjelpe til å verifisere høynivå designidéer eller egenskaper ved systemarkitekturen, slik at man unngår ressurskrevende implementering og å gjøre arbeid om igjen (Tian 2005). En annen undergruppe er lastgeneratorer, verktøy som genererer store menger testdata til for eksempel test av transaksjonsbaserte systemer og telekommunikasjonssystemer (Burnstein 2003).

6.7 Sammendrag

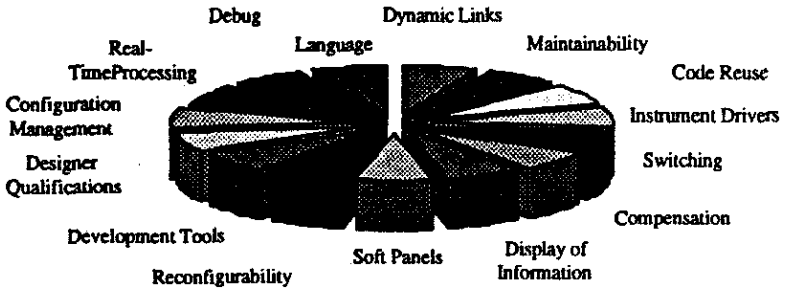
Dette kapittelet har tatt for seg testmiljø som er hovedtemaet for oppgaven, og hele kapittelet er derfor høyst aktuelt. Undergrupper av testmiljø vil diskuteres i analysen, relatert til hvilke testmiljø som benyttes i bedriftene. Delkapittel 6.3 om bygging, etablering og vedlikehold danner et teoretisk fundament for diskusjonen i kapittel 17 og for retningslinjene i kapittel 18. Virtualisering av testmiljø vil diskuteres ytterligere i kapittel 8 der aktuelle virtualiseringsverktøy blir gjennomgått, før vi i analysedelen beskriver hvordan bedriftene benytter virtualisering.

Kapittel 7
Modeller for testmiljø

Som nevnt i kapittel 4 har vi funnet lite tilgjengelig litteratur på området testmiljø og enda mindre om metodikker for testmiljø. I dette kapittelet presenterer vi det vi har funnet.

7.1 Artikkel av Richard Caesar

Richard E. Caesar har publisert en artikkel i IEEE om designmetodikk for å lage et effektivt testmiljø (Caesar 1997). I artikkelen hevder forfatteren at et testmiljø må være informativt, allsidig og vedlikeholdbart for å støtte stramme utviklings- og integrasjonstidsrammer, redusere kostnader ved brukerstøtte og være utvidbart uavhengig av testnivå. Dersom et testmiljø skal støtte disse kravene, må det inneholde elementer som kan utvikles uavhengig, men likevel fungere i et tett integrert miljø. Forfatteren deler testmiljøelementer i to kategorier, testutviklingsmiljø og testutførelsesmiljø.



Figur 7.1: Konsept i testmiljø (Caesar 1997)

Artikkelen deler designet av et testmiljø i moduler, som vist figur 7.1, og konkluderer med at ved å dele opp designet av testmiljøet, og adressere hver designmodul inkrementelt, kan man designe et effektivt testmiljø (Caesar 1997).

7.1.1 Diskusjon av Richard Caesars artikkel

Artikkelen definerer ikke begrepet testmiljø eksplisitt. Det virker som om forfatteren inkluderer utviklingen av tester i testmiljøet ettersom testutviklingsmiljøet er ett av to testmiljøelementer. Vår definisjon av testmiljø definerer ikke utviklingen av tester som en del av testmiljøet. I tillegg fokuserer artikkelen hovedsakelig på testmiljøelementenes kortsiktige og langsiktige innvirkning på kostnader forbundet med et testsystem. Kostnad ved testmiljø er ikke et fokusområde for vår oppgave, likefullt er denne artikkelen den mest relevante fra litteratursøket i artikkeldatabaser.

7.2 Certified Software Tester (CSTE)

Vedlegg D gjengir en retningslinje for bygging av testmiljø, utgitt av CSTE.

7.2.1 Om CSTE

CSTE er en sertifisering som tilbys av organisasjonen Software Certifications, administrert av Quality Assurance Institute Worldwide (QAI). QAI ble grunnlagt i USA i 1980 og har som mål å øke kvalitet og produktivitet og fremme effektive metoder for prosesstyring i IT-bransjen. I dag er QAI en internasjonal medlemsorganisasjon, og tilbyr konferanser, seminarer og sertifiseringer (QAI Worldwide 2006).

CSTE er en av tre sertifiseringer som tilbys, de andre to er Certified Software Project Manager (CSPM) og Certified Software Quality Analyst. Sertifiseringen stiller krav til kunnskap på ti definerte områder, og kandidater blir eksaminert i disse. Kunnskapskategori to handler om å bygge testmiljø, og er gjengitt i vedlegg D.

7.2.2 Gjennomgang av kunnskapsområdet testmiljø

CSTE finnes i to utgaver, en fra 2005 og en fra 2006. Kunnskapsområdet testmiljø er omskrevet mellom de to utgavene, og vi har valgt å gjengi begge utgavene fordi vi finner interessante aspekter til vår oppgave både i 2005- og i 2006-utgaven.

Vedlegg D.1 gjengir 2005-utgaven. Kunnskapsområdet deles i seks fokusområder, med punktvis beskrivelse innenfor hvert område. Vi gjengir et kort sammendrag, og henviser ellers til vedlegget.

Teststandarder Testeren skal ha oversikt over teststandarder, både eksterne og interne.

Testmiljøkomponenter Testeren skal kunne utvikle testprosesser, bruke testverktøy og metoder for testutvikling, samt utvikle et testmiljø som

simulerer virkeligheten, inkludert muligheten til å lage og vedlikeholde testdata.

Testverktøy Testeren skal ha kompetanse på verktøy, blant annet verktøy for regresjonstesting, ytelse og for å kartlegge dekningsgrad.

Kvalitetskontroll Testeren må kunne skille mellom aktiviteter som skal forbedre utviklingsprosessen for å hindre innføringen av feil og aktiviteter som skal finne og rette feil, samt kunne analysere innsamlede data for å foreslå feilrettingstiltak.

Bygge arbeidsprosesser i testmiljøet Testeren skal kunne forstå konseptet med arbeidsprosesser, kunne bygge prosesser for testmiljø, sjekke at prosessen blir utført korrekt og analysere prosessens kvalitet i henhold til syv definerte punkter, se vedlegg D.1.

Tilpasse testmiljøet til forskjellige teknologier Testmiljøet må kunne teste teknologiene brukt i programmet under test, som klient/tjener-oppsett, datavarehus og internettbaserte system

Vedlegg D.2 gjengir 2006-utgaven av kunnskapsområdet testmiljø (Certified Software Tester 2006b). Denne er omskrevet i forhold til 2005-versjonen ved at den er oppdelt i større fokusområder, og definerer testmiljø eksplisitt. Sertifiseringen deler forståelsen om bygging av testmiljø opp i tre deler; kunnskap om testprosessutvelgelse og -analyse, testverktøy og støtte fra ledelsen for effektiv testing.

Testprosessutvelgelse og -analyse Testeren skal ha kunnskap om utvelgelse av testprosess og analyse, og utvikle et testmiljø som simulerer virkeligheten, inkludert muligheten til å lage og vedlikeholde testdata

Testverktøy Testeren må kunne bruke og forstå testverktøy og metoder, og forstå kunnskapene som trengs for testutvikling, utførelse, sporing og analyseverktøy.

Støtte fra ledelse Ledelsen må tilby retningslinjer for testing, og følge opp disse, samt gi testing en naturlig plass i organisasjonen.

7.2.3 Diskusjon av CSTE

Definisjon av testmiljø mangler i 2005-utgaven, mens den er på plass i versjonen fra 2006. Vi ser at denne definisjonen av testmiljø er mer omfattende enn vår da den tar for seg tema som organisasjonskultur, holdninger og belønning. Dette fører til at denne metodikken er mindre egnet i henhold til vår definisjon, ettersom vi her får en metodikk som tar for seg andre tema enn det vi ser etter. Det er likefullt deler av metodikken som er relevant da det er overlapping mellom vår og deres definisjon av testmiljø.

Begge retningslinjene trekker inn testverktøy i stor grad, og vi er enig i at testverktøy har en sentral rolle i et testmiljø. Alle typer verktøy som nevnes i

CSTE er derimot ikke omhandlet med vår definisjon, og det er derfor viktig å ta dette i betraktning når man benytter seg av CSTE.

7.3 Mal for testmiljø

Hans Schaefer er en av Norges ledende personer innenfor området testing. Han har blant annet holdt flere seminar og skrevet bøker om testing, og er leder for den norske avdelingen av International Software Testing Qualifications Board (ISTQB). Han har laget en mal for spesifikasjon av et testmiljø, som er gjengitt i vedlegg E.

Malen er en punktvis framstilling av aspekter ved testmiljøet det er viktig å spesifisere. I begynnelsen av malen registeres når testmiljøet må være tilgjengelig, når det er ledig, og hvilke krav som stilles til tilgjengelighet utenom normale arbeidstider. Deretter spesifiseres blant annet tjener- og klientprogramvare grundig, med versjoner av operativsystem, mellomvare, databaser og utviklingsverktøy, i tillegg til nettverk og diskkonfigurasjoner. Det anbefales også å ha med en beskrivelse av hvordan testmiljøet brukes, blant annet bør man registrere hvordan programvaren under test, databasen og nettverket startes og stoppes.

7.3.1 Diskusjon av mal for testmiljø

Malen viser ikke hvordan testmiljø bør bygges, men fokuserer på hvilke aspekter ved testmiljøet som må dokumenteres. Dette gjelder særlig tekniske aspekter. Malen definerer ikke begrepet testmiljø, men da vår definisjon av testmiljø er Scheafers omskrivning av britisk standard og den samfaller med innholdet i denne malen, ser vi ingen problemer med dette.

7.4 Testmiljø i Test Maturity Model (TMM)

TMM ble presentert i delkapittel 5.7 som et rammeverk til hjelp for testprosessforbedring. Dette rammeverket inneholder krav som testmiljø må oppfylle for å nå et høyere modenhetsnivå på TMMs modenhetsskala, som vist i figur 5.7.

Som nevnt i delkapittel 5.7, var testmiljø ikke en del av den opprinnelige modellen slik den ble publisert av Illinois Institute of Technology. Flere forfattere har imidlertid lagt til testmiljø som en fjerde prosess på nivå 2, og vi velger å ta utgangspunkt i denne reviderte modellen.

7.4.1 Hovedmål

Vedlegg F gjengir retningslinjer for TMM publisert av interesseorganisasjonen for TMM (van Veenendaal 2006). Retningslinjene tar for seg aktivitetene som må adresseres for prosessområdet testmiljø. Målet med et testmiljø er å etablere og vedlikeholde et integrert program- og maskinvaremiljø som muliggjør kjøring av tester på en håndterbar og repeterbar måte (van Veenendaal 2002). Målet deles i tre delmål:

1. Testmiljø blir spesifisert tidlig, og de er tilgjengelig i tide i prosjektet.
2. For høyere testnivåer er testmiljøet så likt virkeligheten som mulig.
3. Testmiljø blir styrt og kontrollert ved hjelp av dokumenterte prosedyrer.

Spesifikasjoner bør lages tidlig i testprosjektet. Spesifikasjoner må gjennomgås for å forsikre seg om at de er korrekte, passende, gjennomførbare og representativ for virkeligheten. Ved å lage spesifikasjonen tidlig, har man bedre tid til å utvikle spesielle simulatorer, samt stubber og drivere som trengs i miljøet (van Veenendaal 2002).

TMM sier at tester må kjøre under så virkelighetsnære forhold som mulig, det vil si i et produksjonslikt miljø. Dette gjelder først og fremst høynivåstester (van Veenendaal 2002). Modellen presiserer ikke nærmere hvor likt "så likt som mulig" er, men dette er en avveing bedriften må foreta.

Et testmiljø kan forandre seg i løpet av prosjektets levetid, for eksempel grunnet maskinvareendringer eller endringer på testobjektet. For å adressere disse problemene krever TMM gjennomført konfigurasjonsstyring av testmiljøet (van Veenendaal 2002).

7.4.2 Antall testmiljø per testnivå

En aktuell problemstilling er om det er nødvendig med ett miljø per testnivå. TMM fastslår at dette kan være svært dyrt, og at man derfor bør bruke testmiljøene så effektivt som mulig. Et forslag kan være å dele miljø mellom utviklere og testere, men dette krever god styring og kontroll for å unngå problemer som at miljøet er opptatt når noen ønsker å bruke det, eller at det ikke er tilbakestilt til starttilstand (van Veenendaal 2002).

7.4.3 Nødvendige aktiviteter i TMM

I vedlegg F gis en punktvis og detaljert framstilling av hvilke aktiviteter som bør dekkes i prosessområdet testmiljø. I dette avsnittet vil vi kort oppsummere hovedpunktene.

Ressurser Nok ressurser må være tildelt for å spesifisere, styre og kontrollere testmiljøet. Dette inkluderer trenet personell, styringsverktøy for testmiljøet, tid og ressurser for implementering av testmiljøet og nok tid til å utvikle stubber og drivere for lavnivås-testing.

Testmiljøansvarlig En gruppe eller en person er ansvarlig for å styre og kontrollere testmiljøet. Ansvarsområdene dette innebærer er listet opp, og inkluderer implementasjon av testmiljøet, å løse tekniske problem i miljøet og å være ansvarlig for testmiljøets oppetid.

Prosedyre for testmiljøspesifisering Prosedyren inkluderer blant annet å definere krav til testmiljøet, fordele ansvar og arbeidsoppgaver og estimere kostnader.

Prosedyre for styring og kontrollering av testmiljøet Denne prosedyren bør inkludere sikkerhetskopier og gjenoppretting, endringshåndtering og konfigurasjonsstyring. Foreslåtte endringer skal gjennomgås av testansvarlige. Prosedyren skal følges opp.

Prosedyre for å sikre tilgjengeligheten til testmiljøet Inneholder reservasjoner for testmiljøet, fordeling av tid mellom testing og utvikling, og dokumentasjon på hvordan testmiljøet skal avsluttet korrekt etter bruk. Prosedyren skal følges opp.

Testmiljøet spesifiseres tidlig i prosjektet Testmiljøspesifikasjonen bør bestå av nettverk- og programvarekomponenter, simulatorer, brukermanualer og verktøy for utvikling av stubber og drivere. Spesifikasjonen bør sees over av både prosjektledelse, tekniske eksperter og andre interessegrupper med hensyn til teknisk riktighet, hensiktsmessighet, hvor virkelighetsnært det er, hvor teknisk og økonomisk gjennomførbart det er og om det kan leveres i tide.

Realistisk testmiljø ved høynivås tester Det operasjonelle testmiljøet bør være i samsvar med spesifiserte krav, og risikoer ved at det ikke stemmer overens med virkeligheten skal være diskutert med ledelsen

Status på testmiljøaktiviteter skal måles Eksempler på målinger inkluderer antall konfliktreservasjoner, antall tester som feiler grunnet testmiljøet og prosentvis hvor stor andel av tiden et testmiljø er tilgjengelig i henhold til spesifikasjon.

Evaluering og granskning av testmiljø En egen kvalitetsgruppe skal granske og evaluere testmiljøet og rapportere til ressursstyring, testledelse og prosjektledelse. Et minimumsnivå er å verifisere at testmiljøspesifikasjoner blir skrevet tidlig i prosjektet i henhold til prosedyrer, at testmiljøet er virkelighetsnært, at tilgjengeligheten er akseptabel og at prosedyrer for styring og kontroll blir fulgt opp.

Evaluering av aktiviteter i testmiljøet Aktivitetene i testmiljøet bør evalueres på periodisk basis for å sjekke om testmiljøet er tilstrekkelig med hensyn på de tekniske, kostnadsmessige og personalmessige ytelsene, samt konflikter og problemer som ikke lar seg løse. Deretter skal en sammendragsrapport distribueres til affiserte parter.

7.4.4 Sjekkliste for testmiljø i nivå 2

Som oppsummering gjengir vi TMMs sjekkliste som oppsummerer hovedpunktene en bedrift må oppfylle i prosessnivået testmiljø.

- En detaljert spesifikasjon av testmiljøet er dokumentert i testplanen.
- Testmiljø blir styrt og kontrollert i henhold til dokumenterte prosedyrer for konfigurasjonsstyring.
- En adekvat back-up- og gjenopprettingsprosedyre eksisterer for testmiljøet og dets databaser.
- For høynivåstester er testmiljø så likt som mulig virkeligheten (van Veenendaal 2002).

7.4.5 Diskusjon av TMM

TMM beskriver grundig retningslinjer for testmiljø, og inkluderer detaljerte punktvise framgangsmåter og sjekklister. Heller ikke TMM definerer testmiljø eksplisitt, men ut fra retningslinjene virker det som om det TMM legger i testmiljøbegrepet i stor grad samsvarer med vår definisjon. Basert på grundigheten testmiljø er behandlet med, virker det som om TMM anerkjenner viktigheten av et etablert testmiljø ved testing, selv om testmiljø ikke opprinnelig var en del av TMM.

7.5 Evaluering og sammenligning av modellene

De fire modellene, skiller seg i stor grad fra hverandre ved at de omhandler og fokuserer på ulike aspekter ved testmiljø. De to første modellene, Caesers konsept i testmiljø og CSTEs kunnskapsnivå om testmiljø, er minst relevant for vår oppgave. Førstenevte fokuserer først og fremst på kostnader forbundet med testmiljø, mens sistenevnte definerer testmiljø altfor bredt slik at mesteparten ikke er relevant for oss. Vi kan likevel trekke ut noen punkter fra 2005-versjonen på et overordnet nivå, nemlig at det er viktig å etablere, analysere og følge opp testprosesser, at et testmiljø skal simulere virkeligheten, viktigheten av å beherske testverktøy samt at et testmiljø enkelt må kunne tilpasses forskjellige teknologier. 2006-versjonen påpeker viktigheten av at et testmiljø skal kunne tilpasses ulike utviklingsmetoder, som vannfallsmodell, objekt-orientert eller smidig utvikling.

Hans Schaefers spesifikasjon av et testmiljø er i høyeste grad aktuell, selv om den fokuserer på et begrenset område og ikke sier noe om viktige tema som for eksempel hvordan et testmiljø bør bygges. Vi vil bruke denne spesifikasjonen

som mal for første sjekklistepunkt i TMM, nemlig at en detaljert spesifikasjon av testmiljøet er dokumentert i testplanen.

TMM er mest relevant for oss, da den inneholder mye informasjon om hvilke aktiviteter og prosesser en bedrift bør gjennomføre i testmiljøet sitt. TMM er en anerkjent og mye brukt metodikk, derfor vil denne danne en del av grunnlaget for våre retningslinjer for testmiljø i kapittel 18.

Det er interessant å legge merke til at bare 2006-versjonen av CSTE eksplisitt definerer testmiljø. Dette er symptomatisk for litteraturen om testmiljø, uten at vi er i stand til å se noen klar årsak. Det er tydelig at kildene legger forskjellig betydning i hva et testmiljø er, og hva det omfatter, derfor synes det rart at svært få definerer begrepet.

En annet likhetstrekk mellom metodikkene er at de er enige om at det er viktig at et testmiljø, spesielt for høynivås tester, er produksjonslikt, men ingen går nærmere inn på hva "produksjonslikt" eller "nært virkeligheten" innebærer. Dette vil vi diskutere nærmere i kapittel 17.

7.6 Sammendrag

Dette kapittelet har gjengitt og diskutert modellene fra teorien som ble funnet på grunnlag av litteratursøket. To av modellene, nemlig Richard Caesars og CSTE ble vurdert som lite passende for vår oppgave, mens Hans Schaefers mal for spesifikasjon av testmiljø og TMM vil bli brukt som teoretisk grunnlag for retningslinjer for testmiljø i kapittel 18.

Kapittel 8
Hva finnes på markedet

En viktig del av oppgaven har vært å undersøke hvilke produkter som finnes på markedet, og som kan adopteres til bruk for testing hos EDB. EDB har allerede noen verktøy og systemer de bruker, men ønsket innspill på hvilke produkter som finnes på markedet, og hvordan disse blir vurdert i tester og rapporter.

8.1 Produkter på markedet

Det finnes mange verktøy og programmer for testing. Utvalget spenner fra store, integrerte løsninger som består av mange verktøy, til små, uavhengige programmer eller verktøy, som har til hensikt å teste en spesifikk ting, for eksempel kode skrevet i Python.

Det som er viktig å kartlegge for oss er verktøy for virtualisering og testmiljø, der nytten av førstenevnte er diskutert i delkapittel 6.5. Vi har derfor sett bort fra små programmer som bare utfører spesialiserte oppgaver, men konsentrert oss om de store, integrerte løsningene og rene virtualiseringsverktøy, hovedsakelig fra kjente og markedsledende leverandører. Hvilke verktøy vi har valgt å se nærmere på er basert på hvem som er ledende i markedet, forslag fra EDB og systemer nevnt i bøker om testing.

Vi har valgt å se nærmere på verktøy fra følgende leverandører:

- VMware
- Microsoft
- IBM
- XenSource
- Borland
- xUnit
- Mercury

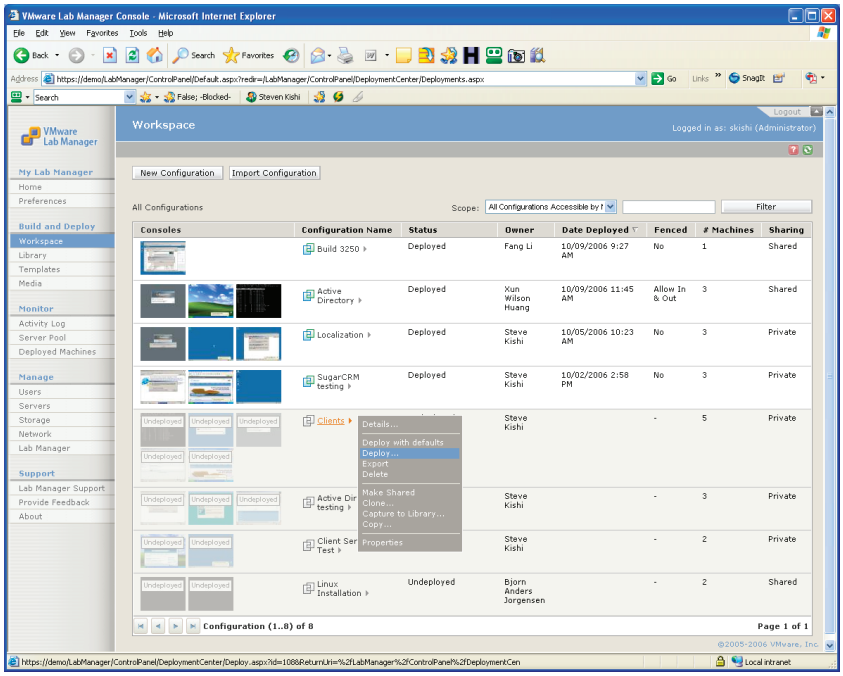
Dette er leverandører av verktøy vi mener er relevante, selv om det ikke er en komplett liste over leverandører av verktøy og programmer på markedet.

8.1.1 VMware

VMware er en stor og velkjent aktør innen virtualisering, og har 4 millioner brukere og 20 000 bedriftskunder (VMware 2007a). VMware leverer to produkter rettet mot utvikling og testing, Lab Manager og Workstation. I tillegg til disse kommersielle produktene, markedsfører VMware to gratis virtualiseringsprodukter, ment å være en døråpner til virtualisering, og med oppgraderingsmuligheter til betal-løsningene senere. Verktøyene heter VMware Player og VMware Server. VMware Player er en nedskalert utgave av virtualiseringsverktøy for skrivebordsmaskiner, med mulighet for å dele virtuelle maskiner med andre, men ikke opprette nye, og VMware Server virtualiserer tjenere. Som sagt er dette enkle utgaver beregnet på å komme i gang med virtualisering, og vi vil derfor fokusere på Lab Manager og Workstation.

VMware Lab Manager

Lab Manager er et relativt nytt produkt, lansert i desember 2006, og bygger på VMware Virtual Infrastucture 3, et system for virtualisering av infrastruktur som tjenere, lagring og nettverk (VMware 2007a). Lab Manager er en infrastruktur for test og utvikling som automatiserer oppsett og nedrigging av programvarekonfigurasjoner som bruker flere fysiske eller virtuelle maskiner. (VMware 2006).



Figur 8.1: Skjermbilde fra Lab Manager(VMware 2006)

Systemet bruker et delt lagringsbibliotek, hvor testere og utviklere kan lagre samlinger av kjørende, uavhengige distribuerte system. Deretter kan brukeren

hente en konfigurasjon i brukergrensesnittet, og Lab Manager tildeler oppsettet de best tilgjengelige ressurser fra ressursene som er ledige. Et eksempel for bruk kan være å bygge et produksjonslikt miljø for testing, lagre ressurser og systemkonfigurasjoner i Lab Manager og på denne måten raskt kunne kjøre et tilsvarende oppsett ved en senere anledning (VMware 2006). Det er også mulig for flere brukere å hente samme konfigurasjonen i biblioteket og kjøre samtidig, og på denne måten kjøre tester i samme miljø parallellt.

Figur 8.1 gjengir et skjermbilde fra VMwares Lab Manager som viser en oversikt over tilgjengelige konfigurasjoner. Oversikten inneholder kolonner for konsoller, om konfigurasjonen er klar til bruk, navn på konfigurasjonen, konfigurasjonens eier, antall maskiner konfigurasjonen inneholder og om den er delt eller privat.

Lab Manager kan integreres med SilkTest, SilkCentral Test Manager, Rational ClearQuest og Mercury Quality Center.

Kunder har 30 dagers gratis prøvetid før man eventuelt bestemmer seg for å kjøpe Lab Manager. I følge VMware starter prisene for Lab Manager på \$15 000[1], men tar man med kostnadene forbundet med VMware Virtual Infrastructure 3, vil systemet koste minst \$35 000.

VMware Workstation

VMware Workstation virtualiserer flere operativsystem på samme datamaskin. Plattformer som støttes er Windows, Linux, NetWare og Solaris x86 (VMware 2007b). Det finnes to versjoner av systemet, en for Windows-klientmaskiner og en for Linux klient-maskiner. Nettverksløsninger muliggjør sammenkobling mellom virtuelle maskiner, vertsmaskinen eller offentlige nettverk. Dette åpner for testing av lagvise konfigurasjoner, nettverksløsninger og flere operativsystemer på samme datamaskin. Brukeren trenger ikke å lage en partisjon for hvert operativsystem eller omstarte maskinen ved bytte av operativsystem, dette gjøres av programmet. Antall virtuelle maskiner er begrenset av diskplassen, og antall virtuelle maskiner som kan kjøre samtidig er begrenset av hvor mye minne vertsmaskinen har installert.

Et skjermbilde fra programmet er vist figur 6.3.

Nåværende versjon er 5.5, men versjon 6.0 forventes lansert i 2007 med full støtte for Windows Vista. VMware tilbyr 30 dagers gratis prøvetid og salgsprisen er \$189 og \$199 for henholdsvis elektronisk nedlastbar og fysisk versjon (VMware 2007b).

8.1.2 Microsoft

Microsoft tilbyr verktøy for virtualisering av enkeltmaskiner og tjenere. Verktøyene er Microsoft Virtual PC og Microsoft Virtual Server.

[1] Alle priser oppgis i amerikanske dollar

Microsoft Virtual PC 2007

Microsoft Virtual PC 2007 er et verktøy for å kjøre flere operativsystemer på samme maskin og skifte mellom operativsystemene på en enkel måte (Microsoft Corporation 2007a). Den kan dermed sammenlignes med WMware Workstation ettersom begge er løsninger for enkeltmaskiner.

Microsoft Virtual PC 2007 muliggjør lagring av individuelle virtuelle maskiner til disk, for å restarte dem på et senere tidspunkt. Programmet kjører bare på operativsystemer fra Microsoft, på samme måte som operativsystemene som er gjester må være produsert av Microsoft. Støttede operativsystem inkluderer Microsoft-løsninger fra Windows 98 til Windows Vista (Microsoft Corporation 2007a).

Nåværende versjon av Virtual PC er 2007, og denne versjonen kan lastes ned gratis fra Microsoft sine hjemmesider (Microsoft Corporation 2007a). Opprinnelig var Virtual PC betal-programvare, men Virtual PC 2004 ble tilgjengelig gratis i 2006, og Virtual PC 2007 følger opp dette.

Microsoft Virtual Server

Mens Microsoft Virtual PC og WMware Workstation er et program som er beregnet på enkeltmaskiner er Microsoft Virtual Server beregnet for større system og tjenere.

Virtual Server kjører på en vertsmaskin med en versjon av Windows Server 2003 eller Windows XP som operativsystem, og kan i tillegg til disse kjøre Windows 2000 eller Windows NT Server 4.0 som gjester. Ved å legge til en tilleggspakke, Virtual Machine Additions, kan man også få flere operativsystem fra Linux.

I tillegg til ren virtualiseringsfunksjonalitet har Virtual Server løsninger innenfor administrasjon av utvikling, skalerbarhet, sikkerhet og ressurshåndtering (Microsoft Corporation 2007c).

Nåværende versjon av Virtual Server er Virtual Server 2005 R2, og denne versjonen kan lastes ned gratis fra Microsoft sine hjemmesider (Microsoft Corporation 2007b).

8.1.3 IBM

IBM markedsfører flere løsninger når det gjelder å forbedre testprosessen. Dette inkluderer små enkeltløsninger, fra valg av operativsystem til programvare for testing av en bestemt type programvare, til WebSphere Test Environment, som er et testmiljø. Vi vil i dette avsnittet først presentere AIX, som er et operativsystem, før vi beskriver produktserien Rational i avsnitt 8.1.3. Vi avslutter presentasjonen av IBM-produkt med WebSphere Test Environment i avsnitt 8.1.3.

IBM AIX 5.3

AIX er et UNIX-basert operativsystem med åpen kildekode. Den inneholder likevel en del funksjonalitet innenfor virtualisering og utvikling og blir derfor nevnt her. AIX har funksjonalitet for sikkerhet, pålitelighet, systemadministrasjon, distribuerte filsystem og virtualisering. Når det gjelder virtualisering støtter AIX 5L virtualiseringsevnen i IBM UNIX-tjenere. AIX krever med andre ord maskinvare fra IBM, og AIX 5.3 er den nåværende utgaven.

IBM Rational Software

IBM har flere produkter og mange av disse produktene er samlet under produktserien Rational. Rational Quality Management er en løsning for å styre utvikling av programmer og programvarearkitektur. Løsningen inkluderer blant annet verktøy for automatisk funksjonell testing av flere programmeringsspråk og plattformer, ytelsestester og generelle testautomatiseringsverktøy for klient/tjener-programmer (IBM Corporation 2007). En del av disse verktøyene er aktuelle for test og er listet opp her.

ClearQuest Verktøy for testledelse og funksjonell programvaretesting.

Functional Tester Verktøy for automatisering av tester av Java, Web og VS .NET WinForm-baserte applikasjoner. Functional Tester kommer også i utgavene Extension og Plus med utvidet funksjonalitet.

Manual Tester Verktøy for manuell testskriving og utførelse.

Performance Tester Verktøy for å verifisere applikasjoners responstid og skalerbarhet under variert last. Performance Tester kommer også i Extension-utgave med utvidet funksjonalitet og z/OS-utgave for ytelsestesting av z/OS.

Purify Verktøy som oppdager minnelekkasje og -ødeleggelse for Linux, UNIX og Windows. Purify kommer også i utgaven PurifyPlus, som er et verktøy for kjøretidsanalyse i tre versjoner.

Robot Verktøy for automatisering av tester for klient/tjener-applikasjoner.

Tester for SOA Quality Verktøy for funksjonell testing og regresjonstesting for tjenester uten grafisk grensesnitt.

Test RealTime Verktøy for å teste komponenter og kjøretidsanalyse for integrerte og sanntidssystemer på tvers av plattform.

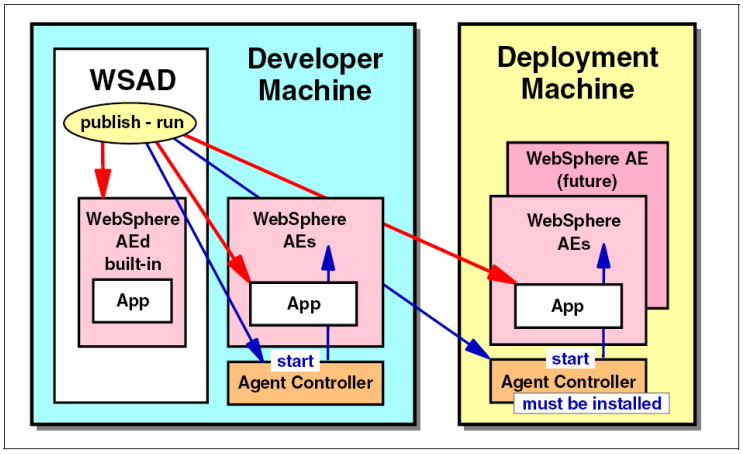
IBM WebSphere Test Environment

Websphere er en produktgruppe fra IBM som består av flere produkter. Websphere har løsninger både for utvikling, som vi ikke vil gå nærmere inn på her, og testmiljø.

Blant programmene i produktserien Websphere, er Websphere Studio Site Developer og Websphere Application Developer de programmene som har

funksjonalitet for enhetstestmiljø innebygget i programmet. Websphere Studio Site Developer inneholder et testmiljø for nettbaserte prosjekt (Tretau 2002). Hjelpemidlene på tjenersiden omfatter en lokal kopi av Websphere Application Server kjøretidsmiljø og et enhetstestmiljø for Tomcat. Websphere Application Server har det samme testmiljøet som Studio Site Developer, men inneholder i tillegg støtte for utvikling av EJB og forretningsprogram.

I tillegg til de enkelte testmiljøene som fins i Websphere Studio Site Developer og Application Developer, finnes det eksempel på hvordan man bygger større testmiljø med bruk av Websphere Application Developer. Websphere anbefales for eksempel til testing av applikasjoner og EJB. Figur 8.2 illustrerer hvordan Websphere Application Developer lager et lokalt og et fjerntliggende testmiljø for testing av nettsystemer,.



Figur 8.2: Testing av applikasjoner og EJB(Tretau 2002)

8.1.4 XenSource

XenSource tar sikte på å være en utfordrer til VMware i virtualiseringsmarkedet, og er basert på åpen kildekode fra Xen-prosjektet. Xen-prosjektet startet som et forskningsprosjekt ved universitetet i Cambridge. Lederen av forskningsprosjektet, Ian Pratt, grunnla firmaet XenSource Inc., som selger kommersiell programvare basert på Xen (Wikipedia - the Free Encyclopedia 2007b).

XenSource markedsfører tre versjoner av programvaren, XenEnterprise, XenServer og XenExpress. XenSource XenEnterprise ble lansert i versjon 3.2 i slutten av februar 2007, og kjører på Windows og Linux vertsmaskiner (XenSource 2007). Programmet tilbyr muligheten for å lage, kjøre og styre virtuelle maskiner, der støttede operativssystemer inkluderer Windows XP, Windows Server 2003 og flere Linux-distribusjoner, hovedsakelig tjenerversjoner.

XenServer tilbyr de samme egenskapene som XenSource, men kjører bare i Windows-miljø og er begrenset til åtte virtuelle tjenere. Figur 8.3 viser et

Figur 8.3: Skjermbilde fra XenSource XenServer (XenSource 2007)

skjermbilde fra styringsverktøyet til programmet, som viser en oversikt over prosessorbruk, ledig minne, disk og nettverk.

XenExpress er en gratis, nedskalert utgave av XenSource, og støtter Windows og Linux. Systemet har visse begrensninger i forhold til betalversjonene, blant annet er det en grense på 4GB med minne og maksimalt fire samtidige virtuelle tjenere. XenExpress kan anses som en inngansport til Xen, og kan enkelt oppgrades til XenEnterprise eller XenServer.

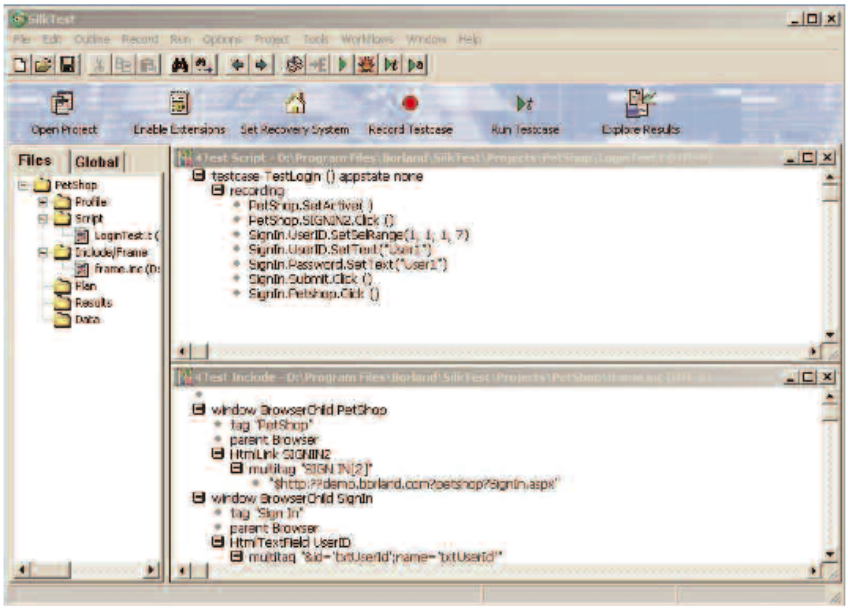
Prisen på XenEnterprise starter på $488 for årlig lisens og $750 for fast lisens. I tillegg tilbys 30-dagers gratis prøvetid. XenServer koster $99 for årlig lisens mens XenExpress er gratis.

8.1.5 Borland

Borlands verktøy for testing består av fire produkter: SilkCentral Test Manager, Gauntlet, SilkTest og SilkPerformer. Silk-serien markedsføres som et produkt fra Borland etter oppkjøpet av den opprinnelige produsenten Segue i 2006 (Borland 2007)

SilkTest

SilkTest er et verktøy for automatisk funksjonell testing, og kan brukes til å teste alt fra grafiske grensesnitt via nettlesere til databaser. Figur 8.4 viser et skjermbilde fra verktøyet SilkTest i bruk. Verktøyet har et eget objektorienterte skriptspråk kalt 4Test (Borland 2007). Språket støtter flere plattformer og programmer, blant annet .NET.



Figur 8.4: Skjermbilde fra SilkTest (Borland 2007)

SilkTest består av fire deler:

- Integrated Development Environment (IDE)
- Agent
- Extension Kit
- Silk TrueLog Explorer

IDE er SilkTests grafiske brukergrensesnitt, som muliggjør utvikling, endring, kjøring, kompilering, analysering og feilsøking, ved hjelp av 4Test. Extension Kit lar brukeren skrive nye 4Test Agent-funksjoner i C eller C++. For å analysere og feilsøke skriptlogger fra individuelle tester for å finne feilsituasjoner brukes Silk TrueLog Explorer (Borland 2007).

Verktøyet har et innebygget Recovery System som muliggjør tilbakestilling til en brukerdefinert starttilstand, og dermed kan nye automatiske tester kjøre med suksess selv om en skulle feile.

Verktøyet kjører på Windows, RedHat Linux og Sun Solaris som operativsystem, og støtter en lang rekke miljø, blant annet har det støtte for Java og .NET. SilkTest kan prøves gratis i 30 dager.

Gauntlet

Gauntlet er et verktøy for integrasjonstesting, myntet på smidig utviklingsmiljøer. Verktøyet fungerer sammen med versjonskontrollsystemer, slik at det bygger og tester programvaren hver gang utviklere sjekker inn kode. Gauntlet tilbyr også muligheten til å isolere kode med feil, slik at feil ikke hindrer andre utviklere (Borland 2007).

Verktøyet støttes av Windows XP, Windows 2003 Server og RedHat Linux, og kan prøves gratis i 30 dager.

SilkCentral Test Manager

SilkCentral Test Manager er et verktøy for teststyring som blant annet støtter nettbasert kvalitetsrapportering, styring av testkrav, planlegging av tester ved hjelp av en testplan og utføring av tester i testplanen gjennom en nettportal (Borland 2007).

I tillegg til å kunne integreres med de andre produktene i Borlands testpakke, muliggjør arkitekturen integrering med andre leverandørers verktøy, blant annet VMwares Lab Manager, JUnit og NUnit.

SilkCentral kjører kun på operativsystem fra Windows, og støtter nettleserne Explorer og Mozilla Firefox. Borland tilbyr 30-dagers gratis prøving av verktøyet.

SilkPerformer

SilkPerformer brukes til ytelses- og lasttesting. Verktøyet tilbyr muligheten for å simulere tusenvis av samtidige brukere, med ulike datamaskinmiljø, på en maskin. SilkPerformer støtter mange miljøer, inkludert Java- og .NET-rammeverkene. Grensesnittet støtter å lage tester ved hjelp av et pek-og-klikk-grensesnitt, import av tester laget i JUnit eller NUnit eller å bygge nye tester i det aktuelle programmeringsspråket ved hjelp av SilkTest Java Editor eller en Visual Studio-tilleggspakke.

SilkPerformer kan integreres med de andre produktene i pakken, for eksempel SilkCentral Test Manager for å generere mer avanserte rapporter fra testene enn det verktøyet selv tilbyr.

Borland tilbyr 30-dagers gratis prøvetid på verktøyet. Av operativsystem støtter SilkPerformer Windows 2000, Windows 2003 og Windows XP (Borland 2007).

8.1.6 xUnit

xUnit er en serie av rammeverk basert på åpen kildekode beregnet på å implementere enhetstester for flere programmeringsspråk. Det første miljøet som ble populært var et testmiljø for Java-klasser kalt JUnit (Doar 2005). JUnit var en viktig pådriver til den økende interessen rundt eXtreme Programming (XP) og testdrevet utvikling (Fowler 2007), fordi JUnit integrerer godt med disse utviklingsmetodikkene. Et eksempel på en JUnit-test fra (Wikipedia - The Free Encyclopedia 2007a) er:

```
public class HelloWorld
{
```

```
    @Test public void testMultiplication()
    {
      // Testing if 3*2=6:
      assertEquals ("Multiplication", 6, 3*2);
    }
}
```

Som vi ser, er syntaksen lik Java, og dette gjør at JUnit er lett å komme i gang med for utviklerne.

Etterhvert som JUnit ble populært, ble rammeverket portet til flere språk. I dag finnes xUnit for de fleste språk. Noen eksempler er:

NUnit for C#

PyUnit for Python

fUnit for Fortran

CPPUnit for C++

JSUnit for JavaScript (Wikipedia - The Free Encyclopedia 2007a)

Alle disse testmiljøene bruker samme grunnleggende arkitektur som JUnit, selv om de er spesialisert for ulike språk. Dette gjør at det er enklere å skrive tester i forskjellige språk og å forstå andres tester (Doar 2005).

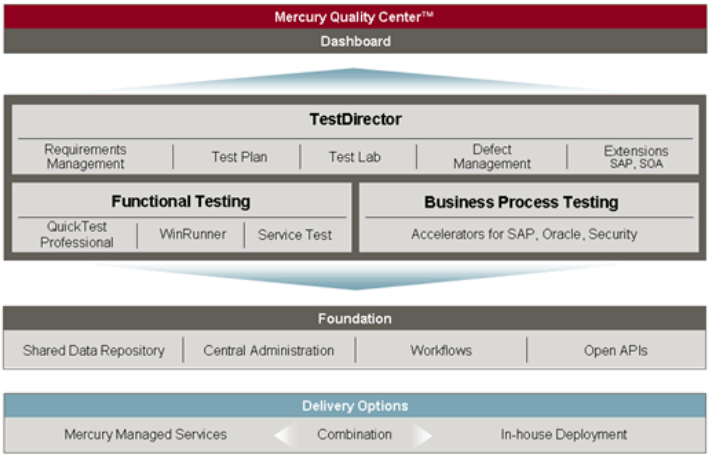
Som nevnt er alle rammeverk i xUnit gratis, og de kjører på tvers av plattformer.

8.1.7 Mercury Quality Center

Mercury Quality Center er en programvarepakke for å kjøre funksjonelle tester på Windows, markedsført av selskapet Hewlett-Packard. I tillegg til Quality Center, tilbyr HP løsninger for virtualisering av infrastruktur som tjenere, nettverk og lagringsplass, men disse løsningene omtales ikke nærmere her.

Quality Center tilbyr en nettbasert løsning for automatisert programtesting og styring, og består av flere deler, som figur 8.5 viser.

TestDirector øverst på figuren søker å støtte hele testprosessen, deriblant styring av kravspesifikasjoner; planlegging, bygging, tidsstyring og utførelse av tester, feilhåndtering og prosjektstatusanalyser. Når TestDirector brukes sammen med QuickTest Professional og/eller WinRunner, kjører de to sistenevnte automatiske tester, mens TestDirector lagrer resultatene og testskriptene (Testapps.com 2007). Deretter er pakken delt opp i to systemer, ett for funksjonell testing og ett for testing av forretningsprosesser. Delen for funksjonell testing er satt sammen av produktene Mercury QuickTest Professional og Mercury WinRunner, i tillegg til Service Test.



Figur 8.5: Oversikt over Mercury Quality Center (Hewlett-Packard 2007)

Service Test er en løsning for ytelse- og funksjonstester av Service-Oriented Architecture (SOA)-programmer. QuickTest Professional brukes for automatiserte funksjons - og regresjonstester. QuickTest støtter flere utviklingsteknologier, som for eksempel Web Services, .NET, Java og J2EE. Mercury hevder at QuickTest er enklere å bruke enn WinRunner, fordi mange av tingene som krever koding i WinRunner, for eksempel å lage if-setninger, kan gjøres i QuickTest ved hjelp av innebygget funksjonalitet (Testapps.com 2007).

WinRunner brukes også for å lage og kjøre automatiske testskript og støtter et stort antall teknologier, programmeringsmiljø og tjenere, deriblant Java og Oracle. For å utvikle testskript over et basisnivå, brukes Mercurys Test Script Language (TSL), et språk som har arvet mye fra C. Dette gjør opplæring enklere for personer med programmeringserfaring (Testapps.com 2007).

I Mercury Quality Center integreres WinRunner og QuickTest Professional, og åpner for at systemene kan bruke testskript fra hverandre, og resultatet vises i et felles vindu.

Mercury har annonsert at de vil prioritere QuickTest Professional og anbefale dette systemet for nye kunder som bruker moderne utviklingsplattformer. Grunnet stor markedsandel kan Mercury likevel ikke slutte å støtte WinRunner for eksisterende kunder (Testapps.com 2007).

Den siste den av Quality Center er Mercury Business Process Testing. Dette er et system for å designe funksjonelle case for forretningsprosesser for både automatiske og manuelle tester.

Når en bedrift ønsker å innføre Mercury Quality Center, er det mulig å innføre så få eller mange deler som bedriften selv ønsker, og eventuelt bygge på senere med nye deler. Som figur 8.5 viser, er det mulig å implementere Mercury Quality Service ved hjelp av Mercury Managed Services, der Mercury eller partnere drifter systemet, og blant annet tilbyr opplæring og telefonstøtte. Deretter kan systemet overføres helt til bedriften.

Mercury markedsfører også LoadRunner, et verktøy for ytelsestesting som kan

etterligne et høyt antall samtidige brukere for å teste ytelsen til et system. Aktuelle bruksområder innenfor ytelsestesting kan være stresstesting og å identifisere flaskehalser i systemet (Hewlett-Packard 2007).

Flere av delene kan testes gratis, blant annet tilbyr TestDirector 30-dagers prøvetid og QuickTest Professional 14-dagers prøvetid. Prisene varierer med hvilke og hvor mange deler man ønsker, implementeringsløsning og antall lisenser. Mercury anslo en kostnad på $50 000 med 15 brukere ved kjøp av Quality Center med Business Process Testing i 2005 (ITworld.com 2005).

8.1.8 Andre verktøy

I tillegg til verktøyene som er beskrevet i ovenfor, undersøkte vi noen få andre verktøy som av ulike årsaker ikke var så relevante som vi opprinnelig hadde håpet på. Disse oppsummeres kort her.

DejaGnu Testmiljø basert på åpen kildekode under GNU-lisens. Programmet bruker Tool command language (Tcl) som skriptspråk, og er skrevet i Expect. Hovedfokuset til DejaGnu er systemtester.

Virtual Iron Software Inc. Virtualiseringsverktøy som konkurrerer med XenSource om å være et alternativ til VMware, og markedsfører seg som virtualisering til en femtedel av VMwares kostnader. Kjører på Linux- og Windows-tjenere.

8.1.9 Oppsummering av verktøy

Tabell 8.1 viser en oppsummering av testverktøy, med prisanslag og vertsplattform. I tillegg til oppgitt lisenspris, kommer utgifter til opplæring av brukerne.

8.2 Tester og rapporter

For å sammenligne produktene på markedet, er det interessant å undersøke publiserte anmeldelser av verktøy og tilgjengelige rapporter. Spesielt interessante er tester som sammenligner flere produkter.

Det viste seg at det finnes mange tester som behandler WMwares Workstation 5.0 og 5.5, Virtual PC 2007 og til dels XenSources XenEnterprise og XenServer. Vi tror dette skyldes at virtualiseringsløsninger er mer kommersielt tilgjengelig og interessante for en større kundegruppe, og derfor mer aktuelle å teste. Pakkene av testverktøy de store leverandørene som IBM, Borland og HP tilbyr er dyre, delvis beregnet på et profesjonelt marked og mer komplekse.

Verktøy	Lisens	Vertsplattform
VMware Lab Manager	30 dagers prøvetid Fra $15 000 Fra $35 000 med Virtual Infrastructure	Windows Linux
VMware Workstation	30 dagers prøvetid $189 elektronisk versjon $199 fysisk versjon	Windows Linux
Microsoft Virtual PC 2007	Gratis	Windows
Virtual Virtual Server	Gratis	Windows
IBM AIX 5.3	Lisensiert, med en lisens per prosessor. Leveres ofte med maskinvare fra IBM	Unix
XenSource XenEnterprise	30-dagers gratis prøvetid Fra $488 årlig Fra $750 fast	Windows Linux
XenSource XenServer	$99	Windows
XenSource Express	Gratis	Windows Linux
IBM Rational Software	Lisensiert av IBM, pris per verktøy	Linus, UNIX og/eller Windows avhengig av verktøy
IBM Websphere		
SilkTest	30-dagers gratis prøvetid	Windows
Gauntlet	30-dagers gratis prøvetid	Windows RedHat Linux
SilkTest	30-dagers gratis prøvetid	Windows RedHat Linux Sun Solaris
SilkCentral Test Manager	30-dagers gratis prøvetid	Windows
xUnit	Gratis	Alle
Mercury Quality Center	Noen deler gratis prøvetid Pris varierer med oppsett	Windows

Tabell 8.1: Oppsummering av testverktøy

8.2.1 Virtualiseringsløsninger

Nettavisen ITWorld refererer til en undersøkelse publisert av Forrester Research i begynnelsen av februar 2007 (Robert Mullins for IDG News Service 2007). Denne viser at 58% av USAs IT-sjefer mest sannsynlig ville vurdert VMware for å implementere virtualisering. 11% ville foretrukket Hewlett-Packard, og øvrige produkter oppnådde ikke to-sifret prosenttall.

Databladet PC Magazine testet Microsoft Virtual PC 2007 1. mars 2007 (PC Magazine 2007). Kort oppsummert konkluderer bladet med at produktet ikke har så mange funksjoner som VMware Workstation, men den store fordelen er at det er gratis. De påpeker også at verktøyet ikke kjører under Home-versjoner av XP eller Vista, ikke tilbyr støtte for USB-enheter eller flere snapshot, men bare en lagre/gjenoppta-funksjon, og gir verktøyet tre av fem mulige prikker. Til sammenligning fikk VMware Workstation 5.0 fire og en halv prikk av fem mulige i en tilsvarende test utført av samme blad 13. april 2005, og utmerkelsen redaktørens valg (PC Magazine 2005).

Det britiske databladet PC Pro testet VMware Workstation 5.5 6. april 2006, gav verktøyet fem av fem stjerner og utmerkelsen "PC Pro Recommended" (PC PRO 2006). Magasinet mener at Workstation fortsatt er det beste virtuell maskin-verktøyet på markedet, og trekker spesielt fram støtten for både Linux og Windows som vertsmaskiner og at verktøyet har det beste utvalget på markedet i støttede gjestmaskiner.

Nettavisen Infoworld har også testet VMware Workstation 5.0 22. august 2005 (Infoworld 2005). Avisen gir verktøyet en poengsum på 8.8 av 10 mulige poeng og mener at Workstation slår konkurrentene Microsoft Virtual PC og XenSources Xen.

En interessant sammenligning av Microsoft Virtual PC 2007 og VMware Workstation 6.0 Beta 3, i tillegg til to andre verktøy, foretatt av Infoworld 22. mars 2007 konkluderer også med at Workstation er best (Infoworld 2007). Virtual PC ender på 7.4 poeng av 10, mens Workstation oppnår 8.3 poeng. Testen plasserer Virtual PC 2007 i samme klasse som gratisverktøyet VMware Player grunnet funksjonalitet, og mener at VMware Workstation er det eneste valget for seriøse virtualiseringsbrukere.

XenSource XenServer ble testet i Personal Computer World 2. mars 2007 (Personal Computer World by Alan Stevens 2007). I denne testen kommer verktøyet dårlig ut, og oppnår to av fem mulige stjerner. Anmelderen Alan Stevens konkluderer med at XenServer ikke er en truende konkurrent til VMware eller Microsoft i nåværende form. Også XenEnterprise 3.0 er blitt testet 12. september 2006, og anmelderen mener produktet virker bra innenfor sine grenser (Personal Computer World by Alan Stevens 2007). Bladet peker på at XenSource er en ung teknologi i forhold til VMware, og at XenEnterprise trenger bredere støtte for gjeste-operativsystemer for å utfordre løsninger som for eksempel VMwares.

8.2.2 Mercury og SilkTest

I følge (Testapps.com 2007) er Mercury WinRunner en av markedslederne innenfor automatiske funksjonelle testverktøy, men kilden definerer ikke nærmere hva dette innebærer. Også Doar mener Mercury WinRunner er et standard testmiljø i industrien, sammen med SilkTest (Doar 2005). Som nevnt vil Mercury prioritere QuickTest Professional i framtiden, og en anbefaling om å ta i bruk WinRunner vil derfor virke lite framtidsrettet.

8.3 Sammendrag

Dette kapittelet har presentert verktøy for virtualiseringsløsninger og for bruk under arbeid med testmiljø. VMwares virtualiseringsløsninger ble vist å være best på grunnlag av sammenligninger, fulgt av Microsofts løsninger. Vi vil komme tilbake til virtualisering i kapittel 14 og i kapittel 17 hvor vi vil se på de involverte bedriftenes bruk av virtualisering.

Kapittel 9
Risikoanalyse

En del av oppgaven vil være å vurdere hvilke risikoer som finnes ved bygging og bruk av testmiljø. Vi vil derfor her gi en kort introduksjon på hva en risiko er, hvordan risikoer kan analyseres og hvorfor det kan være nyttig å gjøre risikoanalyse av testmiljø.

9.1 Risiko

En risiko er en hendelse som kan bli et problem i framtiden. Den har to parametre, konsekvens og sannsynlighet (Stålhane & Skramstad 2006, Graham et al. 2007). Konsekvens angår det som vil skje dersom hendelsen inntreffer. Sannsynligheten er sjansen for at hendelsen vil bli et problem.

Risiko defineres dermed som $R = C * p$ der
R = Risiko
C = Konsekvens
p = Sannsynlighet (Stålhane & Skramstad 2006)

Noen vil velge å erstatte sannsynlighet med frekvens f, og man får $R = C * f$ (Spillner, Linz & Schaefer 2006).

En risiko kjennetegnes ved tre egenskaper.

1. Risikoen har usikkerhet, ved at det er en hendelse som kan inntreffe.
2. Det er mulig å identifisere hendelsene, men konsekvensene og sannsynligheten for hendelsen er usikre.
3. Resultatet av hendelsene kan bli påvirket av våre handlinger (Stålhane & Skramstad 2006).

9.2 Gjennomføring av risikoanalyse

Problemene som oppstår ved at risikoer inntreffer er ikke ønskelige for en bedrift. Ved en risikoanalyse, vil man prøve å finne hvilke risikoer man har, hvilke konsekvenser de gir og hvor stor sannsynlighet det er for at hendelsene inntreffer. Målet er å finne tiltak og å vurdere kostnaden på tiltakene opp mot alvorlighetsgraden på risikoen.

9.2.1 Innhenting av data

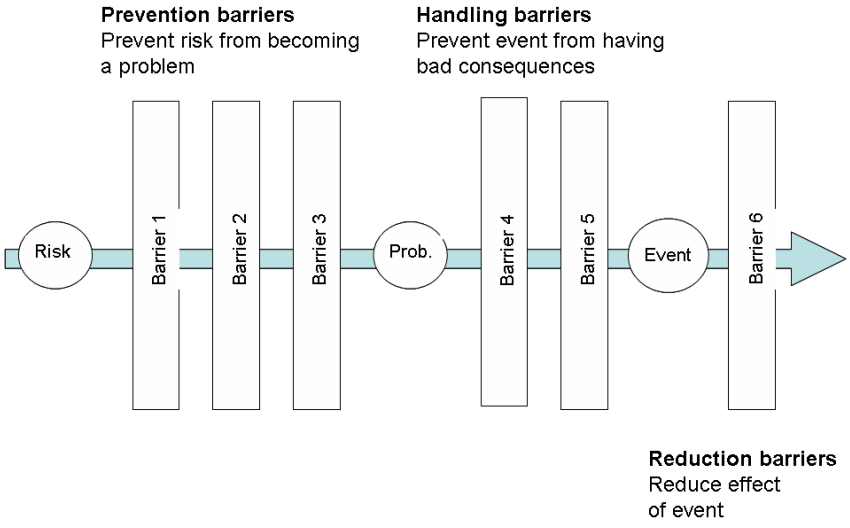
Det finnes flere måter å innhente data til en risikoanalyse. Graham et al. nevner tre hovedmetoder for innsamling av data.

1. Man henter informasjon om risiko gjennom å analysere eksisterende dokumentasjon, som kravspesifikasjon, brukerdokumentasjon og lignende.
2. Idémyldring med interessentene i prosjektet.
3. Enkeltsamtaler eller gruppesamtaler blant forrestnings- og teknologiekspertene i selskapet (Graham et al. 2007).

Stålhane og Skramstad foreslår også bruk av idémyldring for å innhente data. Her er idémyldring derimot dekkende for både idémyldring og samtalene hos Graham et al. Begge legger vekt på at man gjennom samtale vil komme fram til flere hendelser og tiltak ved å snakke med de som arbeider med emnet. Mye av grunnen til dette ligger i at risikoanalysen bygger på kunnskap og erfaring (Stålhane & Skramstad 2006, Graham et al. 2007).

9.2.2 Tiltak

Tiltak i forbindelse med risiko kan deles i tre kategorier: Tiltak for å unngå at risikoen skal bli et problem, tiltak for å unngå at problemet gir negative konsekvenser og tiltak for å redusere effekten av hendelsen (Stålhane & Skramstad 2006). Hvordan disse tiltakene henger sammen i et risikoforløp vises i figur 9.1.



Figur 9.1: Tiltak mot risiko (Stålhane & Skramstad 2006)

9.3 Behovet for risikoanalyse

Graham et al. deler risikoene i to kategorier, produktrisikoer og prosjektrisikoer. En produktrisiko vil være relatert til kvaliteten på produktet man skal levere, og kan for eksempel være at at systemet ikke oppfyller kundens forventninger. En prosjektrisiko vil derimot være relatert til håndtering og kontroll av testprosjektet, og et eksempel kan være at testdata er forsinket fra leverandør. Begge disse risikotypene vil man kunne få oversikt over gjennom en risikoanalyse (Graham et al. 2007).

Som nevnt i kapittel 5, er det umulig å teste alle mulige hendelser i en testprosess. Man må prioritere hva som er viktig og dette kan gjøres på bakgrunn av en risikoanalyse (Spillner et al. 2006). Resultatet av en risikoanalyse er en oversikt over hvilke risikoer man står ovenfor og hvilke tiltak man har mulighet til å gjennomføre. Risikoene prioriteres og man får en oversikt over hvilke problemer man kan forhindre. På bakgrunn av risikoanalysen kan man utarbeide rutiner for hvordan man skal håndtere problem som oppstår, ved at man vet hvilke problemer dette er og hvilke tiltak man har.

9.4 Risikoanalyse av testmiljø

Et testmiljø består av blant annet programvare, maskinvare og testdata. Man er avhengig av at alle delene fungerer hver for seg og at de samarbeider som ønskelig. Det medfører derfor en risiko å bygge og drifte et testmiljø. Dersom en del svikter vil dette ofte få innvirkning på hele testmiljøet. Ved å gjennomføre risikoanalyse av testmiljø, vil man kunne få en oversikt over hvilke problem som kan oppstå, og hvilke tiltak som eventuelt kan settes inn.

9.5 Sammendrag

Dette kapittelet har gitt en kort innføring til risikoanalyse og hvordan den kan gjennomføres. Dette er teori som vi tar med oss videre til gjennomføringen av risikoanalyse i bedrifter. Kapittel 11 skilderer hvordan risikoanalysen er utført på bakgrunn av hele delkapittel 9.2, mens vi i kapittel 17 diskuterer hvordan risikoene er fordelt i forhold til tiltaksoppdelingen delkapittelet gir.

Del II

Case

Kapittel 10
Introduksjon til Case

I denne delen presenterer vi det praktiske arbeid vi har gjennomført. Dette inkluderer en presentasjon av involverte bedrifter, og metode for gjennomføring og resultater fra samtalene.

10.1 Delens oppbygging

Metode Metodekapittelet presenterer hvilken metode som er brukt for å innhente informasjon fra ulike bedrifter, og skildrer hvordan intervju og risikoanalyse er blitt gjennomført.

EDB Dette kapittelet presenterer EDB, EDBs testmetodikk og framgangsmåte for testing. I tillegg beskriver vi dagens situasjon, hvordan testmetodikken omhandler testmiljø og hvorfor det er ønskelig å kartlegge hvor EDB står i dag.

Intervjuede bedrifter I tillegg til EDB, gjennomførte vi også intervju og risikoanalyse i flere andre bedrifter. Disse bedriftene er presentert i dette kapittelet.

Oppsummering av intervjuer Intervjuene ligger utskrevet i vedlegg B, mens dette kapittelet sammenfatter den informasjonen fra intervjuene som vi vektlegger i kapittel 17.

Oppsummering av risikoanalyser Risikotabeller fra risikoanalyse i de enkelte bedriftene ligger i vedlegg C, mens de viktigste resultatene er utskrevet i dette kapittelet. Risikoene er sortert i de tre kategoriene organisatoriske risikoer, samarbeidsrisikoer og tekniske risikoer.

Kapittel 11
Metode

Dette kapittelet beskriver metoden vi har benyttet ved gjennomføring av intervjuer og risikoanalyse hos EDB og andre firma.

11.1 Forberedelse

Før vi avtalte møter med bedrifter forberedte vi et intervju og en risikoanalyse.

11.1.1 Spørsmål

Vi forberedte 11 spørsmål. Disse er gjengitt i vedlegg A, og ble utdypet med oppfølgingsspørsmål i løpet av intervjuet. Spørsmålene fokuserer på testmiljø, og inneholder både spørsmål som går på rutinen i bedriften, og mer åpne spørsmål om emner som hvor nært et akseptansemiljø kan være et produksjonsmiljø og hvorfor det ikke finnes metodikker for testmiljø i litteraturen.

Det er nærliggende å tro at det vil variere hvor mye informasjon vi får fra intervjuene, ettersom noen personer vil være enklere å snakke med, og selv vil ta initiativ til å fortelle utover det vi spør om. Dette vil føre til forskjeller i kvaliteten på informasjon, og de forberedte spørsmålene medvirker i så måte til at vi får et minimum av informasjon som kan sammenlignes mellom bedriftene.

11.1.2 Risikoanalyse

For å lettere kunne forklare risikoanalysen for intervjuobjektene, laget vi en veiledning som vist i figur 11.1.

11.1.3 Veiledning til risikoanalyse

Veiledningen ble laget på bakgrunn av stoff fra en forelesning. (Stålhane & Skramstad 2006). Som diskutert i 9.1 kan det være aktuelt å erstatte sannsynlighet med frekvens i en risikoanalyse. Vi valgte å gjøre dette fordi sannsynlighet er et abstrakt begrep som kan være vanskelig å relatere seg til. Det er enklere å si hvor mange ganger en hendelse har inntruffet basert

Beskrivelse av risikoanalyse-skjema

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig

Skjemaet fylles ut med tanke på hvilke hendelser som kan skape problem i forbindelse med testmiljø.

Hendelse:	Hendelse som kan skape problem.
	F.eks. testmiljøet går ned rett før en viktig test skal kjøres.
K:	Konsekvens
	Settes som H (høy), M (middels) eller L (lav)
f:	Frekvens
	Settes som H(høy), M(middels) eller L(lav)
R:	Risiko= K*f
Tiltak:	Hva kan man gjøre for å redusere risikoen?
Kostnad:	Hva vil tiltaket koste?
	Settes som H (høy), M (middels) eller L (lav)
E:	Hvilken effekt har tiltaket?
	Settes som H (høy), M (middels) eller L (lav)
L:	Leverage = (K*f*E – Kostnad) / Kostnad
Ansvarlig:	Hvem er ansvarlig for at dette blir fulgt opp.

Figur 11.1: Veiledning til skjema for risikoanalyse

på tidligere erfaringer, enn å estimere sannsynligheten for at dette vil skje i framtiden.

Vi valgte å bruke idémyldring med ulike interessenter i prosjektet (Graham et al. 2007). Gjennom samtale vil vi komme fram til aktuelle hendelser og tiltak ved å snakke med dem som arbeider med emnet (Stålhane & Skramstad 2006, Graham et al. 2007). Testmiljø har i tillegg vist seg å være et emne som ikke er så godt dokumentert i bedriftene, og derfor bygger mye på kunnskap og erfaring blant testerne i de involverte firmaene.

11.1.4 Utvelgelse av intervjuobjekter

Personer fra EDBs testsentre i Trondheim og Oslo var aktuelle intervjuobjekter, og her ble møtetidspunkt avtalt via e-post.

Vi ønsket også å komme i kontakt med personer fra andre bedrifter som hadde kompetanse på testmiljø. Vi tok kontakt med Hans Schaefer som gav oss en del navn, deriblant Hans-Erik Ballangrud hos IBM, som har jobbet med Telenors 2000-årsprosjekt. Fra Ballangrud fikk vi flere navn på personer i aktuelle bedrifter. Vi avtalte intervjutidspunkt via e-post med personene som var villige til å stille opp. Bedriftene vi avtalte intervju med var Helse Midt-Norge IT (HEMIT), Abeo og Domstoladministrasjonen, i tillegg til Ballangrud selv. Møtene med EDBs testavdeling i Oslo og Ballangrud hos IBM fant sted i Oslo, mens de øvrige ble gjennomført i Trondheim.

Utvelgelsen av bedrifter var tilfeldig. Alle bedriftene var aktuelle, siden vi eksplisitt spurte etter bedrifter med kompetanse på testmiljø. Det er likevel mulig at andre bedrifter kunne vært mer aktuelle eller at vi kunne intervjuet flere bedrifter. Angående sistenevnte var tiden vi hadde til rådighet en begrensning. Underveis i samtalene fikk vi flere andre navn på aktuelle bedrifter, men grunnet tidsrammer rakk vi ikke å gjennomføre flere enn seks samtaler.

Selv om bedriftene vi gjennomførte samtaler med ikke er et representativt utvalg av bedrifter i Norge, oppnådde vi en spredning i bransjeområder og av offentlige og private bedrifter. Ettersom Telenors 2000-årsprosjekt kun er ett prosjekt innenfor Telenor, og i tillegg var høyt prioritert med tilgang på mye ressurser, er ikke nødvendigvis resultatene direkte sammenlignbare med andre bedrifter. Det er mulig at vi kunne fått andre svar dersom vi hadde diskutert ett konkret prosjekt i samtalene med de øvrige bedriftene. Vi legger likevel vekt på resultatene, siden Ballangrud har lang erfaring innenfor testmiljø, og erfaringene fra dette prosjektet er viktige nettopp fordi prosjektet hadde mye ressurser.

11.2 Gjennomføring

En risikofaktor kan være rekkefølgen vi gjennomfører intervjuene på. Vi fikk etterhvert mer trening og så hvilke oppfølgingsspørsmål som bør stilles. Det er derfor en risiko for at vi vil få mindre ut av det første intervjuet enn de siste. Grunnet tidspress både for oss og intervjuobjektene rakk vi ikke å ta det første intervjuet om igjen selv om dette sannsynligvis hadde eliminert risikoen. Istedenfor stilte vi oppfølgingsspørsmål på e-post. Det første intervjuet ble gjennomført hos EDB i Oslo, og de stilte som eneste bedrift med tre personer på samtalen. Dette åpnet for diskusjon innad blant intervjuobjektene, og at de utfylte hverandre. Vi hadde også kjennskap til EDB gjennom at bedriften er samarbeidspartner for oppgaven. Tilsammen mener vi at tiltakene har resultert i at vi får like mye ut av de første intervjuet som de andre.

11.2.1 Utskriving av notater fra intervju

Vi valgte å ikke bruke båndopptaker under intervjuene. Istedet noterte vi i stikkordsform og skrev ut i etterkant av intervjuet. Vi vurderte de potensielle fordelene ved båndopptaker opp mot ulempene, og kom fram til at vi ønsket å gjennomføre intervjuene uten dette hjelpemiddelet. Potensielle ulemper er usikkerhet om hva intervjuobjektene sa, misforståelser og at vi ikke får med oss all informasjon som blir gitt. Ulemper med båndopptaker kan være at intervjuobjektet blir usikker og hadde snakket friere uten båndopptaker, samt mye arbeid med å transkribere opptakene til tekst i etterkant av intervjuet i forhold til gevinst. Vi vurderte det slik at fordelene ikke oppveiet ulempene, og valgte heller å ta kontakt med intervjuobjektene i etterkant for å rydde opp i eventuelle uklarheter under utskriving fra notater. Dette medfører også at intervjuene i vedlegg B ikke er ordrett gjengitt, men skrevet med våre ord på grunnlag av notater. For å forsikre oss om at intervjuobjektene kan stå inne for utskriften vår, er denne sendt til godkjenning via e-post. På denne måten mener vi at vi får pålitelig informasjon fra intervjuene, og at denne er gjengitt på en god måte i oppgaven vår.

Vi tror at spørsmålene vi hadde forberedt og stilte under intervjuene fungerte bra. I utgangspunktet er det en risiko for at vi ikke stiller de rette spørsmålene, og i etterkant, og da spesielt under diskusjonen, ser at vi ikke har nok informasjon til å svare på forskningsspørsmålene. Dette problemet opplevde vi i liten grad, og tror at det faktum at spørsmålene var åpne, og at vi stilte oppfølgingsspørsmål underveis har bidratt til dette. Det eneste vi ser i etterkant er at vi burde spurt intervjuobjektene om hvilket verktøy for test de benyttet seg av. Virtualiseringsløsninger ble diskutert, men ikke andre verktøy relatert til test. Hadde vi spurt om dette, kunne vi hatt et bedre grunnlag for å komme med en anbefaling på testverktøy til EDB.

11.2.2 Risikoanalyse

Veiledningen i figur 11.1 ble delt ut til deltakerne i begynnelsen av møtet, og vi gjennomgikk den i felleskap. Skjemaet ble så fylt ut på grunnlag av prinsippene i delkapittel 9.2. Bruk av skalaen med høy-middels-lav er ingen eksakt vitenskap, men basert på skjønn og individuelle vurderinger fra hvert intervjuobjekt. Dette kan føre til at det kan være vanskelig å sammenligne leverage-verdiene mellom bedriftene.

Under møtene ble skjemaet prosjektert på lerret og oppdatert slik at alle deltakerne kunne se på lerretet hva som ble skrevet. Dette fungerte bra, da vi fikk umiddelbar respons på om vi hadde misforstått noe, og alle deltakerne fikk med seg hva som ble sagt. Vi tror denne framgangsmåten gjorde at alle fulgte med på det som ble sagt, og at den åpnet for diskusjon rundt konsekvens og effekt hos EDB i Oslo, der tre personer deltok på intervjuet.

Vi fylte ut en og en rad i skjemaet, med unntak av risiko og leverage, som ble regnet ut i etterkant.

11.3 Etterarbeid

I etterkant av møtene skrev vi ut intervjuene basert på våre notater som gjengitt i vedlegg B. På risikoanalyseskjemaene ble risiko og leverage regnet ut. For å regne ut leverage-verdien satt vi høy til verdien 10, høy/middels til 8, middels/høy til 7, middels til 4, middels/lav til 3, lav/middels til 2 og lav til 1. Dette er en vanlig skala å bruke, og gir en maksimal leverage på 999, hvilket vil si at tiltaket er maksimalt lønnsomt å gjennomføre. I andre enden av skalaen kan man oppnå minus-verdier, men der dette var tilfelle justerte vi verdien til 0. Vi ser at noen bedrifter brukte høy på skalaen oftere enn andre, og at dette førte til at disse bedriftene hadde høyere utslag på leverage-verdiene. Vi velger å ikke justere for dette, ettersom vi mener bedriftene har et bedre grunnlag enn oss til å vurdere risiko, og det er mulig at den hyppige bruken av høy er velbegrunnet. Vi tror ikke forskjellene er så store at det blir vanskelig å sammenligne leverage-verdiene mellom bedriftene.

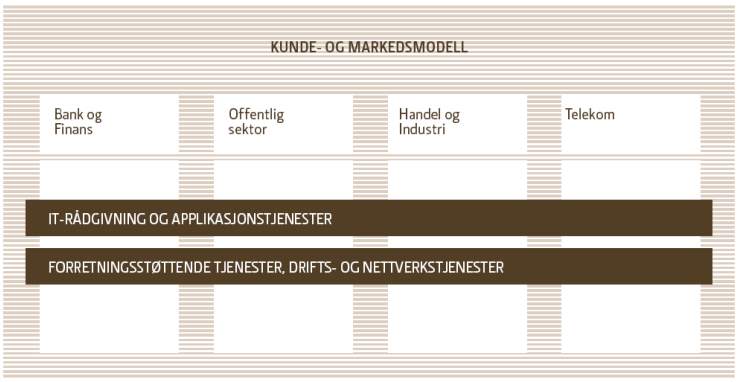
Til slutt sendte vi utskriften og risikoanalysen til intervjuobjektene, slik at de kunne rette opp i eventuelle misforståelser Vi fikk intervjuene tilbake med noen få rettinger utført av intervjuobjektene. Endringene var ikke av en slik art at intervjuene måtte omskrives, men bestod av endringer på for eksempel årstall og navn på prosjekter. Vi mener derfor at denne framgangsmåten sikrer at informasjonen fra intervjuene er gjengitt på en god måte.

Kapittel 12
EDB

Oppdragsgiver for oppgaven har vært EDBs testsenter i Trondheim. Dette kapittelet presenterer denne bedriften, og hvordan de i dag jobber med testing.

12.1 Generelt om EDB

EDB Business Partner er et av Nordens ledende børsnoterte IT-konsern, og har 3 900 ansatte (EDB Business Partner 2007a). EDB har deltatt i norsk IT-historie i mer enn 40 år, etter at Elektronisk Databehandling ble grunnlagt i 1961 (EDB Business Partner 2007b). I dag har EDB flere kontorer i Norge, Sverige og Danmark, og er i stadig vekst. Selskapet fokuserer på to områder, nemlig drift og løsninger. EDB omsatte i 2006 for 5,8 milliarder kroner.



Figur 12.1: EDBs kunde- og markedsmodell (EDB Business Partner 2007b)

Organisatorisk er EDB delt opp i fire bransjeområder og to bransjeuavhengige områder som vist i figur 12.1. De fire bransjeområdene har ansvar for leveranser innenfor sitt område. De fire bransjeområdene er:

- Bank og finans
- Offentlig sektor
- Handel og industri
- Telekom

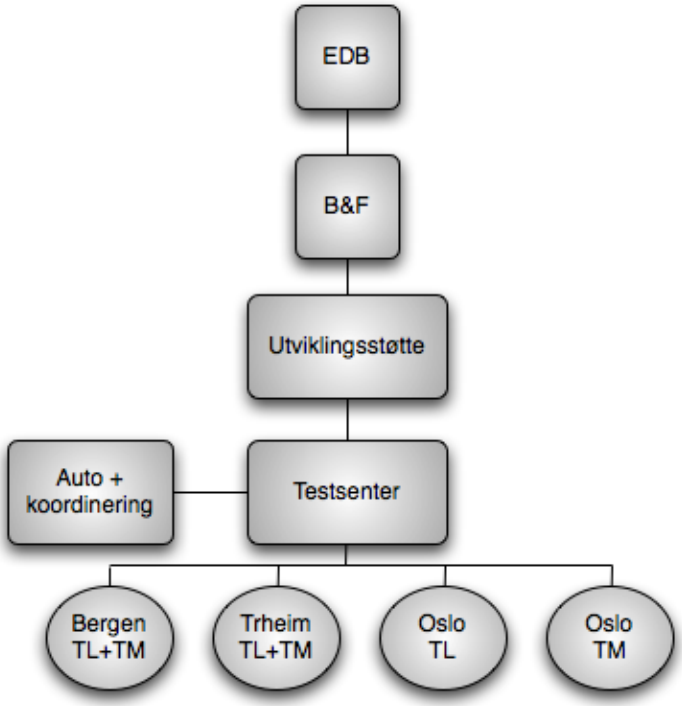
De to bransjeuavhengige områdene skal håndtere leveranser på tvers av områdene:

- IT-rådgivning og applikasjonstjenester
- Drift- og nettverkstjenester

Vår samarbeidspartner i arbeidet med denne oppgaven har vært EDB Bank og finans, så når betegnelsen EDB blir brukt vil det være nettopp dette bransjeområdet vi henviser til.

12.2 EDBs testavdelinger

EDB har testavdelinger i Trondheim, Oslo og Bergen, der Oslo er den største avdelingen. Figur 12.2 viser et organisasjonskart over EDB, og hvordan



Figur 12.2: Organisasjonskart EDB

testavdelingene er organisert. Vivi Horn er leder for testsenteravdelingen, som består av fire enheter. Bergen og Trondheim har testledelse og testmiljø i samme avdeling, mens Oslo er delt i to avdelinger. Testavdelingen i Trondheim ledes av Trygve Lauritzen, og Oslo-avdelingen ledes av Arve Loraas. Bergensavdelingen har vi ikke hatt noe kontakt med under arbeidet med oppgaven. Testavdelingene er en undergruppe av Bank og finans og tester løsninger som:

- Kortløsninger
- Minibanker
- Nettbanker

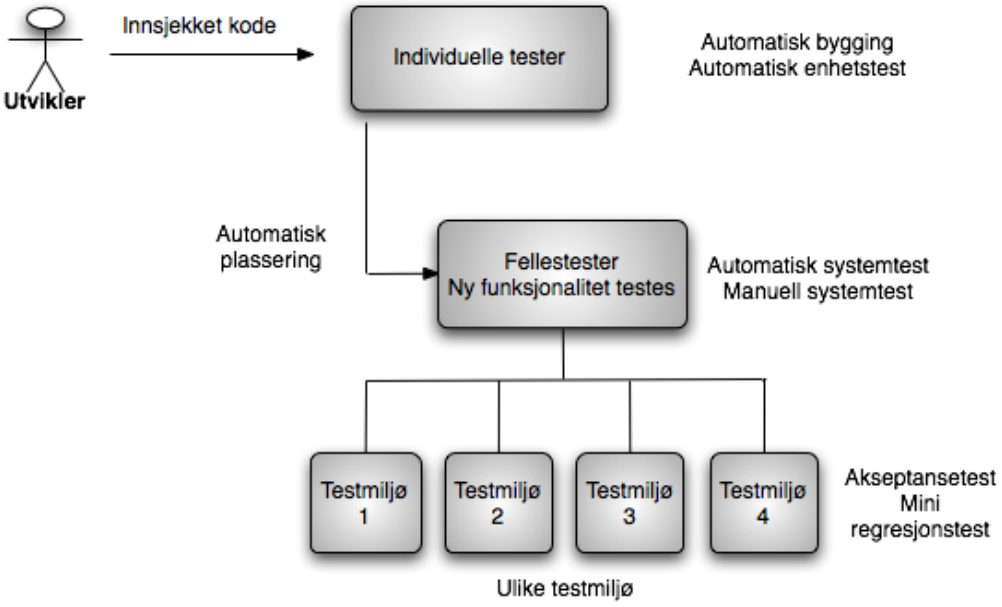
- Portaler
- Mobilbanker

I arbeidet vårt har vi hovedsakelig samarbeidet med Trondheimsavdelingen, men har avholdt et møte med Oslo-avdelingen på deres kontor i Nydalen.

12.2.1 TMM i EDB

Testavdelingen bruker ulike elementer på TMM-nvåene som mål for sitt forbedringsarbeid. Avdelingen anser seg som god på testmetodikk og utviklingsmetodikk, men med forbedringspotensiale på testmiljø.

12.2.2 Framgangsmåte ved testing



Figur 12.3: Testing ved EDB Trondheim

Figur 12.3 viser hvordan Trondheimsavdelingen jobber når de tester programvare. Etter at kode er sjekket inn av utviklerne, kjøres individuelle tester. Deretter sendes koden til fellestesting der ny funksjonalitet testes ved hjelp av automatisk og manuell systemtest. Hittil har koden blitt testet på ett testmiljø, men akseptansetest, mini-regresjonstest og ytelsestest kjøres på flere testmiljø.

Testmiljøet for enhetstester kan bygges med annen maskin - eller programvare enn produksjonsmiljøet, eksempelvis på maskiner som kjører en Linux-distribusjon. System-, akseptanse-, og ytelsestester kjøres på samme programvare som i produksjonsmiljøet, men kan kjøres med nedskalert maskinvare, dog med samme produsent som i produksjon. På denne måten kan for eksempel den faktiske ytelsen estimeres ved hjelp av konverteringstabeller publisert av leverandørene for maskinvaren.

12.3 EDBs testmetodikk

Ettersom testing er en stor del av EDBs virksomhet, er det utarbeidet en grundig testmetodikk. Denne beskriver hvordan testing skal gjennomføres hos EDB og inneholder alt fra hvorfor testing er viktig til hvem som skal gjennomføre hva og hvilke dokumenter som skal produseres under og etter testing.

12.3.1 Testnivå hos EDB

EDB Bank & Finans har valgt å dele testene sine i tre nivåer, nemlig leveransetest, systemtest og enhetstest. Disse testnivåene ble diskutert på et teoretisk nivå i delkapittel 5.4. EDBs definisjoner av testnivåene finnes i testmetodikken, og er gjengitt nedenfor:

Enhetstest Test for å prøve at programmet møter de krav som er satt i teknisk spesifikasjon av programmet, og at programmet er integrert med sine nærmeste omgivelser.

Systemtest Test av om et enkeltprogram er fullt ut integrert mot de øvrige systemer det skal virke sammen med, og at et system eller en sammensetning av systemer tilfredsstiller funksjonelle krav og krav til egenskaper for systemet.

Leveransetest Testing for å vise at det totale systemet tilfredsstiller funksjonelle og kvalitetsmessige krav, der ny funksjonalitet er integrert med eksisterende funksjonalitet (EDB Business Partner Norge AS 2006).

Som vi ser, benytter EDB seg av leveransetest. Dette er den siste testen som gjennomføres på et prosjekt. Den har samme rammer som en akseptansetest, og blir i praksis som en intern akseptansetest. I testmetodikken spesifiseres akseptansetest som en test som gjennomføres dersom det er en ekstern leveranse, og her kan kunden enten bestemme opplegget for testen, delta eller få innsyn i EDBs leveransetest og godta denne som sin akseptansetest (EDB Business Partner Norge AS 2006). Integrasjonstesting påbegynnes under enhetstest, og fullføres normalt som første del av systemtest.

EDB benytter seg av V-modellen som ble vist i figur 5.6 i delkapittel 5.6, modifisert med deres tre testnivåer. Dette medfører at de tilstreber å starte arbeidet med et testnivå parallelt med tilsvarende fase i systemutviklingen (EDB Business Partner Norge AS 2006).

12.3.2 Testmetodikkens rammer for testmiljø

EDBs testmetodikk omfatter en egen del om testmiljø, som tar for seg teknisk testmiljø, med linker til overblikk, beskrivelse og bruk til delmiljøene, og et overblikk over testmiljøet i Bank & Finans. Delen er plassert bakerst i testmetodikken, og strekker seg over seks sider. Det er likefullt tydelig at håndtering

av testmiljø ikke er like spesifisert som andre deler av testmetodikken. Dette gjelder for eksempel ved bruk av testmiljøbegrepet som ikke er entydig definert.

> "Begrepet testmiljø brukes på flere nivåer og er derfor ikke entydig."
> (EDB Business Partner Norge AS 2006)

I tillegg står den om testmiljø på ulike steder i resten av rapporten, temaer som tas opp inkluderer nedetid i testmiljø, testmiljøbehov og testmiljøansvarligs ansvarsområder.

Overblikk over testmiljøet

I følge testmetodikken er "teknisk testmiljø maskinvare, programvare og testdata som er tilrettelagt for testing (EDB Business Partner Norge AS 2006). Metodikken slår videre fast at et testmiljø skal finnes på alle plattformer man utvikler for eller på. Testsenteret er ansvarlig for å administrere testmiljøet, utføre tester og besørge informasjon til dem som trenger det. Vedlikehold av testmiljøene deles inn i maskinrettet vedlikehold, datarettet vedlikehold og nettverks- og kommunikasjonsvedlikehold.

Figur 12.4 viser et eksempel på testmiljø fra en nettbank. Vi ser at testmiljøet består av mange komponenter, og er komplekst.

Hva skal et testmiljø omfatte

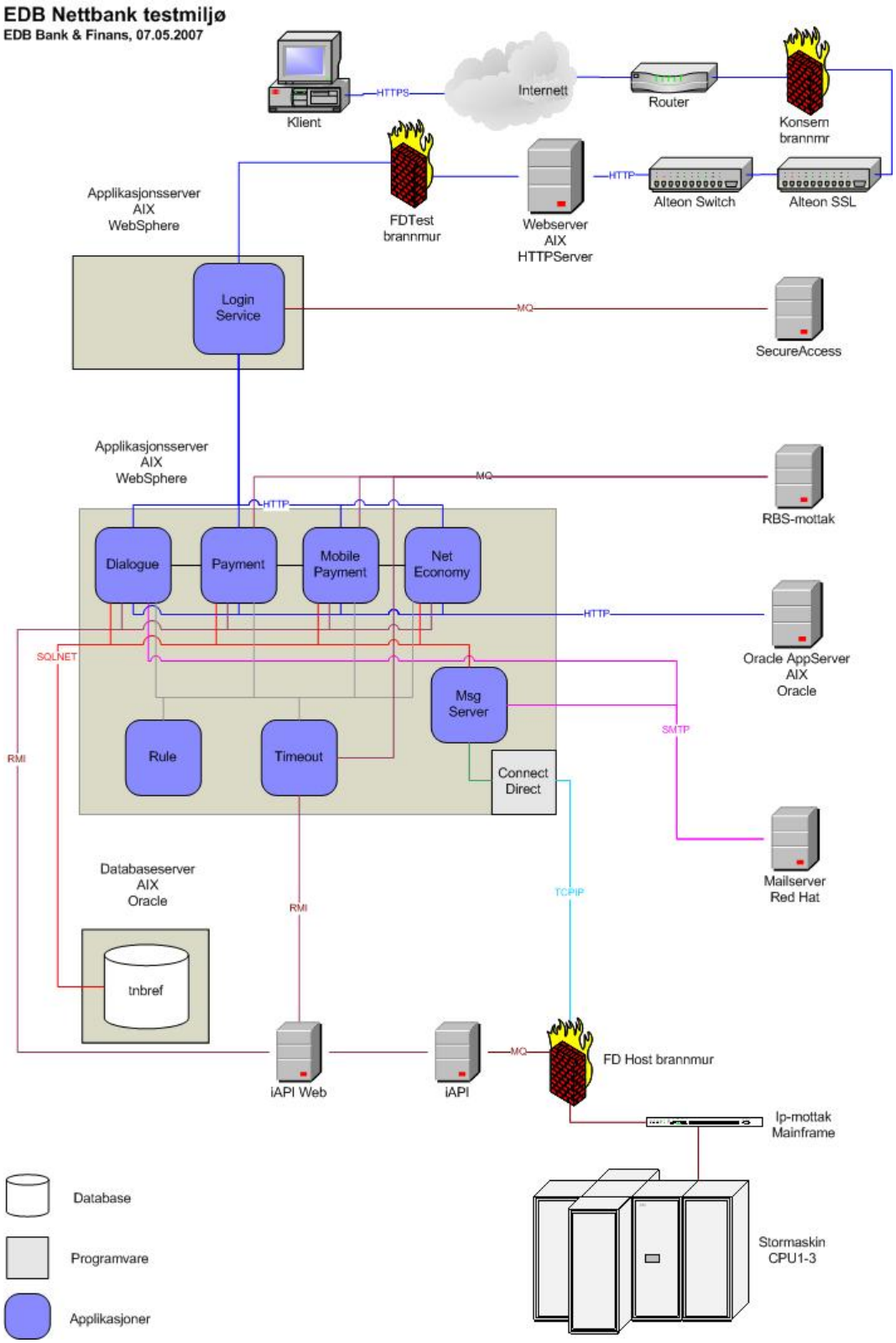
Hvilke komponenter og deler et testmiljø må omfatte, vil variere med hvilket testnivå man skal bruke det i. EDBs testmetodikk sier at systemtester skal utføres i et miljø som tilsvarer produksjonsmiljøet systemet skal installeres i, og at data skal være produksjonslike (EDB Business Partner Norge AS 2006). Minimumskriteriene for et testmiljø er at det skal omfatte systemene som er aktuelle for test av funksjonaliteten i det som er fokus for den utvikling som ligger til grunn for testen.

Leveransetester skal gjennomføres i et testmiljø som er "tilnærmet likt et produksjonsmiljø i teknisk oppbygging, og i utgangspunktet skal alle systemer også være i miljøet" (EDB Business Partner Norge AS 2006).

Nedetid i testmiljø

I følge testmetodikken har testleder ansvar for å fortløpende registrere hendelser som forårsaker nedetid i testmiljøene. Hendelsene skal årsaksforklares basert på forhåndsdefinerte årsaker. Med fargekoder registreres status på leveransene, der rød er meget utilfredsstillende, gul er utilfredsstillende og hvit er tilfredsstillende.

Testmetodikken omhandler ikke hva som kan gjøres for å forhindre nedetid i testmiljøene.



Figur 12.4: Eksempel på testmiljø

Testmiljøbehov

Testmetodikken krever at en testmiljøplan alltid skal lages, enten som et eget dokument eller som et kapittel i testplanen. Hovedtrekkene til innhold er spesifisert, og inkluderer maskinvare, testdata, programvare og personell.

12.3.3 Verktøy i bruk hos EDB i dag

I testmetodikken er det spesifisert retningslinjer for hvilken maskin- og programvare EDB skal bruke til testing i ulike prosjekt.

Quality Center Hovedverktøyet til EDB, og brukes til en rekke formål, blant annet planlegging og kjøring av automatiske og manuelle tester, registre feil funnet under testing, med koder for når feilen skal rettes og hvor i utviklingsfasen feilen ble introdusert, og beskrivelse av testtilfeller for system- og levereansetester.

XAL Datafangsverktøy, brukes til registrering av styringsdata.

ARS Også et datafangstverktøy som brukes til registrering av styringsdata.

LoadRunner Brukes til ytelsestester

LoadController Makker til LoadRunner som muliggjør spilling av ett eller flere skript på en eller flere maskiner til en får fram den riktige lasten for konfigurasjon av leveransen en tester.

WinRunner Tester grafiske brukergrensesnitt.

Som skriptspråk benyttes Test Script Language (TSL) og QuickTest Professional (QTP).

Testdata

For å kunne teste bank- og finanssystemer, er det nødvendig med realistiske og gode testdata. Det er vanlig å teste på ekte data, for eksempel kontotransaksjoner til eksisterende kunder. Dette kan være et etisk dilemma, fordi kunden ikke vet om dette, og kan mislike at personlige data blir brukt til testformål. Det er derfor viktig å definere klare rutiner, tilgangskontroll og taushetsplikt ved slik testing. Alternativet til å bruke ordentlige testdata, nemlig selv generere data, er sannsynligvis ikke godt nok.

Det er viktig at testdata ikke kommer på avveie, og i uheldigste fall havne i produksjonsmiljøet, og testmetodikken inneholder noen retningslinjer for å hindre dette. Testmetodikken slår også fast at testdata stort sett eies av Bank og finans' kunder, og at disse derfor skal ha melding dersom data brukes til testing, og at dette skal være en del av Testsenterets administrative rutiner.

Kapittel 13
Bedriftene vi intervjuet

Dette kapittelet presenterer de fire bedriftene utenom EDB vi har gjennomført samtaler med. Hvordan bedriftene har blitt valgt ut er beskrevet i kapittel 11.

13.1 Domstoladministrasjonen (DA)

Domstoladministrasjonen (DA) er den administrative overbygningen for landets domstoler og jordskifteretter; Høyesterett, 6 lagmannsretter, 74 tingretter, 5 jordskifteoverretter og 34 jordskifteretter(Domstoladministrasjonen 2007). DA ble skilt ut fra Justisdepartementet i 2002, etter et vedtak i Stortinget. Dette for å løsrive påtalemyndigheten fra politisk styre.

DA ble flyttet til Trondheim i 2005 og har i dag 90 ansatte. De utvikler ikke programvare selv, men drifter og administrerer store datasystem for domstolene. Vi har snakket med Ståle Risem-Johansen som er ansvarlig for test og kvalitetsledelse hos DA.

Testmiljøet til DA skal tilsvare et produksjonsmiljø på rundt 20 servere, der mye av informasjonen er svært sensitiv. Av den grunn er det store begrensninger på hvem som har tilgang til produksjonsmiljøet, og man er avhengig av et testmiljø for å kjøre tester. Selv om utviklingen skjer hos leverandører, bygger og drifter de selv testmiljøet.

13.2 Helse Midt-Norge IT (HEMIT)

HEMIT er IT-avdelingen til Helse Midt-Norge og har eget styre. HEMIT ble etablert som en egen regional IT-enhet den 1. juni 2003, og de har ansvar for å skaffe IT-tjenester til sykehusene i Møre og Romsdal, Sør-Trøndelag og Nord-Trøndelag(Helse Midt-Norge IT 2007). Dette inkluderer 6 helseforetak og 9 sykehus, og inkluderer drift av IT-tjenester for 17 000 brukere.

Selve utviklingen av datasystemene foregår eksternt, men planlegging og drift har HEMIT ansvaret for. Til sammen har de fire store fellesprosjekter i tillegg til flere små enkeltsystemer. De fire fellesprosjektene er:

- Elektroniske pasientjournaler
- PACS RIS Røntgensystem

- PAS Pasientadministrasjon
- LAB Laboratoriedatasystem

HEMIT bygger og drifter også et større testmiljø der de tester systemene sine. Dette delkapittelet bygger på en samtale med Vidar Myklebust som er ansvarlig for testmiljø hos HEMIT.

13.3 IBM/Telenors 2000-års prosjekt

Vi har snakket med Hans Erik Ballangrud som har lang erfaring med testarbeid. Han jobber i dag som avdelingsleder for Rational og Tivoli sin tekniske gruppe i IBM Software Group, men har tidligere erfaring med test fra flere bedrifter og større prosjekter. I forbindelse med år 2000, var han ansvarlig for å gjennomføre et større prosjekt hos Telenor som skulle forhindre at de fikk IT-relaterte problemer ved overgangen til år 2000. Svarene til spørsmålene under vil derfor være i forbindelse med nettopp dette prosjektet og ikke dagens arbeidssituasjon i IBM.

Telenors 2000-års prosjekt startet et delprosjekt med å etablere et testsenter i mars 1997, og det ble avsluttet i etterkant av årtusenskiftet i år 2000.

13.4 Abeo

Abeo er et konsulentselskap med 85 ansatte i Trondheim og Oslo. De ble etablert i 1997 og har i dag konsulenter i flere prosjekt (Abeo 2007). Abeo fokuserer på fire hovedområder:

- Strategisk rådgivning og prosjektledelse
- Etablering av tjenesteorientert arkitektur
- Identitetshåndtering
- realisering av skalerbare løsninger (Abeo 2007)

Fra Abeo har vi snakket med Jan-Erik Hagelund som har jobbet med systemutvikling og test siden 1985. Hagelund jobber i dag som konsulent hos Abeo og har blant annet ledet en gruppe på 10 mennesker som testet for Q-free[1].

[1]Q-Free et selskap som levererer utstyr, programvare og tjenester for veiprisingssystemer, tilgangskontroll- og sporingssystemer, billettløsninger og tjenester knyttet til dette.

Kapittel 14
Oppsummering av intervju

Dette kapittelet oppsummerer de gjennomførte intervjuene. Spørsmålene vi stilte er gjengitt i vedlegg A, og utskrevne intervjuer finnes i vedlegg B. Framgangsmåten ved intervjuer er beskrevet i kapittel 11.

14.1 Metodikk for testmiljø

Metodikker for testmiljø var en viktig del av intervjuet. Selv om det ikke var vanlig med en offisiell nedskrevet testmiljømetodikk, diskuterter vi her hva som finnes av testmetodikker, hvilke føringer den legger og hvilke kilder som er benyttet.

14.1.1 EDB

Dokumentert metodikk

EDB i Oslo forteller at de ikke har en egen testmiljømetodikk, men at bedriftens testmetodikk har et avsnitt om testmiljø. Avsnittet forteller ikke hvordan testmiljø skal bygges og brukes, men at det skal finnes et testmiljø og en ansvarlig for dette.

EDB i Trondheim bekrefter at testmetodikken inneholder et avsnitt om testmiljø, men at det ikke finnes noe nedskrevet metodikk for testmiljø utover dette. Avsnittet inneholder mer om hva et testmiljø skal omhandle enn hvordan det skal bygges. Internt er det kommunisert at et testmiljø bør bygges ved hjelp av samme metoder som i produksjon og med nedskalert maskinvare fra samme produsent. På denne måten kan man oppnå produksjonslikhet. Bygging av testmiljøet går gjennom IT-drift ved at et bestillingsark med spesifikasjoner på komponenter, nettverk, programvare og så videre leveres til dem.

Føringer for bedriften

EDB i Oslo påpeker at siden testmiljømetodikken ikke er nedskrevet, er det så og si ingen føringer på bygging og bruk av testmiljø. Avsnittet fra testmetodikken gir imidlertid noen organisatoriske føringer på hvordan testmiljøavdelingen skal være bygd opp, og hvilke tester som skal gjennomføres

i testmiljøet. Et eksempel som nevnes er at definisjonen av akseptansetest gir føringer for at akseptansetestmiljøet skal være produksjonslikt. På denne måten legger testmetodikken visse føringer på arbeidsrutinene i avdelingene. EDB i Trondheim mener at testmiljømetodikken legger sterke føringer på system- og akseptansetest, i tillegg til valg av arkitektur. Som rådgivere på maskinvare brukes kompetansen til IT-drift og 3. partsleverandører.

Bakgrunn for metodikk

For å lage metodikken for testmiljø som eksisterer i dag har Trondheim støttet seg på IT-drift og deres erfaringer, og produksjonsmiljøet. Man har også satset sterkt på kursing og sertifisering av ansatte, blant annet innen AIX, Java og Oracle. Også interne fagfora er en viktig kilde til opplæring. Oslo er enig i at erfaringer internt i bedriften er hovedkilden, da man har ikke funnet mye stoff i litteraturen.

Det arbeides nå med å lage en testmiljømetodikk, og målet er at denne skal være ferdig i løpet av 2008.

14.1.2 Andre bedrifter

Dokumentert metodikk

Med unntak av Telenors 2000-årsprosjekt har ingen av de fire andre bedriftene en offisiell, dokumentert metodikk for testmiljø. Bedriftene har likevel sine framgangsmåter for testmiljø, selv om disse ikke er dokumentert.

Domstoladministrasjonen har ikke en nedskrevet metodikk hverken for testmiljø eller test. Kunnskap om testmiljømetodikk finnes i form av erfaring blant de ansatte, men dette er problematisk ved nyansettelser. For å forbedre dette er man i ferd med å utvikle et kompetansesystem.

HEMIT har heller ikke en nedskrevet metodikk for testing eller testmiljø. Det arbeides med å etablere en metodikk for testmiljø, som vil bygges i samsvar med en eksisterende standard for arkitekturstrategi basert på SOA.

Abeo skiller seg ut fra de øvrige bedriftene ved at det er et konsulentselskap, og derfor er mye av framgangsmåten avhengig av kunde og prosjekt. Abeo har derfor ikke en egen offisiell, nedskrevet metodikk. Ofte vil man skrive litt om testing i tilbudet som sendes ut til kunde, men hver kunde har forskjellige behov, og det vil derfor være vanskelig å følge en bestemt metodikk.

På motsatt ende av skalaen finner vi Telenors 2000-årsprosjekt, der man i ettertid kan si at prosjektet ble overdokumentert. Det eksisterer flere hyllemeter med dokumentasjon, og man brukte tvang for å få folk til å dokumentere. Testmiljø var et viktig aspekt ved programmet for år 2000, og testmiljømetodikken var en del av en omfattende testmetodikk.

Føringer for bedriftene

Domstoladministrasjonen mener at det som finnes av ikke-dokumentert testmiljømetodikk legger svært få føringer for arbeid med testmiljø i bedriften. Bygging av testmiljø er basert på erfaringer fra drift. Det som er spesielt med Domstoladministrasjonen i forhold til andre bedrifter vi har intervjuet, er at man er avhengig av bevilgninger fra Staten for å gjennomføre store prosjekt. Et eksempel på dette er at man per i dag har ett testmiljø, men som et resultat av bevilgninger i september/oktober 2006 jobber man med å øke dette til tre testmiljø. For virtualisering har DA valgt programvare fra Microsoft, og det at de har et godt samarbeid med Microsoft kan ha påvirket dette valget.

I HEMIT er det størrelsen på systemet som legger føringer på hvordan testmiljøet bygges. De fire større systemene bruker testmaskiner som er spredt i produksjonsmiljøet, og dermed benytter samme nett som i produksjon. De mindre systemene er avhengige av leverandørene, da dette gjerne er systemer som brukes på enkeltavdelinger på et sykehus. I dag pågår det er prosjekt for å lukke nettverket av testmaskiner, både gjennom fysisk flytting og bruk av virtualisering.

Abeo bygger testmiljøene hos kundene, og det er det konkrete prosjektet og dets innhold som legger føringer på testmiljøet. Kunden kommer med sin portefølje, og dette omfatter verktøy kunden disponerer og metodikk for testmiljø.

Telenors 2000-årsprosjekt er den eneste bedriften med en offisiell, nedskrevet testmiljømetodikk, og de sier også at testmiljømetodikken la sterke føringer på at alt skulle være være identisk med produksjonsmiljø. Et eksempel er at man opplevde at programvare systemer skulle kjøre på ikke var tilgjengelig siden man startet testing tidlig. Dette ble løst ved å teste den aktuelle koden i flere omganger etterhvert som programvare ble tilgjengelig.

Bakgrunn for metodikk

Felles for alle bedriftene er at egen, praktisk erfaring er en viktig kilde for testmiljømetodikk.

Erfaringene DA benytter som metodikk er basert på teoretisk kunnskap, som lesing, kurs og konferanser, leverandørinformasjon og praktisk erfaring fra DA og andre bedrifter. Testforumet til Dataforeningen blir nevnt som et forum med mye aktivitet og diskusjoner rundt testing.

I HEMIT bygger testmiljømetodikken stort sett på kunnskapen blant de ansatte, da disse har bygd testmiljø tidligere og sitter på mye kunnskap. I tillegg har SOA vært en kilde med tanke på at man velger å bygge testmiljøet adskilt, der man definerer tjenester i testnettet. Dette bygger også på erfaringen med produksjonsnettet, siden man er avhengig av et testmiljø som ligner på produksjonsmiljøet.

Telenors 2000-årsprosjekt var høyt prioritert blant ledelsen, og hadde muligheten til å benytte mange kilder siden det hadde god økonomi. Testmiljømetodikken bygger på Ballangruds erfaringer, deltakelse på testkonferanser og bøker om testing. Hans Schaefer ble leid inn til opplæring av de ansatte og prosjektet samarbeidet med Vivi Horn og driftsgruppen i TeamCo som dengang driftet Telenors systemer. Konsulenter fra Icon Medialab var rådgivere på prosjektledelse og prosedyredefinering, og QA Labs i samspill med Universitetet i Lund ble involvert, men etterhvert viste det seg at sistenevnte var mindre aktuelle.

I Abeo er kunden en viktig kilde ved bygging av testmiljø, men også IEEEs mal for oppsett av testplan er brukt. I tillegg brukes utviklernes kunnskap, spesielt på enhetstesting.

14.2 Testmiljø i litteraturen

Litteraturstudiumet i kapittel 4 viser at det finnes lite om testmiljø i litteraturen. Her diskuteres hvilken litteratur bedriftene har kjennskap til, og årsaker til at lite litteratur finnes.

14.2.1 EDB

EDB har tidligere kartlagt andre bedrifters testavdelinger, og ikke funnet noen med en så omfattende testavdeling som EDB i Norden. Det er uvanlig at testavdelingen er sentralisert i så stor grad, og EDB har et høyt modenhetsnivå i henhold til TMM. At EDB har kommet så langt mener man er en medvirkende årsak til at EDB ikke finner så mye litteratur.

EDB i Oslo foreslår fire årsaker til at testmiljø ikke er dokumentert i litteraturen i stor grad.

1. Det er lite utbredt blant bedrifter i Norge å ha egne store testmiljøavdelinger.
2. Testere er stort sett teknikere, og teknikere liker generelt ikke å dokumentere.
3. Testmiljøet ligger nært produksjonsmiljøet, og man kan derfor bruke driftsmetodikken som testmiljømetodikk.
4. Å bruke testmiljø bevisst er relativt nytt. EDB startet med testmiljø i Trondheim i 1995, og testmiljømetodikk blir gjerne utviklet som en del av modenhetsprosessen i en bedrift.

EDB benytter seg av TMM, og denne modellen har resultert i større krav til dokumentasjon og innføring av nye prosedyrer. Selv om TMM sier mye om testmiljø, blir det ofte tungt og byråkratisk. Det er usikkert om modellen har hjulpet ved bygging av testmiljø, men den kan ha ført til en bedre rollefordeling internt, i følge EDB i Trondheim.

14.2.2 Andre bedrifter

Med unntak av Abeo, er alle bedriftene enig i at det ikke finnes metodikker for testmiljø i litteraturen. Abeo er uenige i påstanden, og mener at slike finnes. Intervjuobjektet kan likevel ikke nevne eksempler på slike metodikker. Abeo mener også at det ofte er gitt hvordan testmiljøet bør bygges. Det bør være så nært som mulig produksjonsmiljøet, vurdert opp mot kostnadene for et slikt oppsett. Der dette er for kostnadskrevende må man vurdere om deler av totalsystemet bør erstattes av dummies.

Domstoladministrasjonen mener det finnes mye om testing, men lite om testmiljø, men har ingen forslag på hvorfor det er slik. HEMIT mener at testmiljø kanskje ikke er interessant nok, selv om dette strider mot hans personlige overbevisning.

Telenor foreslår at utviklingsarbeid blir sett på som mer kreativt, og at det her satses mye på innovativitet. Test assosieres med en mer destruktiv handling og har generelt en lav status, selv om man ser en endring i dette. Telenor var forøvrig den eneste bedriften, i tillegg til EDB, som brukte TMM, men da i en tidlig versjon som ikke sa noe om testmiljø.

14.3 Roller i testmiljø

Hvem som jobber med testmiljø, og hvordan arbeidsrollen deres er definert, varierer fra bedrift til bedrift. Avsnittet tar hovedsakelig for seg testmiljøleders ansvar, og hans/hennes arbeidsoppgaver.

14.3.1 EDB

EDB i Oslo har en egen testmiljøansvarlig på hvert prosjekt, og en person kan være testmiljøansvarlig for flere prosjekt samtidig. Denne personen kan bygge testmiljøet selv, eller delegere større eller mindre deler til de 10 ansatte i testmiljøavdelingen. Ansvarsområdet til testmiljøansvarlig inkluderer også integrasjonstest, systemtest og akseptansetest.

I Trondheim har man også en egen testmiljøansvarlig på hvert prosjekt. Denne personen har ansvaret, men ikke nødvendigvis det utførende ansvaret. Dette er ofte lagt til drift. Man har også en testmiljøkoordinator, som har ansvar for å sikre utnyttelsen av testmiljøene, se behov for oppgraderinger og oppdager svakheter.

14.3.2 Andre bedrifter

I Domstoladministrasjonen er det er opp til hvert enkelt prosjekt å avgjøre om det trengs en egen ansvarlig for testmiljø. Det er vanlig at drift har ansvar for det tekniske ved testmiljøene. I framtiden mener kvalitetsansvarlig at dette

bør være en del av hans jobb og han skal også ha et overordnet ansvar for at testmiljø blir bygget.

Intervjuobjektet i HEMIT er selv hovedansvarlig for testmiljø, men har andre oppgaver i tillegg til dette. De store systemene har gjerne en egen testmiljøansvarlig som har ansvar for at systemet blir bygget, mens det i mindre delsystemer gjerne skjer i samarbeid med systemansvarlig.

Telenors 2000-årsprosjekt ble delt opp i 100 verdikjeder som hvert ble opprettet som et prosjekt. Hvert av disse hadde en egen tesmiljøansvarlig. Testmiljøet ble bygget og drevet av testerne som jobbet i testsenteret.

Abeo har ikke en egen ansvarlig for testmiljø, men intervjuobjektet blir oppfattet som en ressurs på testing. Hvorvidt prosjektet har en egen testmiljøansvarlig kommer an på prosjektets ressurser. Det er også opp til hvert prosjekt hvem som har ansvar for å bygge og vedlikeholde testmiljø. Ofte involveres driftsavdelingen hos kunden, da kunden sjelden har råd til en egen ansvarlig person for testmiljø.

14.4 Testnivå

Det varierer hvilke testnivå man gjennomfører, og i hvilket testmiljø man gjennomfører testnivåene.

14.4.1 EDB

I EDB i Oslo kjøres enhetstester av utviklere i utviklingsmiljøet. Integrasjonstest, systemtest og akseptansetest kjøres stort sett i det samme testmiljøet. Dette er begrunnet økonomisk, da bedriften mener fordelen ved å teste i tre seperate miljø ikke oppveier kostnaden.

I Trondheim har man også et eget enhetstestmiljø for alle prosjekt. Dette er viktig fordi enhetstestmiljøet gjerne har mer nedetid fordi kode sjekkes inn hyppig. Hvorvidt andre testnivå har et eget miljø, varierer fra produkt til produkt, hvor i verdikjeden man befinner seg og kundens engasjement. Et alternativ til seperate miljø kan være å definere hvilke testnivå som kjøres i et testmiljø i hvilken tidsperiode. Integrasjonstestmiljøet blir i de aller fleste tilfeller videreført til et systemtestmiljø. Man må skille systemtestmiljø og akseptansetestmiljø fordi akseptansetester må simulere produksjonsmiljøets oppetid og stabilitet, og derfor er avhengig av at nye versjoner sjekkes inn mindre hyppig enn i systemtestmiljøet, typisk hver 3. dag.

14.4.2 Andre bedrifter

Hvilket testnivå som kjøres og i hvilke testmiljø varierer i stor grad fra bedrift til bedrift.

Ettersom Domstoladministrasjonen ikke utvikler produkter selv har de ikke behov for alle testnivå. Bedriften kjører stort sett akseptansetester og ytelsestester, og ser ingen grunn til at det skal være problematisk at disse kjører i samme miljø.

Heller ikke HEMIT gjennomfører alle testnivåer selv. Enhetstest og systemtest blir gjennomført i utviklermiljøet hos leverandøren, og integrasjonstester og akseptansetester gjennomføres hos HEMIT. Dette er viktig for å teste hver enkeltdel i detalj, samtidig som man tester systemet sammen med andre system i produksjon.

Telenors 2000-årsprosjekt hadde forskjellige testmiljø for testnivåene. I testmiljøet kjørte man kun akseptansetest, og det var forutsatt at alle tester skulle ha gjennomført systemtest før de ble levert inn til akseptansetest. Det ble ansett som for dyrt å kjøre enhetstest i produksjonslike systemer og man mener at man bør ha et så lite testmiljø som mulig for å dekke testen, da testing generelt er dyrt.

I Abeo benyttes forskjellige testmiljø. Hvor mange testmiljø og tester man kjører er igjen avhengig av prosjektets størrelse. Produksjonsbasert akseptansetest kjøres som oftes hos kunden, og er avhengig av kundens ressurser. Ideelt sett bør testmiljøene være seperate, men med økt grad av samtidighet etterhvert som man når høyere testnivå.

14.5 Oppetid og tilgjengelighet i testmiljø

For å gjennomføre tester er man avhengig av tilgang til et testmiljø, og dette er avhengig av oppetiden til systemet.

14.5.1 EDB

Testerne i Oslo mener at testmiljøet er oppe når det skal, selv om oppetiden ikke er 100%. Per i dag er det en subjektiv holdning om et testmiljø har mye nedetid, da nedetid påvirker ulike deler av systemet. EDB jobber med å lage målinger på hvilket system som går oftest ned, ved hjelp av blant annet et system som skal registrere nedetid. Testerne føler testmiljøet er stabilt, men at det blir kommentert de få gangene det er problemer, selv om problemene kan ligge hos eksterne leverandører, som nettverksleverandør. Testerne synes det er uheldig at de får skylden for dette.

I Trondheim jobbes det også med at testmiljø er tilgjengelig når de trengs. Et eget måleprogram der testmiljøleder rapporter om nedetid og konsekvens for prosjektet er innført. I tillegg har man katastrofeplaner som beskriver hvilke tiltak som skal iverksettes, mulighet for gjenoppretting og SLA-planer[1].

[1]SLA- Service Level Agreement- kunde og leverandør blir enige om krav til systemets oppetid

14.5.2 Andre bedrifter

Domstoladministrasjonen har i dag bare ett testmiljø og det er ikke alltid at dette tilgjengelig når det trengs. Dette problemet er estimert til å bli nærmest ikke-eksisterende ved den planlagte etableringen av ytterligere to testmiljø.

HEMIT har generelt en bra oppetid på sine testsystemer. Det presiseres at det ikke foregår testing hele tiden, men at testmiljøet er tilgjengelig når hvert enkelt prosjekt trenger det. Et prosjekt har sjelden eksklusiv tilgang i testmiljøet, men dette koordineres, og testmiljøet settes opp slik at det er mulig for flere prosjekt å teste systemer i ulike tidsrom. Når det er behov for nye testmiljø blir disse bygget, gjerne ved hjelp av virtualisering og ved tett samarbeid med driftsavdelingen.

Telenors 2000-årsprosjekt kunne oppleve forsinkelser noen ganger, men testmiljøet var stort sett tilgjengelig når det trengtes. Testmiljøet var en prioritert del hos konsernet, og forarbeidet ble grundig gjennomført, slik at planleggingen av testen og bygging av testmiljøet ble ferdig samtidig. Man testet også testmiljøet ved hjelp av inspeksjoner, og ved å sende transaksjoner gjennom systemet.

I Abeo varierer tilgjengeligheten fra prosjekt til prosjekt. I noen prosjekter er testmiljøet tilgjengelig når det trengs, i andre er det ikke det. Det er også variasjoner på oppetid. Man mener at dette kan ha med prosjektets forståelse av behov for testing, og handler mye om prioritering.

14.6 Akseptanstestmiljøets likhet til produksjon

For høyere testnivå er det viktig at testmiljøet ligner på produksjonsnivået. Det stilles spørsmål om hvordan man kan tilnærme seg produksjonsmiljøet, og hvor likt produksjonsmiljøet man må være. Dette gjelder spesielt i akseptansetestmiljøet.

14.6.1 EDB

EDB i Oslo forteller at i noen tilfeller kan en akseptansetest gjennomføres i produksjonsmiljøet, men den kalles da akseptanse-/pilottest. Testmiljøet er svært produksjonslikt, da de grunnet gode avtaler får gratis programvare og databaselisenser fra leverandørene. Den største forskjellen er at maskinvare som prosessor og minne er nedskalert. Årsaker til dette er at testmiljøet skal være billigere å teste på enn produksjonsmiljøet. Produksjonlikheten gjelder ikke enhetstestmiljøet, her kjører man stort sett på gratis programvare, og dette kan føre til feil ved overgangen til mer produksjonslikt testmiljø.

EDB i Trondheim mener at det er mulig å gjøre et akseptansemiljø produksjonslikt, men at dette samtidig er svært kostbart. Avdelinger bruker i likhet med Oslo nedskalert maskinvare, som mindre CPU og minne, i tillegg til å

ha flere applikasjoner på serveren samtidig. Man mener at tilnærmingen til bygging av testmiljø er veloverveid, og den er vurdert til å ha svært liten risiko.

14.6.2 Andre bedrifter

Produksjonsmiljøet til DA er relativt komplisert med et stort antall servere og DA ønsker å framstille dette ved hjelp av fire-fem servere og virtualisering. Kvalitetsansvarlig mener at det bør være mulig å få akseptansetestmiljø likt produksjonstestmiljø. For at systemer skal testes ordentlig før de kommer i produksjon bør testmiljøet være produksjonslikt. Da deler av produksjonsmiljøet ligger utenfor DA, samarbeider de blant annet med Brønnøysundregistrenes testmiljø. Det påpekes at testdata bør være lik de man benytter i produksjon. Dette gjøres i DA ved eksport og vasking av data hver natt fra produksjon.

HEMIT mener at man aldri vil få et akseptansetestmiljø helt likt et produksjonsmiljø i volum, men at de bør være like oppsettsmessig og funksjonelt innholdsmessig.

Telenor er enig i at det i praksis ikke er mulig å få et akseptansetestmiljø likt produksjon. Det er mulig å oppnå produksjonslikhet gjennom simulering, men ikke 100% likhet. Dette er heller ikke nødvendig, da man kan nedskalere med tanke på kompleksitet og antall samtidige brukere. I 2000-årsprosjektet ble virtualisering ikke brukt, da dette ikke var vanlig på denne tiden. Man brukte istedenfor partisjonering. I dag ville derimot virtualisering ha blitt benyttet.

14.7 Sammendrag

Kapittelet oppsummerer de viktigste temaene vi gjennomgikk under intervjuene med bedriften, og vil derfor være en pekepinn på hva vi vil diskutere i del 3. I kapittel 17 diskuterer vi tema som hvilke kilder som finnes når man skal benytte testmiljø og hvor likt produksjonsmiljøet akseptansetestmiljøet må være, på bakgrunn av funnene i dette kapittelet. Dette kapittelet er også en del av grunnlaget når vi presenterer retningslinjene for testmiljø i kapittel 18.

Tema	EDB	Abeo	DA	HEMIT	Telenor
Metodikk for testmiljø	Metodikk for testmiljø en del av testmetodikk. Få føringer på bygging og bruk av testmiljø. Erfaringer internt er hovedkilden, spesielt fra drift og produksjon.	Konsulentselskap, derfor ingen dokumentert metodikk for testmiljø. Det konkrete prosjektet og dets innhold legger føringer på testmiljøet. Kunden en viktig kilde ved bygging.	Ingen dokumentert metodikk for testmiljø. Få føringer, bygging basert på erfaringer fra drift. Ikke-nedskrevet metodikk bygger på teoretisk kunnskap og praktisk erfaring.	Ingen dokumentert metodikk for testmiljø. Størrelsen på systemet legger føringer på bygging av testmiljø. Ikke-nedskrevet metodikk bygger hodedsakelig på kunnskap blant de ansatte.	Dokumentert metodikk for testmiljø. La sterke føringer ved bygging av testmiljø. Metodikk bygger på praktisk erfaring og teoretisk kunnskap.
Testmiljø i litteraturen	Benytter TMM. Har funnet lite ellers.	Mener metodikker for testmiljø finnes.	Det finnes mye om testing, men lite om testmiljø.	Lite om testmiljø, kanskje ikke interessant nok.	Lite om testmiljø, kanskje utvikling blir sett på som mer kreativt.
Roller i testmiljø	Egen testmiljøansvarlig på hvert prosjekt. Ansvar, men ikke nødvendigvis utførende. Også testmiljøkoordinator.	Om prosjektet har en egen testmiljøansvarlig kommer an på prosjektets ressurser.	Opp til hvert prosjekt å avgjøre om man trenger en ansvarlig for testmiljø.	Intervjuobjekt hovedansvarlig for testmiljø. Ellers kommer det an på størrelsen på prosjektet.	Egen testmiljøansvarlig for hvert prosjekt.
Testnivå	Enhetstester i utviklingsmiljø. Om andre testnivå har et eget testmiljø varierer fra produkt til produkt.	Hvilke testnivå som kjøres, og i hvilke testmiljø varierer fra bedrift til bedrift.	Ikke behov for alle testnivå. Kjører akseptanse- og ytelsestest i samme miljø.	Enhetstest og systemtest gjennomføres i utviklingsmiljø hos kunde. Integrasjonstester og akseptansetester i testmiljø hos HEMIT.	Ulike testmiljø for alle testnivå. Enhetstest i utviklingsmiljø. Generelt et så lite testmiljø som mulig for å dekke testen.
Oppetid og tilgjengelighet	Testmiljøet er oppe når det skal. Måleprogram rapporterer om nedetid og konsekvens.	Tilgjengeligheten varierer fra prosjekt til prosjekt.	I dag ett testmiljø som ikke alltid er tilgjengelig. Vil bedre seg med innføringen av ytterligere to testmiljø.	Testmiljøet er oppe når det skal.	Kunne oppleve forsinkelser, men stort sett tilgjengelig ved behov.
Akseptanse-testmiljøs likhet til produksjon	Svært produksjonslikt testmiljø grunnet leverandøravtaler. Maskinvare nedskalert. Mulig å gjøre likt, men svært dyrt.		Produksjonslikt system, ønsker å bruke virtualisering. Samme testdata som produksjon.	Likt oppsettsmessig og funksjonelt.	Produksjonslikhet, men ikke 100%. Nedskalering av kompleksitet og antall samtidige brukere.

Figur 14.1: Oppsummering av kapittel

Kapittel 15
Oppsummering av risikoanalyser

Det er, som beskrevet i kapittel 11, gjennomført risikoanalyse i fem bedrifter. Risikoanalysene finnes i vedlegg C. Vi vil i dette kapittelet oppsummere de viktigste hendelsene sammen med tilhørende tiltak.

Gjennomføringen av risikoanalysene viser at det er stor forskjell i de risikoene de enkelte bedriftene identifiserer. Ettersom risikoene har kommet fram i en idemyldringsrunde, betyr dette at selv om en bedrift ikke har nevnt en risiko kan den likevel være aktuell. Det kan rett og slett være at man ikke kom på det i denne situasjonen. Det er likevel mange likhetstrekk i analysene, og vi har i denne oppsummeringen valgt å dele risikoene i tre hovedgrupper;

- Organisatoriske risikoer
- Tekniske risikoer
- Samarbeidsrisikoer

Disse hovedgruppene er delt opp i etterkant av gjennomføringen, og vil til tider være delvis overlappende.

15.1 Organisatoriske risikoer

For organisatoriske risikoer var det stor forskjell på hvor mye dette ble lagt vekt på under risikoanalysene. Utslagsgivende her kan være arbeidsrollen til de vi snakket med. De som jobbet med testing og oppsett av testmiljø la ikke like mye vekt på organisatoriske risikoer som de som hadde administrativt arbeid.

Det var likevel en del gjengangere når det kommer til det organisatoriske: Manglende ressurser, manglende støtte i ledelsen og forholdet til dokumentasjon.

15.1.1 Ressurser

Flere av bedriftene kommenterte at det ofte var et problem å få de ressursene man behøvde for å bygge og bruke testmiljøet. Dette var alt fra penger til å kjøpe programvare eller maskinvare, til nok testere med kompetanse på området. Manglende ressurser kan føre til at tester ikke blir gjennomført, og de tiltakene som ofte ble nevnt var at ledelsen burde bevilget mer penger til prosjektet. Det ble også nevnt at bestillingsrutinene var for tungvindte og

at man ved å gi testavdelingen større frihet til å bestille selv, kan redusere denne tiden. Andre tiltak som var nevnt er at man må være tydelig på hva man trenger tidlig i prosjektet, og at man bør skape forståelse på tvers av grupperingene, testere, utviklere og ledelse.

15.1.2 Støtte hos ledelsen

Støtte hos ledelsen er viktig når man gjennomfører testaktiviteter. Dette gjelder ikke bare tildeling av ressurser, men også fordi det kan oppstå problem dersom ledelsen ikke forstår nytten av testarbeidet. Dette samsvarer med det som ble sagt i intervjuene, der flere oppga det som svært viktig at ledelsen forstod hvor viktig testarbeidet var. Tilbakemeldingene går på at testarbeid generelt har lav status i Norge og at dette er et problem, da testerne ikke føler seg verdsatt eller får den innvirkningen i prosjektet som de føler de trenger. For eksempel ble det identifisert som en risiko at det er manglende involvering med testfokus fra dag en. De tiltakene som foreslåes er økt fokus og at man stiller krav til organisasjonen, noe man mener at ledelsen bør være ansvarlig for.

Andre risikoer som kommer under ledelseskategorien går stort sett på arbeidet til testerne. Det er nødvendig at de som skal teste systemet får den opplæringen de trenger, noe som igjen er avhengig av at ledelsen har lagt til rette for dette. I tillegg finnes det en risiko i måten de ansatte jobber på. Det kan være at man har en manglende strukturering av arbeidsoppgaver. Det er vanlig å dele en arbeidsoppgave i flere deler og først ferdigstille de delene man føler at man behersker. Her ville det derimot vært mer hensiktsmessig å gjøre det man ikke kan først, da man så vet om dette er mulig å gjennomføre. Denne rekkefølgen kan føre til at man kaster bort unødvendig mye tid, og bør derfor styres gjennom prosjektmetodikk. En annen risiko som nevnes er at noen ansatte har en tendens til å opponere mot rutiner. Dette spesielt ved innføring av nye rutiner, men også generelt ved rutiner. De tiltakene som nevnes for å unngå dette er motivering, forklaring og salg av arbeidsrutiner til de ansatte. Dette bør i følge risikoanalysen komme fra ledelsen.

15.1.3 Dokumentasjon

Manglende dokumentasjon blir kun nevnt som egen risiko i en risikoanalyse, men viser seg likevel å være viktig, da flere risikoer har dokumentering som tiltak. Det kan være alt fra at testerne ikke har fått tilstrekkelig dokumentasjon om det forretningsområde de skal teste, og derfor ikke har nok kunnskap til å gjennomføre testene, til at testmiljøet ikke er som planlagt. Generelt blir det nevnt hos flere at det er viktig med god dokumentasjon av testmiljøet, spesielt når man skal samarbeide med andre, som for eksempel utviklere, leverandører og de som er ansvarlige for utstyrsbestillinger.

15.2 Samarbeidsrisikoer

En stor del av arbeidet med testmiljøet foregår i samspill med andre. Det produktet som skal testes kommer vanligvis fra en utviklingsavdeling enten internt eller eksternt, og det er derfor viktig med et godt samarbeid. Dette skaper også flere risikoer.

15.2.1 Mottak av kode

Det er flere risikoer forbundet med den koden som mottas til testing av testgruppa. Et problem som går igjen hos flere bedrifter er at koden som kommer fra utvikling kommer for seint og at den har for lav kvalitet. Når utviklerne leverer etter fristen, må testtiden reduseres, da man sjelden vil utsette levering. Dette skaper et høyt press mot testerne som må jobbe ekstra og/eller teste mindre, og til slutt kan dette resultere i lavere kvalitet på det ferdigstilte produktet. Det som nevnes av tiltak er svært likt de organisatoriske tiltakene og inkluderer bedre planlegging, tidligere involvering fra testavdelingen og forståelse av hvor viktig testingen er.

Et annet risikomoment er kvaliteten på koden når den blir mottatt. Det kan være så mange feil i koden fra utviklerne at det er umulig å teste. Dette gjelder spesielt for høynivåtester, da man her allerede skal ha gjennomført tester på lavere nivå, og man forventer at koden skal ha få feil. Dette fører til forsinkelser når kode må sendes tilbake til retting, og ved at utvikling bruker tid på å rette feil. Når koden kommer fra eksterne utviklere er det også et problem at koden ikke leveres i henhold til etablerte rutiner. For å forhindre problem med programvarekvaliteten, blir det foreslått to tiltak: Å etablere bedre rutiner eller å kontraktsfeste kvaliteten på leveransen.

15.2.2 Mottak av testdata

Testdata er svært viktig for gjennomføringen av tester i testmiljøet, og det er dermed forbundet med risiko å få tak i riktig testdata. For å få tak i testdata er testerne avhengig av at noen leverer testdata. Dette kan være andre deler av bedriften eller eventuelt kunden, slik tilfelle gjerne er for et konsulentselskap. Det er store utfordringer her, og de tiltakene som nevnes er å kontraktsfeste testdataleveranser og å ta kontakt med testdataleverandør tidlig.

15.3 Tekniske risikoer

Et testmiljø er gjerne komplekst og består blant annet av programvare, maskinvare og nettverkstjenester. Man er avhengig av at utstyret fungerer sammen slik det er spesifisert, og i dette området finnes det flere risikofaktorer.

15.3.1 Testmiljøet dekker ikke testmiljøbehovet

En av gjengangerne i risikoanalysen var at testmiljøet ikke fungerer som spesifisert. Det kan være at testmiljøet ikke er som planlagt, at det ikke er dimensjonert for den testingen det skal gjennomføre eller at det rett og slett er feil i testmiljøet. Det nevnes spesielt to tiltak for å unngå dette. For det første er det viktig å etablere gode rutiner og prosedyrer for etablering av testmiljø. For det andre foreslåes det at man kan utnytte automatikk ved at man automatiserer kontrollmekanismer og/eller gjennomfører automatisk bygging av testmiljø. Automatisk kontroll vil blant annet være med på å redusere sjansen for at man tester på feil versjon i testmiljøet.

15.3.2 Kvalitet på testmiljøet

I tillegg til at det er risiko forbundet med å opprette feil testmiljø, er det også utfordringer rundt den teknologien som benyttes. Man er avhengig av et stabilt testmiljø og det finner derfor en risiko for at testmiljøet blir ustabilt og med dårlig ytelse. Teknisk svikt kan føre til at enkelte deler av testmiljøet kan bli slitt ut og gå ned. Dette blir spesielt en utfordring når man skal erstatte enkeltdeler med ny teknologi og man får utviklere som er ferske på nettopp denne teknologien. Det er likevel slik at enkelte bedrifter til tider venter vil dette vil skje, for så å ha gode rutiner for å rette opp denne hendelsen. Disse tiltakene er tiltak for å redusere effekten av hendelsen, som nevnt i figur 9.1 i kapittel 9.

Applikasjonsfeil, maskinvarefeil og kofigurasjonsfeil er alle hendelser som skaper problem når man skal teste i et testmiljø. Dette nevnes av flere bedrifter, men eventuelle tiltak varierer. Felles for tiltakene er likevel at det generelt finnes tekniske løsninger for hvordan disse problemene skal unngåes. For å redusere problemet med maskinvarefeil kan man for eksempel innføre duplisering slik at man fort får systemet opp å kjøre etter feil. Dette må vurderes nøye da det kan være uforholdsmessig dyrt i forhold til konsekvensen. At infrastrukturen i testmiljøet er avansert gir også en risiko med tanke på at det tar tid å bygge testmiljøet, og at man derfor får problem med å levere det avtalte testmiljøet til avtalt tid. Det er derfor viktig for testerne å komme tidlig inn i prosjektet sånn at rett testmiljø bestilles tidlig. I tillegg er det aktuelt å dra inn personer fra infrastruktur inn i prosjektet.

15.4 Sammendrag

Dette kapittelet har gitt et sammendrag av hendelser med tilhørende tiltak, sortert på tre tema: Organisatoriske risikoer, tekniske risikoer og samarbeidsrisikoer. I kapittel 17 vil vi rangere hendelsene basert på utregnet leverage. Kapittelet danner også en del av grunnlaget for våre anbefalte retningslinjer i kapittel 18.

Del III

Analyse

Kapittel 16
Introduksjon til analyse

I denne delen vil vi analysere og diskutere resultatene vi er kommet fram til i Del II, sammenfatte dette i et kapittel med retningslinjer for opprettelse av testmiljø og tilslutt presentere en konklusjon og forslag til videre arbeid.

16.1 Delens oppbygging

Diskusjon Dette kapittelet diskuterer resultatene fra intervjuene og risikoanalysen. Vi diskuterer også hvor EDB står i forhold til andre bedrifter.

Retningslinjer for opprettelse av testmiljø Resultatene fra analysen og teoridelen sammenfattes i konkrete retningslinjer for bruk av testmiljø.

Konklusjon Dette kapittelet konkluderer og oppsummerer arbeidet vi har gjennomført.

Videre arbeid I dette kapittelet identifiserer vi videre arbeid innenfor området.

Kapittel 17
Diskusjon

I dette kapittelet diskuterer vi resultatene fra de gjennomførte intervjuene og risikoanalysene. Kapittelet vil sammen med teorien danne grunnlag for kapittel 18.

Vi diskuterer grunnlaget for arbeidet med testmiljø, om man trenger et testmiljø for hvert testnivå, hvor likt produksjon testmiljøet må være, utfordringer ved testmiljø og verktøy. Tilslutt diskuterer vi hvor EDB står i forhold til andre norske bedrifter.

17.1 Grunnlag for testmiljø

Det er et gjennomgangstema hos bedriftene, med unntak av Telenors 200-årsprosjekt, at de ikke har en egen nedskrevet testmiljømetodikk og at de kjenner til lite litteratur på området. Hvor bedriftene finner kunnskap om testmiljø og hvordan de behandler den vil bli diskutert i dette avsnittet.

17.1.1 Kilder til testmiljø

Samtalene viser at arbeidet med testmiljø i stor grad bygger på erfaringer internt i bedriften. Samtlige bedrifter nevner dette som en viktig faktor, og mener dette kan være et resultat av at det i stor grad ikke finnes informasjon om testmiljø i litteraturen, det være seg retningslinjer, beste-praksis eller metodikk. EDB og Domstoladministrasjonen, som har store driftsavdelinger, mener at spesielt erfaringer fra drift er viktig ved bygging at testmiljø. HEMIT rapporterer at erfaringer fra produksjonsnettet er nyttige. Abeo nevner at utviklernes kunnskap er viktig, spesielt på enhetstesting.

Det kan virke som at bedriftene i mangel på litteratur om testmiljø i stor grad benytter erfaringer fra produksjonsmiljø og driftsgruppen som drifter dette. Denne tilnærmingen til testmiljø virker som en fornuftig framgangsmåte ved mangel på litteratur, ettersom et testmiljø søker å etterligne et produksjonsmiljø og det derfor vil være store fordeler å forstå hvordan produksjonsmiljøet bygges og driftes. For å ivareta disse erfaringene er det viktig å få de dokumentert. Hvorvidt testavdelingen har et tett samarbeid med driftsavdelingen vil sannsynligvis kunne påvirke arbeidet med testmiljø. DA, hvor driftsavdelingen er lokalisert i Oslo, og testavdelingen i Trondheim, påpeker at distribuert plassering kan være en ulempe.

Selv om interne erfaringer synes å være hovedkilden til arbeid med testmiljø, nevnes også andre kilder. Deltakelse på kurs og konferanser, og tett samarbeid med leverandører nevnes av flere. Spesielt Telenors 2000-årsprosjekt hadde stor tilgang på slike ressurser, i tillegg til å leie inn eksterne personer. I ettertid har Ballangrud, som testleder for Telenors 2000-årsprosjekt, tro på at bruken av flere kilder var medvirkende til at prosjektet lykkes.

17.1.2 Nytten av offisiell, dokumentert metodikk

Det er kun EDB og Telenors 2000-årsprosjekt som har en dokumentert, offisiell dokumentasjon på metodikk for testmiljø. De øvrige bedriftene har retningslinjer for testmiljø, men disse eksisterer som erfaring blant ansatte, og dette er et problem ved nyansettelser. Både HEMIT og DA jobber med å løse dette, i form av henholdsvis metodikk for testmiljø og et kompetansesystem. Abeo skiller seg igjen ut ved å være et konsulentselskap, der kundens metodikk og ønsker, er det som legger føringer på arbeid med testmiljø.

Også EDB arbeider med å lage en testmiljømetodikk, da dagens metodikk er en del av testmetodikken og ikke dekker alle ønskelige områder for en testmiljømetodikk, som diskutert i delkapittel 14.1. Hvorvidt bedriftene som har nedskrevet metodikk for testmiljø arbeider bedre enn bedriftene uten en slik metodikk, er vanskelig å svare på. Å ha en nedskrevet testmiljømetodikk liggende i en skuff fører ikke automatiskt til at bedriften jobber bedre enn tidligere. Hvordan man skaper testmiljømetodikken og benytter den, er det som får innvirkning for bedriften. Det virker som at testmiljø, i EDB og Telenors 2000-årsprosjekt, ble anerkjent som et viktig område, at det satses mye på opplæring og ressurser, og at man i større grad har benyttet eksterne ressurser ved etableringen av metodikken. Vi tror at arbeidet med testmiljø vil nå et høyere kvalitetsnivå ved bruk av en dokumentert metodikk, da man ved arbeidet med å lage metodikken får gjennomgått avdelingens rutiner, og sikret seg at en offisiell framgangsmåte blir fulgt. Bedriftene har også et bevisst forhold til testmiljø, og det faktum at HEMIT og DA jobber med nedskriving av metode indikerer at dette blir sett på som viktig og nyttig for bedriften.

17.2 Testnivå og testmiljøtype

Fra et produkt planlegges til det går i produksjon, gjennomgår produktet tester på flere nivå, fra lavnivå enhetstest til høynivå akseptansetest. Testnivåene gjennomføres i et testmiljø, og det er interessant å diskutere hvor mange testmiljø man trenger. Må man ha et testmiljø for hvert testnivå, eller kan man bruke det samme testmiljøet til flere nivå? Hvor mange testmiljø trenger man i såfall?

17.2.1 Flere testnivå

Alle bedriftene vi har vært i kontakt med, skiller mellom flere testnivå. Hva de selv gjennomfører varierer, ettersom noen utvikler selv, noen mottar kode fra leverandør, mens andre selv er leverandør. Dette får innvirkning på hvilke testnivå de gjennomfører og dermed også på hvilke krav de stiller til testmiljøet. I Abeo gjennomfører utviklerne enhetstester, mens HEMIT og DA vil motta koden fra leverandør og derfor ikke ser behovet for å gjennomføre enhetstester. EDB kommer her i en særstillig da de er den eneste bedriften av de vi har intervjuet som følger hele løpet med utvikling, test og drift av system.

Produktene bedriftene jobber med gjennomgår typisk enhetstest, integrasjonstest, systemtest og akseptansetest, og dette er også de testene som er vanlig i testlitteraturen, se delkapittel 5.4. Selv om bedriften benytter disse hovedtestnivåene, er det forskjell fra bedrift til bedrift og mellom ulike prosjekt i samme bedrift.

17.2.2 Forskjellige testnivå i forskjellige testmiljø

Tilbakemeldingen fra bedriftene viser at det er vanlig å ha mer enn et testmiljø, men hvilke testnivå som gjennomføres i hvilket testmiljø, er forskjellig ikke bare fra bedrift til bedrift, men også mellom de enkelte prosjektene i en bedrift. Det er derfor vanskelig å å gi et klart svar på hvilke miljø man trenger til hvilket testnivå. Vanligvis gjennomføres enhetstesting i utviklingsmiljøet, mens akseptansetesting vanligvis gjennomføres i et akseptansetestmiljø. Dette samsvarer med de undergruppene av testmiljø vi beskrev i delkapittel 6.2. Utviklingsmiljøet har de egenskapene vi finner i det teoretiske laboratorietestmiljøet. At enhetstesting blir gjennomført i utviklingsmiljø er også noe for eksempel van Veenendaal anbefaler, som nevnt i delkapittel 7.4 (van Veenendaal 2002).

Dette er ikke tilfelle med integrasjonstest og systemtest, der testmiljøet varierer avhengig av behov. Hos HEMIT er integrasjonstest en test for å integrere nye system inn i produksjonsnettverket, og man er avhengig av et testmiljø som er svært produksjonslikt. En integrasjonstest hos EDB kjøres når man integrerer enhetstestet kode inn i prosjektet, og det kan derfor være en del av utviklermiljøet. Pol et al. skiller i delkapittel 6.2 ut et eget systemtestmiljø (Pol et al. 2002). Et eget systemtestmiljø var det bare noen av bedriftene som hadde. Telenors 2000-årsprosjekt hadde som krav at systemtesting var gjennomført før man kom til akseptansetestmiljø, mens i EDB varierer dette fra prosjekt til prosjekt, avhengig av størrelse og hvor i verdikjeden man befinner seg. I noen tilfeller kan systemtestmiljøet være en del av utviklermiljøet eller videreføres til å bli et akseptansetestmiljø i enkelte prosjekt. På bakgrunn av dette er det vanskelig å bedømme hvor mange testmiljø som er nødvendig, men det er fordeler med å kjøre testnivåene i forskjellige testmiljø.

Grunnene til at man velger å kjøre testnivåene i forskjellige testmiljø er flere. Hvilke feil man prøver å avdekke varierer med testnivået, og man vil finne

flere feil i de tidligere testene enn man gjør i de senere. Enhetstester skal avdekke feil i koden hos utvikleren og utviklerne tester sin del av koden uavhengig av hva de andre utviklerne gjør. Man blir dermed avhengig av at enhetstestene gjennomføres samtidig og det er vanlig å benytte testmiljøet som finnes i utviklermiljøet. Komponenter som ikke er tilgjengelig under testing, kan om nødvendig simuleres. Gjennom akseptansetesten ønsker man å sjekke hele systemet, og det testmiljøet som ble brukt i enhetstest vil ikke kunne dekke en akseptansetest. I en akseptansetest er man blant annet avhengig av å kontrollere at systemet er stabilt, og det er gjerne aktuelt å la systemet være i drift over en lengre periode. Som nevnt i avsnitt 6.2.3 kan man også ha flere akseptansetestmiljø, og det er da funksjonell akseptansetest og produksjonsbasert akseptansetest. Abeo nevner også skille mellom disse typene akseptansetest, da det varierer på grunnlag av prosjekt, hva de deltar i. Om prosjektene bruker begge varierer. Hans Erik Ballangrud (IBM) kommenterer at man bør ha et så lite testmiljø som mulig for å dekke testen, da testing generelt er dyrt. Det er dyrt å bruke tid, personell, maskin- og programvare til å bygge og bruke flere testmiljø. Det oppstår altså en smerteterskel der flere testmiljø er uforholdsmessig dyre i forhold til det resultatet man kan oppnå. Man bør derfor vurdere forholdet mellom kostnad og nytteverdi når man velger antall testmiljø.

17.3 Nærhet til produksjon

Kilder i litteraturen mener at et testmiljø må være produksjonslikt, og at tester må kjøres under så virkelighetsnære forhold som mulig (van Veenendaal 2002, Pol et al. 2002). Dette standpunktet begrunnes med at testmiljø må være representative for produksjonsmiljø for å muliggjøre realistiske tester. Det presiseres at man ved produksjonslikt testmiljø først og fremst tenker på høynivås testmiljø, spesielt akseptansetestmiljø. Kildene definerer ikke hva som ligger i begrepet produksjonslikt, og hvor lik produksjon man må bygge for å muliggjøre realistisk testing. Ettersom produksjonslikt ikke er definert, går vi ut fra det som er vanlig framgangsmåte i de bedriftene vi har intervjuet.

17.3.1 Er det mulig å bygge testmiljø likt produksjon

Bedriftene er enige om at dersom et system skal testes ordentlig før det går i produksjon, må testmiljøet være produksjonslikt. Hvorvidt det er mulig å få et akseptansetestmiljø likt et produksjonsmiljø, er bedriftene litt uenige om. EDB og DA mener at dette er mulig, men samtidig svært kostbart. Et testmiljø skal være billigere å teste på enn et produksjonsmiljø, og det er derfor for kostbart å ha et 100% produksjonslikt akseptansetestmiljø. For at dette skal være mulig, må man bygge to kopier av hvert system, et for test og et for produksjon. Dette vil bli svært kostbart. I tillegg er det vanskelig å få antall samtidige brukere og last på systemet likt det man har i produksjon, da man

er avhengig av å simulere dette ved hjelp av lastverktøy. Abeo er enig i at absolutt produksjonslikhet er vanskelig, da enkelte systemer er for komplekse i produksjon til at man kan bygge likt i akseptansetestmiljø, for eksempel bankløsninger. Ingen av bedriftene bruker derfor et fullstendig produksjonslikt system.

17.3.2 Likhet for programvare

EDB legger vekt på å ha den samme programvaren for test som for produksjon, og dette er mulig gjennom gode avtaler med leverandører. HEMIT legger vekt på at det funksjonelle innholdet skal være likt. DA samarbeider med andre bedrifters testmiljø der deler av produksjonsmiljø ligger utenfor DAs systemer, som for eksempel Brønnøysundregistrene. Telenors 2000-årsprosjekt gjennomførte repeterte tester der den programvaren de aktuelle delene av systemet skulle kjøre på ikke var tilgjengelig ved tidspunkt tester ble gjennomført. Bedriftene enes altså om at programvaren bør være den samme som i produksjon.

17.3.3 Likhet for maskinvare

Maskinvare er en kostbar del av produksjonmiljøet, og bedriftene nedskalerer derfor dette i testmiljøet. HEMIT kjører tester på 32-bits maskiner, mens det i produksjon benyttes 64-bits maskiner. EDB benytter nedskalert programvare fra samme produsent som i produksjon, for på denne måten å kunne estimere ytelse ved hjelp av konverteringstabeller. En måte å utnytte maskinvaren effektivt på er å bruke virtualiseringsverktøy. Samtlige bedrifter med unntak av Abeo, der virtualisering ikke var et tema under intervjuet, rapporterer at de bruker eller ønsker å bruke virtualiseringsverktøy.

17.3.4 Bruk av virtualiseringsverktøy

Domstoladministrasjonen ønsker å framstille sitt relativt kompliserte produksjonsmiljø med et stort antall servere ved hjelp av fire-fem servere og virtualisering med Microsoft Virtual Server. Også EDB benytter virtualisering på servere, og ønsker å benytte VMwares virtualiseringsløsninger i enda større grad enn i dag. HEMIT har gjennom et prosjekt for å lukke testmiljøet kjøpt inn servere som man vil virtualisere med VMware. Telenors 2000-årsprosjekt ville brukt virtualisering dersom prosjektet var blitt gjennomført i dag, men på den tiden var partisjonering den vanligste løsningen. Vi ser at virtualisering er en mye brukt løsning for å utnytte maskinvare og simulere produksjonsmiljø. Å bruke virtualisering er derfor en akseptert framgangsmåte ved bygging av testmiljø selv om dette skiller seg fra produksjon. Det hevdes i teorien at man ikke kan ikke kan kjøre ytelsestester ved hjelp av virtualisering (Schaefer 2005b), men dette gjøres hos EDB ved bruk av konverteringstabeller.

Resultater fra ytelsestester med virtualisert maskinvare er likevel ikke helt representative for produksjon.

17.3.5 Testdata

Bedriftene og det som er publisert om testdata er enige i at testdata må være så produksjonlike og fullstendige som mulig, og flere av bedriftene gjennomfører komplekse rutiner for å oppnå dette. Blant annet kopierer DA hver natt data fra produksjon til testmiljø, og kjører automatiserte skript for å vaske de sensitive dataene. Etiske aspekter rundt sensitive data gjelder for flere av bedriftene, blant annet EDB og HEMIT som behandler henholdsvis personopplysninger knyttet til bank og finans, og sensitive helsedata. Ved bruk av testdata fra produksjonsmiljø må man forsikre seg om at de er vasket og at data ikke kommer på avveie. EDB spesielt snakker om at testere må ha taushetsplikt.

Risikoanalysen definerer noen hendelser i forbindelse med testdata. En hendelse er at testdata kan være inkonsistente. Det viktigste tiltaket som blir identifisert er automatiserte skript for synkronisering av testdata. Et annet problem er å få tak i riktige testdata, ettersom testerne er avhengig av at noen leverer disse. Viktige tiltak for å få testdata tidsnok er å kontraktfeste testdataleveranser og å ta kontakt med testdataleverandør tidlig.

17.4 Utfordringer ved testmiljø

Mange utfordringer finnes i arbeidet med testmiljø. Noen av utfordringene ligger i å skape de ressursene man trenger for å skape et godt testmiljø. Dette omfatter å øke statusen til testerne, finansiere testingen og finne kvalifiserte testere. Det er også viktig å dokumentere det arbeidet som gjennomføres.

17.4.1 Status

Status til testing blir trukket fram som en viktig utfordring i testarbeidet, selv om man har sett en forbedring i dette de senere år. Det sees blant annet ved innføring av begrep som "kvalitetssikring" istedenfor testarbeid. Generelt har det versert en oppfatning om at de som plasseres i testavdelingen er de som ikke kan brukes til annet, i følge Ballangrud. Selv om dette er feil, sier det noe om statusen testarbeid har hatt. Når testmiljø bare er ett av områdene innenfor testing, kan dette være en årsak til at lite forskning er gjort på området. I de senere år er mye blitt publisert innen testing, både i bøker og artikler, men den samme tendensen ser man ikke innenfor testmiljø.

17.4.2 Finansiering

En annen utfordring i arbeidet med testmiljø, er finansiering. Det er ofte en utfordring å få finansiert den maskin- og programvaren testavdelingen ønsker å kjøpe inn. I risikoanalysen påpekes hendelser relatert til ressurser for innkjøp av testutstyr. Et eksempel er HEMITs testing på 32-bits maskiner framfor 64-bit i produksjon. Ettersom 32-bits maskiner er billigere, blir disse valgt, selv om dette skaper en forskjell mellom testmiljøet og produksjonsmiljøet.

En måte å forebygge denne utfordringen på kan være å gjøre som diskutert i delkapittel 6.3, nemlig å etablere behovet for et testmiljø tidlig (Fewster & Graham 1999). På denne måten kan man definere hvilke testmiljø man har behov for, og skape aksept fra utviklere og ledelse. Dette kan resultere i at det er lettere å få bevilgninger.

17.4.3 Mangel på kvalifisert arbeidskraft

Abeo påpeker at mens test er en egen utdanning i utlandet, for eksempel i USA, er dette ikke tilfelle i Norge. Testere har gjerne lite om test i utdanningen sin. Det etterlyses flere fag innenfor test, som kunne resultert i større tilgang på fagfolk. Slik markedet er nå med stor kamp om hodene i IT-bransjen, frykter man at testere blir enda vanskelige å finne. Domstoladministrasjonen har funnet flere fordeler ved å la brukerne av systemet stå for testing. Deres erfaringer er at saksbehandlerne gjerne setter pris på å delta i dette, og tilbakemeldingene de har fått fra saksbehandlerne går på at det er gøy med variasjon i forhold til den vanlige arbeidsdagen. Saksbehandlerne blir leid inn på kortvarige oppdrag, og får tillegg i lønnen. Selv om man likevel må ha fagpersonell til utarbeidelse av tester, kan dette være en mulighet i et presset arbeidsmarked.

17.4.4 Mangel på dokumentasjon

Intervjuene bekreftet at det eksisterer lite nedskrevet dokumentasjon på testmiljø, man baserer seg på erfaringer blant ansatte. Utfordringene blir opplæring av nyansatte, og dersom nøkkelpersoner forlater bedriften. I delkapittel 6.3 og delkapittel 7.4, har vi sett at det anbefales å ha oppdaterte prosedyrer for drift, sikkerhetskopier og skript, oppsett og installasjon og overføring av system fra et miljø til testmiljø. Dette er det enighet om blant bedriftene vi har intervjuet, og det jobbes med å få dette på plass. Det blir gjort mye positivt innenfor hver bedrift, men samarbeid om beste-praksis og lignende er mangelvare. Tiltak vi kan foreslå for å forbedre samarbeid og på denne måten forbedre arbeidet med testmiljø, er å få testmiljø inn som tema på konferanser, danne samarbeidsgrupper på tvers av bedrifter, dokumentere erfaringer i en tilgjengelig erfaringsdatabase, og bruke testforum på internett aktivt, for eksempel Dataforeningen som blir dratt fram som et forum med høy aktivitet.

17.5 Verktøy

Flere av bedriftene bruker virtualiseringsverktøy. To løsninger for virtualisering er nevnt, nemlig VMwares virtualiseringsløsninger og Microsoft Virtual Server. De to største leverandørene på virtualiseringsverktøy i følge kapittel 8 brukes altså i de involverte bedriftene. Flere grunner til valg av løsning blir nevnt, blant annet ytelse, markedsposisjon og renommé.

Flere av bedriftene rapporterer at de bruker automatiserte tester. Hvilket program man benytter til dette, har vi ikke lagt vekt på under samtalene. EDB i Trondheim benytter Quality Center som sitt hovedverktøy for laging og kjøring av tester, og registering av testresultat og oppfølgingsplaner. De vil også ta i bruk muligheten for å lagre skript i programmet. Vi fikk verktøyet demonstrert av en av avdelingens ansatte, og inntrykket vi sitter igjen med er at verktøyet fungerte tilfredsstillende, og ble sett på som funksjonelt.

På bakgrunn av intervjuene virket det som om EDB hadde gjort overveide valg for bruk av verktøy. En anbefaling om å ta i bruk et av de andre verktøyene som ble beskrevet i kapittel 8 synes derfor uhensiktsmessig, all den tid dagens løsning fungerer, og løsningen er en av markedslederne innen sitt område av testing. For eksempel virker EDBs ønske om å i økende grad ta i bruk VMware, velbegrunnet i undersøkelser og rapporter om virtualiseringsverktøy.

17.6 Risikoanalyse

Risikoanalysen vi gjennomførte avdekket mange hendelser som kan bli problemer i arbeidet med testmiljø. I kapittel 15 ble hendelsene med mulige tiltak oppsummert i forhold til tema, mens vi i dette avsnittet vil konsentrere oss om tiltakene som nådde høyest da vi regnet ut leverage-verdien i etterkant av samtalene. For forklaring av leverage, se kapittel 11 I tabell 17.1 gjengir vi de ti tiltakene med tilhørende hendelser som oppnådde høyest leverage. At tiltakene får høy verdi, indikerer at det er disse tiltakene som bør bli gjennomført først.

Vi ser at flere av tiltakene knytter seg til prosjektledelse ved at man skal ha bedre planlegging, komme tidlig inn i prosjektet, ha fokus og stille krav til organisasjonen. Flere av disse tiltakene ble også nevnt under intervjuene. Dette er tiltak der man ofte anser kostnaden som lav, da man ikke ser åpenbare kostnader ved å gjennomføre tiltakene, og de får derfor en høy leverage. Dette er også tilfelle med tiltaket om å prioritere test høyere. Selv om det i og for seg koster lite å bestemme seg for å prioritere test, er spørsmålet hvordan man skal gjennomføre tiltaket. Hva betyr det å prioritere test høyere? Betyr det at testleder får mer innflytelse i prosjektet? Betyr det flere ressurser til test? I så tilfelle vil høyere prioritering av test bli en betydelig kostnad. Vi vil likevel hevde at mye kan gjøres ledelsesmessig for å forbedre bruken av testmiljø, selv om vi setter spørsmålstegn ved den lave kostnaden som er satt på disse tiltakene.

Nr.	Leverage	Hendelse	Tiltak
1.	999	Programvarefeil, inkl. feil i 3. part	Programmere seg rundt problemet eller vente på og installere fiks
2.	999	Testtiden blir for kort fordi utviklerne ikke er ferdige	Bedre planlegging
3.	999	Samme som over	Høyere prioritering av test
4.	449	Konfigurasjonsfeil	Registrere endringer ved hjelp av automatisering
5.	399	Testmiljø ikke dimensjonert for aktuell testing	Involvere eksternt firma som utvikler tester
6.	399	Avansert infrastruktur	Tidlig inn i prosjektet, bestille testmiljø tidlig
7.	399	Testmiljøet er ikke som planlagt	Bedre prosedyrer og rutiner for etablering
8.	399	Manglende involvering med testfokus fra dag en	Ha fokus
9.	399	Samme som over	Krav til organisasjonen
10.	399	Opposisjon mot nye rutiner	Forklare, motivere, selge og forberede

Tabell 17.1: Risikoer med høyest leverage-verdi.

De fleste tiltakene i tabell 17.1 er tiltak som skal gjennomføres for å unngå at hendelsen skal skje, i forhold til tredelingen av tiltak vist i figur 9.1 i kapittel 9. Kun tiltak 1 og 4 er tiltak som ikke kommer inn under denne kategorien. Dette er også tilfelle i risikoanalysen, der det er en overvekt med unntakstiltak. Det kan være flere grunner til dette. For det første kan det være bedre for bedriftene å unngå en hendelse, enn å vente til den oppstår og så redusere konsekvensene av hendelsen. For det andre kan det være lettere å tenke på hvordan man kan unngå en hendelse, enn å unngå at hendelsen blir et problem.

Vi stiller også spørsmål om valget av frekvens framfor sannsynlighet i risikoanalysen kan ha hatt betydning for hvilken type tiltak som ble nevnt. Når man snakker om sannsynlighet er det vanlig å reflektere over ting som kommer til å skje i framtiden. Hvor sannsynlig er det at hendelsen vil inntreffe. Når man snakker om frekvens er det derimot vanlig å reflektere over ting som har skjedd i fortida. Hvor ofte skjedde hendelsen. Det viste seg under intervjuene at ved å velge frekvens ble det nettopp mye fokusering på fortida. Det ble snakket mye om ting som hadde gått galt og hvordan man burde prøve å unngå dette. I samtalene ble det automatisk til at dersom en hendelse inntraff så var det et problem, og å forhindre at hendelsen ble et problem selv om den inntraff var ikke en del av tematikken. Dette kan også være en av grunnene til at tiltakene som ble nevnt er vinklet mot å unngå hendelsen.

17.7 EDB i forhold til andre bedrifter

Oppdragsgiver for oppgaven har vært EDB, og de ønsker å vite hvor de står i forhold til andre bedrifter når det gjelder testmiljø. EDB skiller seg ut ved å være en svært omfattende bedrift med en egen, stor testavdeling. EDB er i tillegg den eneste bedriften som utvikler, tester og drifter. Resultene fra samtalene er derfor ikke direkte sammenlignbare, for eksempel vil man ha bruk for flere testnivå dersom man selv utvikler programvare.

Generelt er vår oppfatning at EDB har kommet langt i arbeidet med test og testmiljø. Selv om metodikken for testmiljø er en del av testmetodikken og ikke er så omfattende som man skulle ønske, mener vi likevel at den er et viktig bidrag til at EDB jobber strukturert med testmiljø. At bedriften har en egen testmiljørolle som er definert med arbeidsoppgaver i den nedskrevne metodikken er et eksempel på dette. Telenors 2000-årsprosjekt er det eneste prosjektet som har mye dokumentasjon på testmiljø, men det er ikke et mål å overdokumentere testmiljø, slik det hevdes at det ble gjort i dette prosjektet. At dette ikke er ønskelig, kom også til syne under samtalen med EDB i Trondheim, der man mente at en svakhet med TMM var at det ble for byråkratisk på høyere nivå.

17.8 Sammendrag

Dette kapittelet har diskutert resultatene fra samtalene med de involverte bedriftene. Vi har sett at arbeidet med testmiljø i bedriftene i stor grad er basert på interne erfaringer i bedriftene, og at disse i liten grad er nedskrevet. Kapittelet vil brukes som en del av grunnlaget for våre anbefalinger til retningslinjer for bruk av testmiljø i kapittel 18.

Kapittel 18
Retningslinjer for testmiljø

Dette kapittelet presenterer våre anbefalte retningslinjer for opprettelse av testmiljø beregnet til bruk i EDB. Kapittelet bygger på stoff fra teorikapittel 6 og kapittel 7, i tillegg til resultatene fra gjennomførte intervjuer og risikoanalyser.

Retningslinjene vi presenterer er et utgangspunkt for hvordan man bør bygge en metodikk for testmiljø og er ikke en oppskrift over hvordan alle arbeidsoppgaver rundt testmiljø skal gjennomføres. En metodikk for testmiljø bør være etablert på bedriftsnivå, mens en spesifikasjon for testmiljø bør etableres for hvert prosjekt. I retningslinjene tar vi opp tema det er viktig å få bevissthet rundt i arbeidet med testmiljø. Løsningene vil variere fra bedrift til bedrift og/eller prosjekt til prosjekt, da behovene er forskjellige. En bedrift som EDB, som har både utvikling, test og drift, vil ha behov for en mer omfattende metodikk enn for eksempel Domstolsadministrasjonen som ikke utvikler system selv. Begge selskapene vil likevel kunne nyttiggjøre seg av de etterfølgende retningslinjene.

18.1 Dokumentasjon

1. Det anbefales å ha oppdaterte prosedyrer for testmiljø:
   (a) Prosedyrer for drift, sikkerhetskopier, skript, oppsett og installasjon.
   (b) Prosedyrer for overføring fra et testmiljø til et annet testmiljø.
   (c) Prosedyrer for sikkerhetskopier og gjenoppretting.
   (d) Prosedyrer for endringsstyring og konfigurasjonsstyring. Testmiljø forandrer seg på grunn av maskinvareendringer eller endringer på testobjektet.
   (e) Prosedyrer for tilgjengelighet i testmiljø.
   (f) Prosedyrer for feil i testmiljø (katastrofeplan).
   (g) Prosedyrer for reservasjoner i testmiljøet.
   (h) Prosedyrer for tildeling av tid mellom testing og utvikling.
   (i) Prosedyrer for å stenge ned miljøet korrekt etter bruk til kjent tilstand, og fjerning av testfiler.
2. Spesifikasjon av testmiljøet bør skrives for hvert prosjekt.

(a) Bør skrives tidlig i prosjektet.

(b) Bør gjennomgås av prosjektledelse, tekniske eksperter og andre interessegrupper med hensyn på:

- Hvor korrekt spesifikasjonen er for produksjonsmiljøet.
- Hvor representativt spesifikasjonen er for produksjonsmiljøet.
- Hvor teknisk gjennomførbar spesifikasjonen er.
- Hvor økonomisk gjennomførbar spesifikasjonen er.
- Om testmiljøet kan leveres i tide.

(c) En god spesifikasjon gir bedre tid til å utvikle simulatorer, stubber og drivere som trengs i miljøet, og kan bidra til at man får aksept for testmiljøet tidlig fra utviklere og ledelse.

(d) Spesifikasjonens innhold kan følge mal i vedlegg E.

(e) Identifisere brukerdokumentasjon som bruksanvisninger, tekniske guider, installasjonsmanualer.

(f) Spesifikasjonen må være tydelig på hva prosjektet trenger, også med hensyn på personell, for å sikre nødvendige ressurser.

3. Bestillingsrutiner bør vurderes. Mange velger å bestille gjennom driftsavdelingen, noe som gir unødvendig lang ventetid. Ved å gi testere større frihet til å bestille selv kan man korte ned ventetiden.

18.2 Testmiljøleders ansvarsområder

Testmiljøleder er ansvarlig for:

1. Godkjenne endringer i testmiljøet, som endringer i maskin- eller programvare, testobjekter eller prosedyrer.
2. Dokumentasjon, som spesifikasjon av testmiljøet blir laget, og oppdatere denne.
3. Konfigurasjonsstyring av testmiljø. Registere hvem som gjør endringer. Ved gjengangere ved introduksjon av feil bør man innføre sjekklister eller automatisering.
4. Systemets oppetid.
5. Løse tekniske problem i miljøet.
6. Ansvarlig for at implementasjon av miljøet blir gjennomført.

18.3 Verktøy

1. Bruk funksjonelt verktøy. Hva som oppfattes som funksjonelt, varierer fra bedrift til bedrift. Noen bedrifter vil for eksempel ha behov for støtte

for automatiserte tester, andre trenger ikke denne funksjonaliteten. Det er derfor vanskelig å komme med mer enn en generell anbefaling for framgangsmåte:

   (a) Kartlegg hvilken funksjonalitet man behøver av verktøy.

   (b) Undersøk hva som finnes av tilgjengelig verktøy på markedet.

   (c) Velg verktøy på bakgrunn av behov og marked.

2. Avgjøre om man ønsker å benytte virtualiseringsløsninger. Dette kan blant annet anbefales for å utnytte maskinvare bedre.

18.4 Gjenoppretting av testmiljø

1. Avgjøre om testmiljøet kan blir kopiert og gjenopprettet.
2. Avgjøre hvor raskt gjenopprettingen kan skje.
3. Avgjøre hvor presist gjenopprettingen kan bli.
4. Avgjøre hvor viktig kontinuitet er, og eventuelt arrangere alternativer for testmiljøet, for eksempel duplikatservere.
5. Dokumentere hvorfor testmiljø gikk ned, slik at ansvaret blir plassert på rett sted, og man kan forbedre dette til neste gang. Testerne bør for eksempel ikke få skylden for nedetid som skyldes eksterne forhold som nettverksfeil hos leverandør.

18.5 Produksjonslikhet

For høyere testnivå må testmiljøet være så likt virkeligheten som mulig. Dette vil si:

1. Programvare må være likt, eller så likt som overhode mulig.
2. Maskinvare, som minne og prosessor kan være nedskalert. Bruk fortrinnsvis nedskalert maskinvare fra samme produsent. Man kan bruke virtualiseringsløsninger for å untytte maskinvaren. Gjennomfør analyser for å kartlegge hvilken innvirkning dette har på ytelsestester.
3. Testdata bør være identisk med det man forventer i produksjon. Sensitive data vaskes ved hjelp av automatiserte skript, og testere undertegner taushetserklæring. Behov for testdata må avklares så tidlig som mulig i prosjektet, slik at man forsikrer seg om at de er tilgjengelige ved behov. Ved ekstern kunde, kontraktfest dette med fastlagte konsekvenser.

Et testmiljø trenger altså ikke å være fullstendig likt et produksjonsmiljø, da dette er dyrt. All funksjonalitet som finnes i produksjon må imidlertid dekkes for å sikre at alle tester kan gjennomføres.

18.6 Dynamikk i testmiljøet

1. Det må være mulig å endre testmiljøet etter behov.
2. Testmiljøet bør inneholde fasiliteter for å endre dato.
3. Testmiljøet må tilpasses til å kunne teste aktuelle teknologier.
4. Antall testnivå og tilhørende testmiljø vurderes for hvert prosjekt. Man må vurdere om seperate miljø skal bygges, eller om de skal være en videreføring av hverandre. Enhetstestmiljøet bør være separat, gjerne som en del av utviklermiljøet. Dette ettersom enhetstestmiljøet har mye nedetid og hyppige endringer i kode. Akseptansemiljøet må være så produksjonslikt som mulig.

18.7 Testing og kvalitetssikring av testmiljøet

Når testmiljøet er bygget og installert, må man teste om det fungerer som planlagt. Man ønsker å minimere risikoen for at hendelser relatert til feil med testmiljøet skal oppstå. Dette kan gjøres ved å kjøre tester på testmiljøet.

1. Manuell sjekk av testmiljøet mot dokumentert konfigurasjonsbehov - bekrefter at miljøet er riktig, men garanterer ikke at det vil fungere.
2. Automatiske tester som bekrefter at miljøet er tilgjengelig.
3. Måle kvaliteten etter at testmiljøet er bygget ved å:
   (a) Avgjøre hvor mye nedetid miljøet har, målt i gjennomsnittlig nedetid og prosentvis tilgjengelighet i henhold til spesifikasjon
   (b) Avgjøre hvor mange feil og defekter som skyldes problemer med testmiljøet.
   (c) Avgjøre hvor mye arbeid som kreves for å sette opp, gjenopprette og arbeide på miljøet.
   (d) Måle antall testmiljøreservasjoner som er i konflikt med hverandre
4. Overvåke driften av testmiljøet.
5. Anslå hvor mye ressurser testverktøy bruker.
6. En sammendragsrapport bør publiseres til interessegrupper, som ressursstyring, testledelse og prosjektledelse.
7. En egen kvalitetsgruppe skal granske og evaluere testmiljøet og rapportere til ressurstyring, testledelse og prosjektledelse. Et minimumsnivå er å verifisere at testmiljøspesifiksjoner blir skrevet tidlig i prosjektet i henhold til prosedyrer, at testmiljøet er virkelighetsnært, at tilgjengeligheten er akseptabel og at prosedyrer for styring og kontroll blir fulgt opp.

Kapittel 19
Konklusjon

Denne oppgaven har tatt for seg testmiljø, og er gjennomført i samarbeid med EDBs testsenter i Trondheim. Vi har utredet hvilke metodikker for testmiljø som finnes på markedet, og gjennomført samtaler med fire bedrifter, i tillegg til EDB, for å kartlegge hvilket forhold norske bedrifter har til testmiljø og hvordan testmiljøene bygges. Vi har også kartlagt mulige risikoer i arbeidet med testmiljø. Tilsammen har utredningen og de gjennomførte samtalene resultert i et sett med retningslinjer beregnet for bruk i arbeidet med testmiljø i EDB.

Utredningen av metodikker for testmiljø viser at det finnes lite dokumentert på området. Vi har ikke funnet noen komplette, beste-praksis metodikker for testmiljø, men har funnet fire publiserte modeller. Av disse er Testing Maturity Model (TMM) best dokumentert, og det mest aktuelle rammeverket å ta utgangspunkt i ved utarbeidelsen av en metodikk for testmiljø. For teknisk spesifikasjon av testmiljøet anbefales Hans Schaefers mal for testmiljø.

Ettersom vi i stor grad ikke fant litteratur om testmiljø, holdt vi samtaler med fire bedrifter og EDBs testavdelinger i Trondheim og Oslo. Fra samtalene vi gjennomførte konkluderer vi med at arbeidet med testmiljø i stor grad bygger på erfaringer internt i bedriften, spesielt erfaringer fra drift. Kun to av fem intervjuede bedrifter hadde en offisiell, dokumentert metodikk for testmiljø, og denne var i EDB ansett som mangelfull ettersom den var en del av bedriftens testmetodikk, og ikke sa noe om hvordan testmiljø skal bygges og brukes. Samtlige bedrifter, med unntak av Abeo som anser dette som vanskelig for et konsulentselskap, jobber med å utarbeide en dokumentert metodikk for testmiljø. En av grunnene til at bedriftene i dag ikke har en dokumentert metodikk, menes å være at det finnes lite om testmiljø i litteraturen.

Samtlige bedrifter mener at det er viktig at et akseptansetestmiljø er så nært som mulig et produksjonsmiljø innenfor en kostnadsmessig forsvarlig ramme. Produksjonslikhet er viktig for å kunne kjøre realistiske tester. Bedriftene vi har samtalet med, bruker programvare som er lik produksjonsmiljøet sin programvare. Bedriftene er også enige om at testdata må være så produksjonslike og fullstendige som mulig, og flere løser dette ved å kopiere testdata fra produksjon, og vaske sensitive data ved hjelp av automatiske skript. Når det gjelder maskinvare er bedriftene enige om at denne kan nedskaleres, eller man kan benytte virtualisering for å utnytte maskinvaren bedre. Alle bedriftene utenom Abeo sier de ønsker å benytte eller benytter virtualisering. Med unntak av Telenors 2000-årsprosjekt kjører ingen av bedriftene akseptansetester i et

miljø som er fullstendig likt produksjonsmiljøet.

Alle involverte bedrifter benyttet flere testnivå. Hvilke som benyttes varierer med bedriftenes kjerneområde, da ikke alle har behov for selv å gjennomføre enhetstester og akseptansetester. Hvorvidt testnivåene ble kjørt i samme testmiljø, varierte. Enhetstester gjennomførte samtlige bedrifter der dette var aktuelt i utviklermiljøet. Få av bedriftene hadde et eget systemtestmiljø, mer vanlig var det å gjennomføre dette testnivået i enhetstestmiljøet, eller å videreføre systemtestmiljø til akseptansetestmiljø.

Retningslinjene vi har skrevet anbefaler å ha oppdaterte prosedyrer for testmiljø og skrive en spesifikasjon av testmiljøet for hvert prosjekt. Spesifikasjonen bør gjennomgås av blant annet prosjektledelse og tekniske eksperter. Det anbefales å ha en testmiljøansvarlig, hvis arbeidsoppgaver inkluderer dokumentasjon, konfigurasjonsendring og å være ansvarlig for implementasjon. Bruk av et funksjonelt testverktøy, og gjerne bruk av virtualiseringsløsninger anbefales også. Videre anbefales det å avgjøre om det er mulig å gjenopprette testmiljøet dersom det går ned, og foreta tiltak i forhold til dette. Programvare og testdata må være så produksjonslikt som mulige, mens maskinvare kan være nedskalert, fortrinnsvis fra samme produsent som i produksjon. Man bør kunne endre testmiljøet ved behov, og vurdere antall testmiljø i forhold til testnivå for hvert prosjekt. Etter man har bygget testmiljøet, anbefales det å teste og kvalitetssikre miljøet.

På bakgrunn av de gjennomførte samtalene, konkluderer vi med at EDB har lagt ned mye arbeid i testmiljø i forhold til andre bedrifter. Selv om Telenors 2000-årsprosjekt har et svært gjennomført og dokumentert testmiljø var dette et konkret prosjekt med rikelig ressurser. Selv om EDBs testmiljømetodikk er en del av testmetodikken, inneholder den mye dokumentert informasjon i forhold til andre bedrifter. EDB bruker elementer i de ulike TMM nivåene som mål for sitt forebedringsarbeid, og dette betyr at bedriften arbeider mye riktig med testmiljø.

Oppgaven konkluderer altså med at testmiljø i større grad enn i dag bør dokumenteres, og at bedrifter bør arbeide etter en grundig dokumentert, offisiell testmiljømetodikk.

Kapittel 20
Videre arbeid

Ettersom arbeidet med denne oppgaven begrenset seg til fem måneder, har det underveis i arbeidet dukket opp interessante spørsmål og tema vi ikke har rukket å utforske, selv om vi tror dette kan være interessant å se nærmere på.

Første tema er å utvide samtalene rundt testmiljø til å involvere flere bedrifter. Underveis i samtalene fikk vi flere forslag til andre bedrifter vi kunne kontakte. Vital og Sun ble nevnt som aktuelle bedrifter av flere intervjuobjekter, og vi tror at de, i tillegg til eventuelle andre, kan være aktuelle å ta kontakt med for å ytterligere kartlegge hvordan norske bedrifter står med hensyn på testmiljø.

Underveis med arbeidet har vi forstått at det arbeides mye med testmiljø i norske bedrifter, men at erfaringer i mindre grad utveksles på tvers av bedrifter. Vi tror det kunne vært interessant å samle representanter fra flere bedrifter til en felles samling, der man i fellesskap kan komme fram til beste-praksis for arbeid med testmiljø, og på grunnlag av dette utarbeide et sett med retningslinjer. Dette kunne fungert som et fagforum, og vi tror det ville bidratt til å løfte arbeidet med testmiljø til et høyere nivå.

Andre tema er at det finnes få metodikker om testmiljø på markedet, og flere av intervjuobjektene sier at de har tatt utgangspunkt i rutiner for drift eller produksjon. På grunnlag av dette kan det være nyttig å undersøke om det finnes metodikker for drift eller produksjon, og ta utgangspunkt i dette for å utvide våre retningslinjer ytterligere. Vi tror også at det kan være en innfallsvinkel i den enkelte bedrift ved laging av testmiljømetodikk, men dette trenger nærmere undersøkelser før det konkluderes med noe.

Våre retningslinjer er utarbeidet på et teoretisk grunnlag. Et tredje tema kan derfor være å teste retningslinjene i praksis ved å designe og sette opp et testmiljø basert på våre foreslåtte retningslinjer for testmiljø. Her kan man også installere prøveversjoner av de foreslåtte testverktøy i kapittel 8, og på denne måten komme med ytterligere anbefalinger for valg av testverktøy.

Bibliografi

Abeo (2007). Om Abeo.
www.abeo.no [18.05.2007].

Battle, S. (2003). Boxes: black, white, grey and glass box views of web-services.
http://www.hpl.hp.com/techreports/2003/HPL-2003-30.pdf
[19.02.2007].

Black, R. (2002). *Managing the Testing Process - Practical Tools and Techniques for Managing Hardware and Software Testing*, second edn, John Wiley and Sons.

Borland (2007). Borland SilkTest.
http://www.borland.com/us/products/silk/index.html [16.04.2007].

Burnstein, I. (2003). *Practical Software Testing*, first edn, Spring Science+Business Media.

Caesar, R. E. (1997). Design methodology for creating an effective test environment, *IEEE Autotestcon Proceedings* (7): 129–142.

Certified Software Tester (2005). Certified Software Tester Body Of Knowledge.
http://www.softwarecertifications.com/cstebok/cstebok4.htm
[26.02.2007].

Certified Software Tester (2006a). Certified Software Tester Body Of Knowledge.
http://www.softwarecertifications.org/cstebok/cste6cbok2.htm
[27.02.2007].

Certified Software Tester (2006b). CSTE Body of Knowledge Knowledge Category 2.
http://www.softwarecertifications.org/cstebok/cste6cbok2.htm
[11.05.2007].

Copeland, L. (2003). *A Practitioner's Guide to Software Test Design*, first edn, Artech House, Incorporated.

Doar, M. B. (2005). *Practical Development Environments*, first edn, O'Reilly.

Domstoladministrasjonen (2007). Om domstoladministrasjonen.
www.domstol.no/domstoladministrasjonen [10.05.2007].

EDB Business Partner (2007a). EDB.
http://www.edb.com [22.02.2007].

EDB Business Partner (2007b). Årsrapport 2006 - vi leverer den nye friheten.
http://hugin.info/194/R/1121274/206388.pdf.

EDB Business Partner Norge AS (2006). Testmetodikk.

Eickelmann, N. S. & Richardson, D. J. (1996). An evaluation of software test environment architectures, *Proceedings of the 18th International Conference on Software Engineering* .

Fewster, M. & Graham, D. (1999). *Software Test Automation*, first edn, ACM Press Books.

Fowler, M. (2007). xUnit.
http://www.testapps.com/index.htm [17.04.2007].

Graham, D., Veenendaal, E. V., Evans, I. & Black, R. (2007). *Foundations of Software Testing*, first edn, Thomson Learning.

Helse Midt-Norge IT (2007). Om HEMIT.
www.hemit.no [14.05.2007].

Hewlett-Packard (2007). Mercury Quality Center.
http://www.mercury.com/us/products/quality-center/ [11.04.2007].

IBM Corporation (2007). Products by category - software quality management.
http://www-306.ibm.com/software/rational/sw-bycategory/subcategory/SW730.html [17.04.2007].

Infoworld (2005). VMware sends in the clones to ease testing.
http://www.infoworld.com/article/05/08/22/34TCvmware_1.html [17.04.2007].

Infoworld (2007). Desktop virtualization tools vie for position.
http://www.infoworld.com/Microsoft_Virtual_PC_2007/product_87038.html?view=1&curNodeId=0&index=1 [17.04.2007].

ITworld.com (2005). Mercury extends reach of application-testing tool.
http://www.itworld.com/AppDev/1262/050801mercuryqa/ [22.03.2007].

Jacobs, J., van Moll, J. & Stokes, T. (2000). The Process of Test Process Improvement, *XOOTIC Magazine* .

Kaner, C., Bach, J. & Pettichord, B. (2002). *Lessons learned in software testing*, first edn, Wiley Computer Publishing.

Koomen, T. & Pol, M. (1999). *Test Process Improvement*, first edn, Pearson Education Limited.

Kunnskapsforlaget (2007). Ordnett.no kunnskapsforlagets blå språk- og ordboktjeneste.
www.ordnett.no [20.05.2007].

Microsoft Corporation (2007a). Microsoft Virtual PC 2007. http://www.microsoft.com/windows/products/winfamily/virtualpc/overview.mspx [01.03.2007].

Microsoft Corporation (2007b). Microsoft Virtual Server 2005 R2. http://www.microsoft.com/windowsserversystem/virtualserver/default.mspx [01.03.2007].

Microsoft Corporation (2007c). Virtual PC vs. Virtual Server: Comparing Features and Uses. http://technet2.microsoft.com/WindowsServer/en/library/bcc5e200-88af-4a64-963b-55f1efb251d11033.mspx?mfr=true [01.03.2007].

PC Magazine (2005). VMware Workstation 5. http://www.pcmag.com/article2/0,1759,1785594,00.asp [17.04.2007].

PC Magazine (2007). Microsoft virtual pc 2007. http://www.pcmag.com/article2/0,1759,2099563,00.asp [17.04.2007].

PC PRO (2006). Vmware workstation 5.5. http://www.pcpro.co.uk/shopper/pcpro-reviews/83716/vmware-workstation-55.html [17.04.2007].

Perry, W. E. (2006). *Effective Methods for Software Testing*, third edn, Wiley Publishing, Inc.

Personal Computer World by Alan Stevens (2007). Review: XenSource XenServer for Windows. http://www.vnunet.com/personal-computer-world/software/2184564/review-xensource-xenserver [17.04.2007].

Pol, M., Teunissen, R. & van Veenendaal, E. (2002). *Software Testing*, first edn, Pearson Education Limited.

QAI Worldwide (2006). Qality assurance institute worldwide. http://www.qaiworldwide.org/qai.html [11.05.2007].

Robert Mullins for IDG News Service (2007). Xensource expands its virtualization capabilities. http://open.itworld.com/4915/070403xensource/page_1.html [14.04.2007].

Schaefer, H. (2005a). Specification of test environment.

Schaefer, H. (2005b). Test environment issues.

Schroeder, P. J. & Korel, B. (2000). Black-box test reduction using input-output analysis, *ACM* .

Shankland, S. (2006). The changing world of virtualisation. http://uk.builder.com/programming/unix/0,39026612,39305897,00.htm [01.03.2007].

Spillner, A., Linz, T. & Schaefer, H. (2006). *Software Testing Foundations*, first edn, dpunkt.verlag.

Stålhane, T. & Skramstad, T. (2006). Risk and opportunity. Powerpoint-presentasjon EuroSPI.

Testapps.com (2007). Testapps.com. http://www.testapps.com/index.htm [12.04.2007].

Tian, J. (2005). *Software quality engineering*, first edn, John Wiley & Sons, Inc.

Tretau, R. (2002). *WebSphere Application Server - Test Environment Guide*, second edn, IBM Redbooks.

van Veenendaal, E. (2002). Guidelines for Testing Maturity Part 2: Test Maturity Model level 2, *Professional Tester* **Volume three**(2).

van Veenendaal, E. (2006). Guidelines for Testing Maturity The Test Maturity Model. www.tmmifoundation.org/downloads/ resources/TestMaturity-Model.TMMi.pdf.

van Veenendaal, E. (ed.) (2005). Glossary Working Party - International Software Testing Qualification Board.

van Veenendaal, E. & Swinkels, R. (2002). Guidelines for testing maturity part 1: The ttm model, *Professional Tester* **Volume Three**(1).

VMware (2006). Product datasheet vmware lab manager virtual lab automation system. http://www.vmware.com/products/labmanager/ [11.04.2007].

VMware (2007a). Development and test product. http://www.vmware.com/products/developer_products.html [01.03.2007].

VMware (2007b). Powerful virtual machine software for the technical professional. http://www.vmware.com/products/ws/ [10.04.2007].

Vogel, P. A. (1993). An integrated general purpose automated test environment, *ACM SIGSOFT Software Engineering Notes, Proceedings of the 1993 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '93* **Volume 18**.

Wikipedia - The Free Encyclopedia (2007a). Junit. http://en.wikipedia.org/wiki/JUnit [17.04.2007].

Wikipedia - the Free Encyclopedia (2007b). Xen. http://en.wikipedia.org/wiki/Xen [12.04.2007].

XenSource (2007). Xenenterprise - multi-os virtualizatioin for enterprise it. http://www.xensource.com/products/xen_enterprise/ [11.04.2007].

Del IV

Vedlegg

Tillegg A
Spørsmål stilt til bedrifter

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?
2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?
3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?
4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?
5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?
6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?
7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?
8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhets-testmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?
9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?
10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?
11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

Tillegg B
Utskrift fra intervjuer

B.1 EDB Oslo

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

   Nei, firmaet har ikke en offisiell nedskrevet metodikk for bygging og bruk av testmiljø. Det arbeides derimot med å lage dette nå og målet er at denne skal bli ferdig i løpet av neste år. De som jobber med testmiljø har derimot en enkel håndbok, men denne blir ikke vektlagt i daglig drift, da de baserer seg mest på erfaringer blant ansatte.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

   Selskapet har en gjennomført testmetodikk som nevner testmiljø, men som ikke forteller noe om hvordan testmiljøene skal bygges og brukes. Den forteller derimot at det skal finnes testmiljø og at man skal ha en testmiljøansvarlig. Testmiljømetodikken definerer de testene firmaet bruker.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

   I dag er testmiljømetodikken ikke nedskrevet og det er så å si ingen føringer når det gjelder bygging og bruk av testmiljø. Gjennom testmetodikken er det noen organisatoriske føringer på hvordan testmiljøavdelingen skal være bygd opp og brukes av utviklerne og hvilke tester som skal gjennomføres i testmiljøet. For eksempel gir definasjonen av akseptansetest føringer for at akseptansetestmiljøet skal være produksjonslikt. Testmetodikken legger dermed føringer for arbeidsrutinene i testavdelingen.

   Det er spesielt tre ting de ser på ved utarbeidelsen av metodikk for testmiljø

   (a) Å oppnå metodelikhet. Med en testmiljømetodikk er tanken at det skal bli mer likt fra sted til sted. Hos EDB jobbes det med testmiljø i Oslo, Bergen og Trondheim, og det er i dag forskjeller på hvordan ting gjøres på de ulike stedene. Mye av grunnen til dette

er at arbeidet foregår på bakgrunn av erfaringer, og erfaringene er forskjellige fra sted til sted. Erfaringene er også forskjellige fra person til person, men det er her lettere å overføre denne kunnskapen til de andre man jobber sammen med fysisk enn fra sted til sted.

(b) Krav til de ulike testmiljøene. Det er visse standardkrav for hvordan testene skal gjennomføres i et testmiljø. Akseptansetestmiljøet skal f.eks. være produksjonslikt. En god testmiljømetodikk kan ivareta dette.

(c) Metode for hva man skal velge. Det er kanskje lite å tjene på en testmiljømetodikk som inneholder spesifiseringer på hvilken gratisvare man skal benytte på database, hvilken server man skal benytte og lignende, ettersom dette er veldig forskjellig fra situasjon til situasjon. Konsept som virtuelle maskiner og lignende er derimot mer viktig, da dette er konsept som gjerne vil brukes i flere situasjoner. De bør derfor omfattes av testmiljømetodikken. Et eksempel på dette kan være bruk av virtualisering, der det i dag benyttes virtualisering på servere men ikke på klientene og Windows-komponentene.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   Arbeidet med å lage en testmiljømetodikk baserer seg på erfaringer internt i bedriften. Grunnen til dette er at de ikke har funnet mye litteratur om dette temaet.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

   EDB mener at det er flere grunner til at dette er tilfelle, og fire grunner til at det ikke finnes anerkjente metodikker blir nevnt.

   (a) Det er lite utbredt blant bedrifter i Norge å ha egne store testmiljø-avdelinger.

   (b) Testere er stort sett teknikere, og teknikere liker generelt ikke å dokumentere.

   (c) Testmiljøet ligger svært nærme produksjonsmiljøet og man kan derfor bruke driftsmetodikken som testmiljømetodikk.

   (d) Å bruke testmiljø bevisst er relativt nytt. EDB startet med testmiljø i Trondheim i 1995, og testmiljømetodikk blir gjerne utviklet som en del av modningsprosessen til en bedrift.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

   Bedriften har en bestemt person som testmiljøansvarlig for hvert prosjekt. Dette gir prosjektlederen i prosjektet en person å forholde seg til i

forhold til testmiljøet. Testmiljøansvarlig kan bygge testmiljøet selv helt eller delvis, eller delegere deler av arbeidet til andre teknikere. En person er gjerne testmiljøansvarlig for flere prosjekt samtidig. I et prosjekt gjennomføres enhetstest, integrasjonstest, systemtest og akseptansetest, og testmiljøansvarlig har ansvaret for alle med unntak av enhetstestene som gjennomføres av utviklerne.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Bygging og vedlikehold av testmiljøene hos EDB i Oslo gjøres av 10 ansatte i testmiljøavdelingen. Kapittel 12 gir en grundigere skildring av hvordan EDBs testsenter er strukturert.

   I testmiljøavdelingen i Oslo er de organisert basert på systemer - kanal og kjerne. Kanal er en fellesbetegnelse på produkter som går på Intel-teknologi og Unix-type servere. Kjerne er en fellesbetegnelse på de systemene som kjører på MVS (IBM Stormaskin) De aller fleste testmiljø i Oslo består av både kanal- og kjerneprodukter. Det totale ansvaret for dette testmiljøet har testmiljøansvarlig.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhets-testmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   De tre testene som testmiljøansvarlig har ansvaret for, kjøres stort sett i det samme testmiljøet. Det vil si at integrasjons-, system- og akseptansetestmiljø er likt. Dette skjer av økonomiske grunner. Å teste i tre ulike testmiljø vil koste i både maskinvare og arbeidstimer for de ansatte, og denne kostnaden er ikke verdt det man får tilbake av ulike miljø. Noen ganger vil akseptansetesten kunne gjennomføres i produksjonsmiljø i stedet, og den vil da gå under navnet akseptanse-/pilottest.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Bedriften bruker Test Maturity Model. Mens flere av testerne har liten interesse av denne sertifiseringen, får den innvirkning på hverdagen deres gjennom mer dokumentasjon. Bruken av modellen er ønsket av ledelsen, og målet er at man ved å gjennomføre kostnadskrevende arbeid initielt vil kunne redusere ekstraarbeid i lengden. Bruken av modellen har resultert i større krav til dokumentasjon og innføring av flere nye prosedyrer.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    Testmiljøene er ikke oppe 100% av tiden, men testerne mener at testmiljøene er oppe når de skal. Det er en forskjell på hvor tilgjengelig kanal- og kjernesystemet oppfattes. Dersom et kjernesystem går ned/blir utilgjengelig påvirker dette nesten alle testsystemer og alle får vite at det er en feil". Går et kanalsystem ned kan det i bestefall bare påvikre ett testmiljø/et system og bare de får vite om det". EDB har i dag ingen

harde fakta som kan gi noe svar på hvilket system som oftest går ned, men dette holder de på legge opp målinger på. Per i dag er det en subjektiv holding om hvorvidt et testmiljø eller testsystem har mye nedetid eller ikke har det. Testerne føler at testmiljøet stort sett er oppe når det skal og at det kun blir kommentert de få gangene det er problem her.

I tillegg kan problemer hos underleverandører og andre deler av systemet resultere i testmiljøproblem. For å få en bedre oversikt over hvor ofte og hvorfor testmiljøene går ned, gjennomføres det en undersøkelse i bedriften på nettopp dette. Dette omfatter et system som registrerer nedetid i systemet. Undersøkelsen skal gi et overblikk over stabiliteten i testmiljøene, og på den måten kunne finne ut hva som svikter og hvordan man eventuelt kan forbedre systemet.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

    Testmiljøet de kjører i Oslo er svært likt produksjonsmiljøet. Grunnen til at de får til dette er at de samarbeider godt med drift og har gode avtaler med tredje parts-leverandører når det kommer til delsystem som brukes i testmiljøet. Dette gjelder for eksempel Lindorff og BBS der de i testmiljø for nettbank kan benytte tjenester på deres system. Dette vil da være en testversjon av Lindorf og BBS sine system, men testversjonen skal fungere på samme måte som produksjonsversjonen. Den største forskjellen mellom testmiljøet og driftsmiljøet er at CPU og RAM er nedskalert. Det hevdes likevel at opp mot 100% av RAM blir utnyttet. Grunnen til at enkelte deler er nedskalert er at testmiljøet skal være billigere å teste på enn produksjonsmiljøet, i tillegg til at bruken i test som oftest ikke tilsvarer produksjon når det gjelder belastning og antallet brukere.

    Når testmiljøet er så produksjonslikt som det er, er det en sjelden gang problem i overgangen fra utviklingsmiljøet, der enhetstestene kjøres, til testmiljøet der de andre testene kjøres. Dette tilsvarer ca. en feil pr 10 000 kodelinjer. En av grunnene til dette er at utviklingsplattformen stort sett inneholder gratisbasert programvare, mens det produksjonslike testmiljøet inneholder annen programvare. I denne overgangen i programvare, vil det oppstå en del feil der støttefunksjonene i systemet er ulike.

B.2 EDB Trondheim

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

    Egentlig ikke, det står litt om testmiljø i testmetodikken, men mer hva det skal omhandle enn hvordan det skal bygges. Internt er det

kommunisert hvordan holdningen til testmiljø er, nemlig at det bør være produksjonslikt. Med dette menes at det bør bygges ved hjelp av samme metoder som i produksjon.

Tradisjonelt har et testmiljø bestått av nedskalert maskinvare innenfor samme produsent som i produksjon. Dette gjør at det er mulig å sammenligne ytelse, ved hjelp av konverteringstabeller, og ved at man kan utnytte kompetansen til IT-drift. Komponenter til testmiljø bestilles på et bestillingsark til IT-drift, her spesifiserer også strømbruk, operativssystem, partisjoner med mer. Målet med et testmiljø er at det skal kunne løse 90% av problemene, og resten skal IT-drifteksperter ta seg av. Dette er i teorien, og i dag mener EDB Trondheim at de løser 98%, og de siste 2% er tunge å løse.

Det er viktig at testsenteret driver med applikasjonsdrift og ikke basis serverdrift. Man konsentrerer seg om minnelekkasjer, optimalisering av kode og lignende.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

   Som nevnt er den nedskrevne testmiljømetodikken er del av testmetodikken.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken for valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

   For maskinvare brukes personer på IT-drift som rådgivere, i tillegg brukes 3. parts leverandører. Det er sterke føringer på system- og akseptansetest, og også på valg av arkitektur, som har en levetid på 1-3 år. Man ønsker å bruke mer Linux, da dette operativsystemet er god på automatisk testing og virtualisering. Det er lite Linux i produksjon, men man ser for seg å få det inn i enhets- og integrasjonstest. Man ønsker å gå ned fra 3 testmiljø til 2, der det første er et freeware-plattformsmiljø og det andre er produksjonslikt.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   EDB har støttet seg på IT-drift og produksjonsmiljø ved utviklingen av metodikken. Man har nå en større grad av formalisme og etablerte rutiner. De ansatte kurses og sertifiseres, blant annet på Windows, AIX og Oracle. Interne fagfora er viktige, fordi eksterne kurs ikke gir så mye. Disse er ofte veldig basis i begynnelsen for testere med et høyt kunnskapsnivå, før man kanskje lærer litt mot slutten.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

Få har så stor testavdelingen som EDB, ved undersøkelse har man ikke funnet noen i Norden. Det er uvanlig at testavdelingen er så sentralisert. Andre jobber med noe "vi gjorde for 10 år siden", for å sette det litt på spissen. EDB har et høyt modenhetsnivå, men en del kunne vært dokumentert internt. Det er litt latskap at dette ikke er gjort. Noen ganger i året arrangeres testforumdag, her innprentes viktige ting og man sørger for en felles tankegang.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

   Ja, hvert prosjekt har en egen testmiljøansvarlig. En testmiljøkoordinator sikrer utnyttelsen av miljøet, ser behov for oppgraderinger og oppdager svakheter. Man har ukentlige møter med testmiljøansvarlig, utviklere og drift.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Testmiljøansvarlig er ansvarlig, men har ikke nødvendigvis det utførende ansvaret, Drift har ofte det utførende ansvar.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhetstestmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   Ja, man har ulike miljø. I praksis eget enhetstestmiljø for alle prosjekt, andre testnivå varierer fra produkt til produkt. Et alternativ for mindre produkt er å definere at for eksempel at "denne uken kjøres vi integrasjonstest, neste uke kjører vi systemtest' . Dette er også avhengig av hvor i verdikjeden produktet befinner seg. Er man for eksempel ytterst i verdikjeden er det ikke noe i veien for å teste integrasjon-, system- og akseptansetest på samme server, da produktet ikke brukes av andre. I hvor stor grad man har isolerte miljøer er altså avhengig av hvor i verdikjeden man er, men også kundens engasjement. Noen kunder vil ha et permanent testmiljø.

   Hvorfor er dette viktig? Man trenger et eget enhetstestmiljø fordi utviklere kan sjekke inn kode. Det kan være mye nedetid i et enhetstestmiljø, ergo trengs egne servere og miljøer. Integrasjonstestmiljø blir som oftest videreført til systemtestmiljø. I systemmiljø kan man legge inn nye versjoner av programmer automatisk hver dag, og derfor ha hyppige endringer. I et akseptansemiljø har man nye regler for hvor ofte endringer kan skje. Man trenger å simulere nært produksjonsmiljøets oppetid, stabilitet og pålitelighet, derfor legges nye versjoner gjerne inn hver 3. dag. Sikkerhet er også en faktor i akseptansetestmiljø.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Bedriften bruker TMM. Man undersøkte markedet, men endte på TMM. TMM sier en god del om testmiljø, men kan sies å være tungvindt og byråkratisk. Det kan derfor være greit å stoppe på et lavere nivå, fordi

man ikke blir effektiv nok dersom man skal sertifiseres på et høyere nivå. Et minimum av dokumentasjon er ofte mer hensiktsmessig. Vet ikke om TMM har hjulpet ved bygging av testmiljø, men modellen er muligens ansvarlig for en bedre rollefordeling internt.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    Ja, det er det man jobber for. EDB har innført et eget måleprogram som innrapporterer nedetid målt i timer og konsekvens. Man har også katastrofeplaner dersom noe går ned, i tillegg til å bruke back-up og SLA-planer.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

    Ja, dette er mulig, men samtidig svært kostbart. Man bruker i dag nedskalert maskinvare, som mindre CPU og minne, i tillegg til å ha flere applikasjoner på serveren samtidig. EDBs tilnærming er veloverveid, og er estimert til å ha svært liten risiko.

B.3 Domstoladministrasjonen

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

    Nei, Domstoladministrasjonen har ikke en offisiell metodikk. Det som finnes av kunnskap om testmiljømetodikk, befinner seg stort sett som erfaring hos ansatte i DA. DAs avdeling i Trondheim er relativt ung, og selv om noen av de ansatte fra Oslo har fortsatt, har det vært stor utskiftning av ansatte i løpet av flyttingen til Trondheim. Ståle Risem-Johansen startet selv i DA i juni 2005, og det var for eksempel 20 ansatte som sluttet i løpet av hans første to måneder i jobben. Han har tidligere jobbet i 15 år i flere private bedrifter og har i løpet av denne perioden fått et bredt erfaringsgrunnlag både som utvikler, og i senere tid, administrasjon og ledelse. Dersom du kommer som nyansatt er man derimot avhengig av å snakke med de andre ansatte for å lære om hvordan testmiljøet bygges og brukes. Dette ønsker man å forbedre ved at man er i ferd med å utvikle et kompetansesystem.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

    Nei, de har ingen verken nedskrevet testmiljømetodikk eller testmetodikk.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og

programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

Det som finnes av ikke-nedskrevet testmiljømetodikk stiller svært få føringer for bedriften. Det er derimot en del andre ting som legger mer eller mindre sterke føringer for hvordan man jobber med testmiljø hos DA. DA er for eksempel avhengig av å få bevilgninger av staten for å kunne gjennomføre store prosjekt. Dette gjelder også for testmiljø, der en stor bevilgning i september/oktober 2006 har gjort det mulig å investere i nye testmiljø. Mens det i dag kjøres et testmiljø hos DA, jobber man nå med å øke dette til tre testmiljø.

DA har også et veldig godt samarbeid med Microsoft, og dette gjør at man er generelt positive til hva Microsoft har å tilby. Selv om dette ikke er en sterk føring for testmiljøet, er det ikke tilfeldig at DA for eksempel har valgt Microsoft Virtual Server framfor VMware Workstation, når det kommer til programvare for virtualisering.

Bygging av testmiljø er basert på erfaringer fra drift.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   Utviklingen av testmiljø hos DA, er som tidligere nevnt ikke basert på en nedskrevet testmiljømetodikk. Når det så kommer til de erfaringene bedriften benytter som metodikk er denne bygget på bakgrunn av både teoretisk kunnskap, gjennom lesing, kurs, konferanser og lignende, leverandørinformasjon og praktisk erfaring både i DA og hos tidligere arbeidsgivere. Testforumet til Dataforeningen trekkes fram som et bra forum med mye aktivitet. Det er et generelt inntrykk av at det finnes veldig mye informasjon om testing men svært lite om testmiljø.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

   Intervjuobjektet har ingen forklaringer på dette.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

   Nei, bedriften har i dag ikke en bestemt ansvarlig for testmiljøet. Det er opp til hvert prosjekt å konkludere med om dette trengs, kanskje har driftgruppen anvar for det tekniske testmiljøet. Kvalitetsansvarlig mener at dette bør være en del av hans jobb i framtiden.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Kvalitetsansvarlig mener at han også har ansvaret for at testmiljøet bygges. Dette betyr ikke at han skal stå for selve byggingen, men at han skal ha et overordnet ansvar for at dette blir gjort. Testmiljøet driftes i dag av driftsgruppen i bedriften, og denne avdelingen ligger i dag i Oslo. Det er utfordringer med samarbeidet mellom Trondheim og Oslo, og det

skal evalueres om driftsavdelingen skal flyttes etter til Trondheim eller om de skal bli værende i hovedstaden. Det pågår også en evaluering om sourcing i bedriften, og dette kan også få betydning for driftsavdelingen.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhetstestmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   Ettersom bedriften ikke utvikler produktene sine selv, benytter de ikke alle de ulike testnivåene. DA kjører stort sett ytelsestester og akseptansetest. Disse testene kjøres begge i samme testmiljø, og DA ser ingen grunn til at dette skal være problematisk. Brukes virtualisering kan det være vanskelig å kjøre ytelsestester.

   I dag brukes ikke automatiserte tester, men dette er et satsingsområdet. Automatiserte tester vil ikke erstatte dagens tester, men ambisjonene er å frita testerne fra de kjedelige, gjentakende testene og gjenbruke testskript. Ønsketenkningen er 40% manuelt og 60% automatisert.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Bedriften bruker i dag ingen modell for testing. Kvalitetsansvarlig har lest flere ulike modeller, men har ikke funnet en bestemt testmodell som bør benyttes av bedriften. Han har derimot stor tro på Rational Unified Process (RUP), og mener at et fokus på prosessen også er viktig for testingen. Når det testes er det viktig at man tester hele prosesser og ikke bare delprosesser.

   Hvordan testene blir gjennomført har endret seg i løpet av den perioden DA har vært i Trondheim. Til å begynne med ble det gjort en del prøving og feiling, og man oppdaget at man får best effekt av testingen ved å involvere brukerne av systemet og fagstøtteapparatet. Dette har ført til at tester i dag gjennomføres ved at testspesifikasjonene lages i samarbeid med utviklingsleverandør og utføres av saksbehandlere som kommer til DA i et par uker.

   Dette er spesielt viktig ettersom de som tester vanligvis er saksbehandlere som er frigitt tid fra sin vanlige jobb for å teste systemet hos DA i et par uker. Disse testerne er gjerne svært positive til slike oppdrag ettersom de får et lønnstillegg for dette og får en avveksling fra hverdagen. Samtidig er ikke dette testere med stor teknisk kompetanse og de kan se svært liten nytte i å teste svært små spesifikke delsystem.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    I dag som det kjøres et testmiljø er ikke testmiljøet alltid tilgjengelig. Man tror derimot at ved å få tre testmiljø vil dette problemet bli praktisk talt ikke-eksisterende. Det skal være tre testmiljø, to testmiljø

til bruk for å teste fagapplikasjonen[1], et til drift og et til utvikling, og et testmiljø for de andre applikasjonene som skal testes. Man tror at DA med dette systemet vil kunne teste den det man trenger. Ettersom produksjonsmiljøet er relativt komplisert i dag, med et større antall servere, ønsker DA å framstille dette ved hjelp av fire/fem servere og virtualisering med Microsoft Virtual Server.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

    Dette bør være mulig. Målet med testmiljøet er at man skal teste ting før de kommer ut i produksjonsmiljøet, og da bør det være produksjonslikt. De delene av produksjonsmiljøet som ikke befinner seg hos bedriften bør dekkes ved hjelp samarbeid med andres testmiljø. Dette skjer ved at man benytter testmiljøene hos samarbeidspartnere.

    Testdata bør være likt som i produksjon. DA eksporterer testdata hver natt fra produksjon. Dette er sensitive data, og man er pålagt vasking som utføres ved hjelp av skript.

B.4 Helse Midt-Norge IT

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

    Nei, HEMIT har ikke en nedskrevet, offisiell metodikk for testmiljø. De har heller ikke en nedskrevet metodikk for testing generelt. Strategien til HEMIT er å satse på SOA. Målet er å kunne kjøre flere system sammen i en tjeneste-orientert arkitektur, og dette vil få innvirking for hvordan man bygger og drifter testmiljøet.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

    Nei, de har verken nedskrevet testmiljømetodikk eller testmetodikk, men det er derimot en plan om å få laget en metodikk. Denne vil bygges i samsvar med en nedskrevet standard som skal ivareta arkitekturstrategien.

    Standarden skiller mellom fire ulike miljø;

    (a) Utviklingsmiljø

    (b) Testmiljø

    (c) Produksjonstestmiljø

    (d) Produksjonsmiljø

[1] Fagapplikasjonen er et stort system med informasjon om rettsaker og tvister i norske domstoler.

Ettersom HEMIT ikke utvikler noe selv vil utviklingsmiljøet være hos den eksterne utvikleren.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

   Det som finnes av testmiljømetodikk bygger som sagt på SOA. Dette får innvirkning på valg av oppsett av testmiljø. For eksempel anser HEMIT det som viktig at man setter av et sett maskiner til test, som en del av arkitekturplattformen. De har nå opprettet et adskilt miljø der testmaskinene står i produksjonsnettet og benytter dermed det samme nettet som ved vanlig drift.

   Hvordan man bygger testmiljøet er avhengig av størrelsen på systemet. De fire større systemene benytter diverse testmaskiner som er spredt i produksjonsmiljøet. De mindre systemene har derimot varierende testmiljø avhengig av leverandøren. Mindre system er gjerne system som benyttes av enkeltavdelinger på sykehus eller andre små grupperinger. Dette er gjerne medisin-teknisk spesialutstyr og et eksempel vil være et system for å logge EKG[2] hos pasienter.

   I dag er det et prosjekt å få lukka nettverket med testmaskiner, gjennom både fysisk flytting og bruk av virtualisering. Det er kjøpt inn egne servere til testmiljøet, og HEMIT har valgt å satse på VMwares ESX. Microsoft sin løsning, Virtual Server, ble også vurdert, men HEMIT mener denne løsningen har for dårlig ytelse.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   Testmiljømetodikken bygger stort sett på kunnskapen blant de tilsatte. Disse har bygd opp testmiljø tidligere og sitter på mye kunnskap. I tillegg har SOA vært en kilde med tanke på at man velger å bygge testmiljøet adskilt, der man definerer tjenester i testnettet. Dette bygger også på erfaringen med produksjonsnettet, da man er avhengig av et testmiljø som ligner på produksjonsmiljøet.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

   At det ikke finnes en seperat metodikk for testmiljø vet ikke Myklebust, da han mener at testmiljø er viktig. Det er kanskje ikke interessant. Generelt baserer mye seg på kunnskap blant ansatte, og hovedidéen må være at testmiljøet skal være adskilt fra produksjon.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

[2]Elektrokardiogram (EKG) er registrering av hjertets elektriske aktivitet.

Ja, han er selv ansvarlig person for testmiljø. Dette er ikke en egen rolle da han også har andre oppgaver. De nye større systemene som kalles hovedsystem har gjerne egne testmiljøansvarlige, mens det i enkelte små delsystem gjerne skjer i samarbeid med systemansvarlig.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Hvem som har ansvaret for bygging av testmiljøet er avhengig av hvilket testmiljø det er. Hovedsystemene har en egen testmiljøansvarlig som har ansvaret for at systemet blir bygget, mens det for de mindre systemene ikke er like definert. Her bygges testmiljøene gjerne i samarbeid med systemansvarlig.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhetstestmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   Det er ulike testmiljø for de ulike testene, der noe testes hos leverandør mens andre tester gjennomføres i testmiljøet hos HEMIT. Ettersom HEMIT ikke utvikler programmer selv, blir enhetstest og systemtest gjennomført i utviklermiljøet hos leverandøren. Akseptansetesten og integrasjon mellom de ulike delsystemene utføres i testmiljøet hos HEMIT. Dette er viktig for å få testet hver enkeltdel i detalj, og sammen i produksjonsintegrasjon.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Nei, HEMIT benytter ikke TMM.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    Testmiljøet er tilgjengelig når det trengs. De har generelt en bra oppetid. Dette betyr ikke at det foregår testing i testmiljøet hele tiden. Testmiljøet er derimot tilgjengelig når enkelte prosjekt trenger det. Hvilke prosjekt som skal ha tilgang når blir koordinert. Et prosjekt har sjelden eksklusiv tid i testmiljøet. Men testmiljøet settes opp slik at det er mulig for flere prosjekt å teste samtidig. Dette gjøres ved at de ulike versjonene av testmiljøet får ulike tidsrom.

    Når det er nødvendig med nye testmiljø blir dette satt opp. Får å kunne gjennomføre dette prøver man å utnytte fordelene med virtuelle maskiner, og man samarbeider aktivt med driftsavdelinga. Automatiske tester blir ikke benyttet, ettersom testavdelingen mangler kompetanse når det gjelder å lage de skriptene man er avhengig av for å automatisere testene. Den koden som skal testes hos HEMIT skal i prinsippet være ferdigtestet, og man har dermed ikke det samme behovet for automatiserte tester. Det tar tid å produsere automatiserte tester, og vinninga går fort opp i spinninga, ettersom det er lite repetisjon når man tester.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

    Man vil aldri få et akseptansetestmiljø som er helt likt som produksjonsmiljøet når det gjelder volum. Når det gjelder oppsett av systemet og det funksjonelle innholdet i systemet, bør derimot akseptansetestmiljøet være likt. HEMIT prøver å isolere testmiljøet i et eget nettverk. Det er da avgjørende at de har kompetanse innen nettverksoppsett, brannmurer, distribuerte klienter og lignende. En forskjell mellom akseptansetestmiljøet og produksjonsmiljøet er at de tester på 32-bits maskiner mens produksjonen foregår på 64-bits maskiner.

B.5 IBM/Telenors 2000-års prosjekt

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

    Ja, alt som ble gjort under prosjektet ble grundig dokumentert på papir. Det eksisterer flere hyllemeter med dokumentasjon. Dette omfattet blant annet bakgrunn, prosedyrer og maler. Prosjektet ble nok overdokumentert.

    For å få folk til å dokumentere alt, brukte man tvang. Prosjektet hadde veldig stort fokus i Telenor og man var avhengig av å følge bestemmelsene fra ledelsen. Gjorde man ikke som ledelsen sa, kunne man få sparken.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

    Testmiljømetodikken var en del av den omfattende testmetodikken. Testmiljø var et av flere aspekt som var viktig i programmet for år 2000. Testsenteret som var ansvarlig for bygging og drift av testmiljø, ble etablert høsten 1998, og var ferdig i 1999. Her jobbet 54 ansatte med test, og man administrerte testsenter, testdata og testing av alle prosjektene her ifra. I tillegg til de ansatte var det også flere superbrukere og domeneeksperter inne i testsenteret. Som et resultat av feilene som ble funnet, ble kode omskrevet eller byttet ut.

    I ettertid kan man se at det ville vært ønskelig å lagt mer vekt på sporbarhet i testmiljøet, slik at det ville vært mulig å spore endringer lettere.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

Det var sterke føringer på at alt skulle være identiskt med produksjonsmiljøet. Et problem var at man startet i så god tid før år 2000, at programvaren som man skulle kjøre på ikke var tilgjengelig da koden skulle testes. I slike tilfeller hendte det at man måtte teste koden i flere omganger, der enkeltdeler av koden ble kjørt til ulike tidspunkt. Et annet problem kunne være at produksjonsmiljøet hadde en eldre versjon av program- eller maskinvare enn testmiljøet. I IBM produserer de stort sett hyllevare, men også her har Ballangrud sett at ulike versjoner kan skape problemer. Når man skifter for eksempel databasesystem fra Oracle 9.1 til 9.2, er det fort gjort å glemme dette, selv om det står som punkt i sjekkliste for produksjon. Dette vil trolig få innvirkning på både testmiljø og produksjonsmiljø.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   Når man startet opp dette prosjektet var det Ballangruds oppgave å få gjennomført testene i et velegnet testmiljø. Flere kilder til informasjon ble benyttet for at prosjektet skulle lykkes.

   Egen erfaring Etter å ha jobbet med test i flere år var det opparbeidet erfaring om testing som ble benyttet.

   Testkonferanser Deltok på testkonferanser over hele verden i to til tre år i forkant av prosjektet.

   Hans Schaefer Ble brukt spesielt til opplæring av ansatte i prosjektet.

   Vivi Horn

   Bøker om testing

   Andre norske ressurser Dette gjalt spesielt innenfor feltet kvalitetssikting/Quality Assurence (QA).

   Driftsgruppa i TeamCo Dette selskapet er i ettertid oppkjøpt av EDB, og driftet på denne tiden Telenors systemer. Mange i selskapet satt på stor kunnskap om produksjonsmiljøet hos Telenor og denne kunnskapen ble benyttet.

   Konsulenter fra Icon Medialab Disse ble spesielt brukt til prosjektledelse og prosedyredefinering, da dette er et område der de hadde god kompetanse.

   QA Labs QA Labs i samspill med Universitetet i Lund ble involvert, men det viste seg etterhvert at de var mindre aktuelle.

   Det var altså mange kilder som lå til grunn for testmiljømetodikken. Mye av grunnen til at det var mulig å benytte så mange kilder var at det var et prosjekt med godt med penger ettersom det var høyt prioritert.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

Mye av grunnen til at det ikke finnes metodikker for testmiljø er at dette er negativt arbeid. Utviklingsarbeid blir gjerne sett på som mer kreativt og det satses mye på innovativitet. Test assosieres derimot mer med en destruktiv handling. Testing har generelt lav status. Det kommenteres for eksempel at de som plasseres i testavdelingen er de som ikke kan brukes til noe annet. Test er derimot i ferd med å få mer status nå, og det er en dreining fra testbegrepet til kvalitetssikring. Det er nødvendig med domenekunnskap for å kunne teste.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

   Ja, hver av de under 100 verdikjedene ble enkeltprosjekt og hver av disse prosjektene hadde en ansvarlig for testmiljø.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Testmiljøet ble bygget og drevet av testerne som jobbet i testsenteret.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhetstestmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   Ja, det var ulike testmiljø i prosjektet. I testmiljøet kjørte man kun akseptansetest. Det var forutsatt at alle enkeltprosjektene skulle ha gjennomgått systemtest før de ble levert inn til testmiljøet. Man greier ikke å kjøre alle testene i akseptansetestmiljøet. Det er for eksempel for dyrt å kjøre enhetstest for store system i et testmiljø som er produksjonslikt. Man bør ha et så lite testmiljø som mulig for å dekke testen, da testing generelt er dyrt.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Prosjektet brukte TMM. Dette er en videreutvikling av Martin Pol sitt arbeid. Ballangrud mener at en slik vurdering koster penger, gjerne to til tre hundre tusen. Han har selv oversatt deler av Pol sin bok (Koomen & Pol 1999) til norsk, og holder gjerne halvdagsseminar om modellen.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    Det kunne bli forsinkelser av og til, men testmiljøet var stort sett tilgjengelig når det skulle. Grunnen til at testmiljøet var så tilgjengelig var at testmiljøet var en prioritert del av prosjektet hos konsernet. Det ble fokusert på testing. Dette handler stort sett om innstillingen i bedriften, og bør være mulig også i andre prosjekt.

    Man var også svært nøye med planlegging, sånn at både planlegging av testen og selve byggingen av testmiljøet var ferdig samtidig.

    For at testmiljøet skulle fungere som ønsket, ble det gjennomført tester av testmiljø. Dette skjedde gjennom inspeksjoner av testmiljø og at man

sendte transaksjoner gjennom nettverket for å forsikre seg om at ting hang sammen slik det skulle.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

    Nei, i praksis er det ikke mulig å få akseptansemiljøet helt likt produksjonsmiljøet. Det er mulig å gjøre det produksjonslikt gjennom simulering, men 100% produksjonslikt er ikke mulig. Man må ta noen snarveier. Med erfaring trenger man heller ikke et testmiljø som er helt likt produksjonsmiljøet, og man kan nedskalere mengde med tanke på for eksempel kompleksitet og antall brukere.

    Virtualisering ble ikke brukt under prosjektet, da det ennå ikke hadde blitt vanlig i bruk i denne tidsperioden. Det ble derimot brukt partisjonering. Virtualisering ville trolig blitt brukt om prosjektet skulle gjennomføres i dag, ettersom flere har veldig god erfaring med bruk av virtualisering.

B.6 Abeo

1. Har ditt firma en nedskrevet, offisiell metodikk for bygging og bruk av testmiljø?

    Selskapet har ikke en egen offisiell metodikk for bygging og bruk av testmiljø. Ettersom de jobber som konsulentselskap er mye avhengig av hva kunden vil og hvilket prosjekt det er. Det vil ofte skrives ned litt om testing i tilbudet man sender ut til kunden, men ulike kunder har ulike behov, og det er derfor vanskelig å følge en bestemt metodikk. Testmiljøene bygges ute hos kunde og Abeo som selskap har knapt et utviklingsmiljø. Abeo har derimot ansatte som sitter med kunnskap.

    Ute i prosjekt kan en ansatt bli satt til både utvikling og testing, eller kun testing avhengig av kunden og størrelsen på prosjektet. De kan delta i enhetstest, systemtest og akseptansetest, både Funksjonell akseptansetest (FAT) og Produksjonsbasert akseptansetest (SAT), eller jobbe med testlederaktiviteter.

2. Hvis ja, Hvordan relaterer denne seg til en eventuell testmetodkk, for eksempel, er den en del av testmetodikken?

    Abeo har heller ingen offisiell testmetodikk. De har derimot en utviklingsmodell der ulike testaktiviteter nevnes. Denne utviklingsmodellen bygger på Scrum og ShipIt. Om utviklingsmodellen blir brukt er avhengig av kunde og prosjekt, og Abeos konsulenter spenner derfor over det som finnes av metoder. Når det gjelder testing så lærer de ansatte etterhvert og Hagelund selv blir gjerne sett på bedriftens ekspert innenfor området.

3. Hvor omfattende er metodikken for testmiljø? Det vil si, hvor strenge føringer legger denne metodikken ved valg av for eksempel maskinvare og programvare, og rutiner, er den veldig detaljert eller på et mer overordnet plan?

   De har ingen metodikk som legger føringer. Det konkrete prosjekt og prosjektets innhold legger føringer på testmiljøet. Kunden kommer med en egen porteføljе og dette omfatter hvilke verktøy kunden har og hvilken metodikk som finnes hos kunden.

4. Hvilke kilder har dere støttet dere på under arbeidet med å lage en metodikk for testmiljø?

   Når Abeo bygger og bruker et testmiljø, bruker de flere kilder. Den viktigste kilden er kunden som sitter med egen metodikk og kunnskap. Dersom dette ikke er nedskrevet, kan det brukes tid på å dokumentere kunnskap hos kunden. Det er viktig å få et klart bilde av hva kunden vil.

   En annen kilde for gjennomføring av testingen er en mal fra IEEE som brukes for å sette opp testplan. Dette er det nærmeste de kommer en egen testmetodikk. I tillegg utnyttes utviklernes kunnskap, da de sitter på mye kunnskap, spesielt på enhetstestmiljø.

5. Hvorfor tror du ikke at det finnes anerkjente metodikker for testmiljø i litteraturen, siden det for eksempel finnes mange for utvikling, som smidige metoder?

   Hagelund setter spørsmålstegn ved at det ikke finnes anerkjente metodikker for testmiljø. Trenger man egen metodikk? Med et system som skal testes er det ofte nokså gitt hvordan miljøet må bygges. Det bør være så nært opp til produksjonsmiljøet som mulig, vurdert opp mot kostnadene for et slikt oppsett. I mange tilfeller er det svært kosntandskrevende, og man må vurdere om biter av totalsystemet må erstattes av dummies.

6. Har bedriften en egen ansvarlig person for testmiljø? Er dette en egen rolle, eller har personen andre roller? Er dette er fast person?

   Nei, konsulentselskapet har ingen egen rolle for testmiljøansvarlig. Innenfor et bestemt prosjekt kan derimot dette være en rolle. Store prosjekt kan ha råd til dette, mens mindre prosjekt eventuelt kan leie inn en konsulent som rådgiver for å se igjennom planer for test og lignende.

7. Hvem har ansvar for å bygge og vedlikeholde testmiljø?

   Hvert prosjekt er unikt, og dette gjelder også ansvar for bygging av testmiljø. Ofte involveres driftsavdelinga hos kunden, da de sjelden har råd til et eget testmiljø. Igjen er det kunden og prosjektet som avgjør.

8. Har bedriften forskjellige testmiljø for ulike testnivåer, som enhetstestmiljø, integrasjonstestmiljø, systemtestmiljø og akseptansetestmiljø? Hvis ja, hvor viktig tror du dette er? Hvis nei, hvorfor er ikke dette viktig?

   Ja, det benyttes forskjellige testmiljø for ulike testnivå. Spesielt bør akseptansetestmiljøet være galvanisk separert fra andre. Hvilke testmiljø

og tester som blir kjørt varierer fra prosjekt til prosjekt.

SAT kjøres vanligvis hos kunden, og testmiljøet er avhengig av hva kunden har tilgjengelig.

Enhetstestmiljøet etableres ofte på bakgrunn av hvilke verktøy som er tilgjengelige og hvilke rutiner som finnes for innsjekking av ny kode. Det går ofte fort å få tilbakemelding på feil.

Ideelt bør testmiljøene være ulike, med økt grad av samtidighet etterhvert som man kommer til høyere testnivå. I Clustra[3] var det for eksempel 10 til 15 CPU-er per tester og man hadde en stor testpark der man fordelte ressurser. En av fordelene var at det var lite bruk av Graphical User Interface (GUI) og det var derfor mulig med mye automatisering. Dette var dyrt og i vanlige prosjekt vil det derfor ikke være mulig. Det bør likevel være mulig å kunne kjøre system- og integrasjonstest på en del av systemet samtidig som utviklerne enhetstester. Det er viktig at feil blir funnet så tidlig som mulig.

Til kunden anbefaler man vanligvis ulike testmiljø, men dette må gjennom en budsjettprioritering.

9. Bruker bedriften en modell for testing, for eksempel Test Maturity Model (TMM)? Får denne modellen innvirkning på bedriftens testmiljø?

   Nei, bedriften bruker ikke en bestemt modell for testing. I de prosjektene de har anledning bruker de sin egen utviklingsmetodikk som også omfatter testing. Som nevnt i tidligere spørsmål bygger denne på Scrum og ShipIt og ingen egne testmodeller. Utviklingsmetodikken er skrellet for at det skal være enkelt å sette seg inn i den og bruke den. Den definerer noen testtyper og gjennomføring av ulike testaktiviteter står oppført på en liste over påkrevde aktiviteter i prosjektet. Prosjektleder har like mye innvirkning på testmiljøet som metodikken.

10. Er testmiljø tilgjengelig når det trengs? Hvis ja, hva tror du er grunnen til dette? Hvis nei, hvordan tror du dette kan forbedres?

    Ja, testmiljøet er tilgjengelig når det trengs i enkelte prosjekt. I andre prosjekt er det ikke det. Noen er flinkere på oppetid i testmiljø enn andre. Det er viktig at prosjektet forstår behovene i forbindelse med testing, noe de ikke alltid blir. Dette handler mye om prioritering, og det bør generelt være mer fokus på testing og betydningen det har for prosjektet i helhet.

    Testing bør også bli en større del av utdanningen i Norge. I andre land finnes det egne karriereveier innenfor testing.

11. Er det mulig å få et akseptansetestmiljø likt produksjonsmiljø (Miljøet systemet skal kjøre i)? Hvorfor er ikke dette eventuelt mulig? Tror du dette har betydning for resultatet, det vil si produktets kvalitet? Kunne produktet fått bedre kvalitet dersom testmiljøet var mer produksjonslikt?

[3]Clustra er et selskap etablert med røtter i NTNU, Sintef og Telenor som jobbet med høytilgjengelig database.

Spørsmålet er om det å finne flere feil vil føre til bedre kvalitet. Noen system kan få lavere kvalitet ved feilretting, og man må derfor være forsiktig med å definere kvalitet etter hvor mange feil som ble funnet.

Tillegg C
Resultater fra risikoanalyse

Som beskrevet i kapittel 11 har vi gjennomført risikoanalyse ved flere norske bedrifter. Risikoskjemaene som ble fylt ut i den anledning er samlet i dette kapittelet.

De bedriftene som har deltatt med risikoanalyser er:

- EDB Testavdelingen i Oslo, figur C.1.
- EDB Testavdelingen i Trondheim, figur C.2 og figur C.3.
- Domstoladministrasjonen (DA) i Trondheim figur C.4.
- Helse Midt-Norge IT (HEMIT) figur C.5.
- IBM/Telenors 200-årsprosjekt figur C.6.
- Abeo figur C.7.

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Applikasjonsfeil 1. Lar seg ikke bygge 2. Lar seg bygge, men ikke installere 3. Bygge og installere, men ikke kjøre	1. Lav/ middels 2. Høy 3. Høy	2 ganger per 10 000 kodelinje Lav	1.Lav 2.Mid-dels 3.Mid-dels	Installere automatiske regresjons-tester. Kjøre røyk-test Utviklere rette kode. ACI-automatisering. Continous Integration Cruise-kontroll. Enhetstester, JUnit-tester, kodesjekk.	1. Liten 2. Liten 3. Middels	Iallfall 1 per 10 000, kanskje 1 per 20 000 kodelinjer. Høy	19 99 24	Testmiljø-ansvarlig
Hardwarefeil	Høy	Ca. 1 gang per år Lav	Mid-dels	Innføre disaster recovery- metodikk VMware kan være aktuell. Har løsning på Unix som kan gjennomføres i løpet av 2-3 timer, ikke dokumentert Dupliserer kritiske servere	Disaster recovery: Høy initiell Lav vedlike-hold. Lav ved duplisering	VMware: Vedlikehold i arbeidstid Tiltak her kan forbedre andre ting Høy	24 99	Testmiljø-ansvarlig
Softwarefeil, feil i 3. parts programvare	Høy	Ca 1 gang per prosjekt Høy	Høy	Programmere seg rundt problemet eller vente på fiks, installere fiks	Lav	Høy effekt, maksimal	999	Utviklere og testmiljø-ansvarlig
Konfigurasjonsfeil	Høy	2 ganger per prosjekt Høy	Høy	Registrere hvem som gjør endringer. Ved gjenganger: tiltak f.eks. sjekklister, automatisering	Lav/ middels	Sjekklister: Lav Automati-sering: Høy	49 449	Testmiljø-ansvarlig, system-ansvarlig
Problem ved bytte av freeware utviklingsmiljø til produksjonslikt milø	Lav	Ca. 10 000 kodelinjer per feil Lav	Lav	Bruke samme system hele veien. Oversikt over Websphere. Oprette og bruke en klasse-sammenheng	Høy Middels Lav	Høy Lav Middels	0 0 3	Testmiljø-ansvarlig
Feil i testdata (Inkonsistens)	Middels/ høy	3-4 ganger per år Middels	Mid-dels	Automatiserte skript for synkronisering Rutiner for at alle system blir med i oppfriskning	Middels	Skript: Høy Rutiner: Middels	69 27	Testmiljø-ansvarlig, system-ansvarlig
Feil versjon i testmiljø	Høy/ middels	6-8 ganger per år Middels	Mid-dels	To dedikerte systemer: Et videreutviklings-miljø og et produksjonslikt-miljø	Implement-ering: Middels/ høy Vedlike-hold: Middels/ lav	Høy	45 106	Testmiljø-ansvarlig, system-ansvarlig

Figur C.1: Risikoanalyse EDB Oslo

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Presset på tid og får leveranse for sent til testmiljøet eller er ikke ferdig og leverer i faser	Dårlige kvalitet på leveranse, dårligere enhets-terster Høy	Skjer ofte, men avhengig av prosjekt Middels	Middels/ høy	Tidlig inn i prosjektet, være med å estimere og få prosjektet til evnt. å utsette Nekte å ta i mot programvaren, f.eks. ved enhetstest	Mye overtid for å komme i havn. Middels Middels	Ikke noe særlig effekt per i dag. Middels Høy	39 99	Trygve Testleder
Ny teknologi, problemer med stabilitet, ytelse, samhand-ling. Ferske utviklere på ny teknologi	Ustabilt testmiljø, test blir forsinket i integra-sjonstest Høy	Lav	Middels	It-drift og 3. partsleveran-dør til å sette opp testmiljø sammen med test Ta med ny teknologi i risikobildet, utvide prosjekt-perioden	Middels Høy	Høy Høy	24 9	Prosjekt-leder/ testmiljø-ansvarlig Prosjekt-leder
Kompo-nent fra 3. parts testmiljø. Utfor-dringer rundt testdata. Tidkrev-ende	Får ikke opp hele testmiljøet til avtalt tid, får ikke testet verdi-kjeden Middels	Lav	Lav/ Middels	Tidlig i prosjektet starter med testmiljø, tar kontakt med 3. part	Lav	Høy	39	Prosjekt-leder / testmiljø-ansvarlig
Infra-struktur i EDB, avansert, f.eks brannmur er, rutere. Driftere byrå-kratisk organisert, bruker bestillings verktøy	Klarer ikke å levere testmiljøet til avtalt tid. Høy	Middels	Middels/ høy	Tidlig inn i prosjektet, bestille tidlig Dra med personer fra infrastruktur inn i prosjektet	Lav Middels	Høy Høy	399 99	Prosjekt-leder, testmiljø-ansvarlig Prosjekt-eder

Figur C.2: Risikoanalyse EDB Trondheim side 1

Testmiljø der SW ikke utvikles selv, og kode kommer fra 3. part	Leverer ikke SW i henhold til etablerte rutiner, klarer ikke å styre godt nok ved rettelser fra dem. Middels	Lav	Lav/ middels	Tydelige krav til 3. parts leverandører tidlig i prosjektet ang hvordan ting skal leveres, konsekvens av feil. Kontraksfestet	Lav	Høy	39	Produkt-eier
Lang bestilling-stid/ leverings-tid på servere	Klarer ikke levere testmiljø i tide Høy	Lav	Middels	Tegnet opp behovet for hardware i budsjettprosess en for å være i forkant Delegere økonomiske beslutninger på lavere nivå	Lav Lav	Høy Høy	99 39	Utviklings-ledere, Trygve Forretni-ngsenhets-leder
Treghet i infrastruk-turlever-anse fra IT-drift	Ikke etablert server tidsnok, testmiljø ikke etter avtale Høy	Middels	Middels/ høy	Få på plass OLA-avtale med IT-drift som gir bedre betingelser tidsmessig enn standard	Lav	Høy	279	Trygve

Figur C.3: Risikoanalyse EDB Trondheim side 2

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Testmiljø ikke dimensjonert for aktuell testing	Ved gjennomgått akseptanse-test, er testen egentlig god nok? Høy	Hvis få endring-er i database n, større sjanse for at det går bra. Middels	Middels/ høy	Bruke "pisk" Synliggjøre konsekvens for leverandør Holdnings-skapende arbeid, konsekvens av å gjøre lite gjennomtenkte ting Siste leveranse: Involverte testansvarlig hos Computas, fikk se hvordan DA tester Deltar også i systemtest til Computas	Tid, finansierer testmiljø til Computas Lav kostnad Lav kostnad	Medi-um Høy	159 399	Kvalitet-leder Kvalitet-leder
Testmiljø inneholder ikke funksjonali-tet for gjennomfør-ing av test	Får ikke testet den funksjonalite-ten, vanskelig å finne feil før produksjon Høy	Medium	Middels/ høy	Etablere et testmiljø som dekker all funksjonalitet som finnes i produksjon	Høy kostnad	Høy	39	Kvalitet-leder
Manglende dokumen-tasjon	Medium/høy	Høy	Høy	Mer dokumentasjon	Lav/middels arbeidstimer pga Computas må også involveres	Høy	349	Kvalitet-leder

Figur C.4: Risikoanalyse Domstoladministrasjonen i Trondheim

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Feil i testmiljøet, ikke produksjons-likt	Middels Finner mye feil, bruker tid på ting som egentlig ikke er feil, eller at de ikke vinner	Mid-dels	Mid-dels	Sørge for gode rutiner og prosedyrer Automatiske kontrollmek-anismer Kopi av produksjonsdata Ikke ønskelig, sensitive testdata Kopi av fysisk miljø	Middels Høy Lav Høy	M Høy Høy Høy	15 15 159 15	Testmiljø-ansvarlig
Mangel på ressurser til vedlikehold av testmiljø	Høy	Høy	Høy	Øke bemaninngen	Høy	Høy	99	Ledelse
Driftsstabili-tet	Høy F.eks maskiner går ned, feilsituasjoner i systemer fører til stopp	Lav	Mid-dels	Ha overvåking av testmiljøet (driftsovervåking)	Middels	Høy	24	Testmiljø-ansvarlig
Ressurser til innkjøp av testutstyr	Høy Ting går saktere Ikke sikker på å finne samme feil. Feks 32-bits kontra 64-bits.	Mid-dels	Høy	Kjøpe utstyr	Middels	Høy	99	Ledelse, de som prioriterer pengene

Figur C.5: Risikoanalyse HEMIT

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Testtiden blir for kort fordi utviklerne ikke er ferdig	Testtiden blir kortere, kvalitets-risiko. Høy	Høy	Høy	Bedre planlegging Høyere prioritering av test	Lav, leie inn utvikler-ressurser Lav	Høy Høy	999 999	Prosjektleder, sammen med budsjett-ansvarlig Samme som over
Testmiljøet er ikke som planlagt	Høy	Middels	Middels/ høy	Bedre prosedyrer og rutiner for etablering Automatisere bygging av testmiljø	Drift har ofte verktøy Lav Middels, ved kjøp av verktøy	Høy Høy	399 99	Testleder, QA-ansvarlig Samme som over
Manglende opplæring av testere på det aktuelle systemet	Høy, testen kan ikke utføres	Lav	Middels	Sikre opplæring Forberede seg på forandringer, tester eventuelt være med hele veien	Lav	Høy	99	Testplan-legger, prosjektleder, testleder
Manglende involvering med testfokus fra dag en	Høy	Middels	Middels/ høy	Ha fokus Krav til organisasjon-en	Lav Lav	Høy + Høy	399 399	Ledelse Ledelse
Manglende prioritert rekkefølge, feks gjør de tingene man kan først.	Kaster bort mye tid, Middels	Kommer ikke i mål. Middels	Middels	Test det vanskeligste først, ta største utfordringer og risko først	Lav Kan avbryte prosjektet med en gang	Meget høy	159	Prosjekt-metodikk, metodean-svarlig.
Opposisjon mot nye rutiner, rutiner generelt blant ansatte	Kan være katastrofal. Høy	Middels, de ekstreme tilfellene oppleves ikke så ofte	Middels/ høy	Forklare, motivere, selge, forberede	Lav	Høy	399	Ledelsen, helt på topp

Figur C.6: Risikoanalyse IBM

Hendelse	K	f	R	Tiltak	Kostnad	E	L	Ansvarlig
Testdata er ikke tilgjengelig når testutviklin g starter. Lav kvalitet på testdata	Begrenset dekningsgrad i testing, avhengig av testnivå, enhetstest kanskje mindre avhengig Høy	Alltid utfordring å få raskt nok på plass Middels	Middels/ høy Middels/ høy	Planlegge det inn i prosjektet, derfor ha testledelse som har tilstrekkelig fokus på problemet, og forstår konsekvensen for prosjektet Kontraktfeste testdataleveranse fra kunde	Lav, finne folk som har kompetansen, men dette kan være vanskelig tilgjengelig Lav	Høy Høy	399 399	Prosjektleder/ testleder Linjeledelse, ettersom kontrakter kan arves, prosjektleder /testleder
Testutviklerne har ikke rukket å opparbeide tilstrekkelig forretningsforståelse	Bruker for lang tid for å lage relevante tester. Belaster de i prosjektet som har forretningsforståelsen. Middels	Personavhengig Høy	Middels/ høy	Dedikerte testressurser tidligere inn i prosjektet.	Signifikante kostnader. Middels/ høy	Middels	22	Prosjektleder
Ikke nok fokus på enhetstestnivå og intergrasjonstestnivå	Feil oppdages for sent. Leder ofte til over. Høy	Middels	Middels/ høy	Sørge for at kvalitetsleder stiller krav. Leder av utviklingsgruppe r må ha fokus. Testleder er eier av alle krav til alle testnivå	Det kvalitetsleder vil stille krav om koster. Utvidet enhetstesting. Middels	Høy	99	Prosjektleder/ kvalitetsleder/ testleder

Figur C.7: Risikoanalyse Abeo

Tillegg D
Certified Software Tester

D.1 Build The Test Environment

2005-utgaven av sertifiseringen.

A **Test Standards**

1. **External Standard** - Familiarity with and adoption of industry test standards from organizations such as IEEE, NIST, DoD, and ISO.
2. **Internal Standards** - Development and enforcement of the test standards that testers must meet.

B **Test Environment Components**

1. **Test Process Engineering** - Developing test processes that lead to efficient and effective production of testing activities and products.
2. **Tool Development and/or Acquisition** - Acquiring and using the test tools, methods, and skills needed for test development, execution, tracking, and analysis (both manual and automated tools including test management tools).
3. **Acquisition or Development of a Test Bed/Test Lab/-Test Environment** - Designing, developing, and acquiring a test environment that simulates the real world, including capability to create and maintain test data.

C **Test Tools**

1. **Tool Competency** - Ability to use
   (a) automated regression testing tools
   (b) defect tracking tools
   (c) performance/load testing tools
   (d) manual tools such as checklists, test scripts, and decision tables
   (e) traceability tools
   (f) code coverage tools.

2. **Tool Selection (from acquired tools)** - Select and use tools effectively to support the test plan; and test processes.

D **Quality Assurance / Quality Control**

1. **Quality Assurance versus Quality Control** - Being able to distinguish between those activities that modify the development processes to prevent the introduction of flaws (QA) and those activities that find and correct flaws (QC). Sometimes this is referred to as preventive versus detective quality methods.

2. **Process Analysis and Understanding** - Ability to analyze gathered data in order to understand a process and its strengths and weaknesses. Ability to watch a process in motion, so that recommendations can be made to remove flaw-introducing actions and build upon successful flaw- avoidance and flaw-detection resources.

E **Building the Test Environment Work Processes**

1. **Concepts of work processes** - understanding the concepts of policies, standards and procedures and their integration into work processes.

2. **Building a Test Work Process** - an understanding of the tester's role in building a test work process.

3. **Quality Control** - Test quality control is verification that the test process has been performed correctly.

4. **Analysis of the Test Process** - The test process should be analyzed to ensure:

   (a) The test objectives are applicable, reasonable, adequate, feasible, and affordable.

   (b) The test program meets the test objectives.

   (c) The correct test program is being applied to the project.

   (d) The test methodology, including the processes, infrastructure, tools, methods, and planned work products and reviews, is adequate to ensure that the test program is conducted correctly.

   (e) The test work products are adequate to meet the test objectives.

   (f) Test progress, performance, processes, and process adherence are assessed to determine the adequacy of the test program.

   (g) Adequate, not excessive, testing is performed.

5. **Continuous Improvement** - Continuous improvement is identifying and making continuous improvement to the test process using formal process improvement processes.

F **Adapting the Test Environment to Different Technologies** - The test environment must be established to properly test the technologies used in the software system under test. These technologies might include:

1. Security/Privacy
2. Client Server
3. Web Based Systems
4. E-Commerce
5. E-Business
6. Enterprise Resource Planning (ERP)
7. Business Reengineering
8. Customer Relationship Management (CRM)
9. Supply Chain Management (SCM)
10. Knowledge Management
11. Applications Service Providers
12. Data Warehousing

(Certified Software Tester 2005)

D.2 Building The Test Environment

2006-utgaven av beskrivelsen av kunnskapsområdet testmiljø.

The test environment is comprised of all the conditions, circumstances, and influences surrounding and affecting the testing of software. The environment includes the organization's policies, procedures, culture, attitudes, rewards, test processes, test tools, methods for developing and improving test processes, management's support of software testing, as well as any test labs developed for the purpose of testing software and multiple operating environments.

This category also includes assuring the test environment fairly represents the production environment to enable realistic testing to occur. Specifically this knowledge category will address:

Knowledge of Test Process Selection and Analysis

1. Concepts of Test Processes - the concepts of policies, standards and procedures and their integration into test process.
2. Test Process Selection - selecting test processes that lead to efficient and effective testing activities and products.

3. Acquisition or Development of a Test Bed/Test Lab/Test Processes - designing, developing, and acquiring a test environment that simulates "the real world," including capability to create and maintain test data.
4. Test Quality Control - test quality control to assure that the test process has been performed correctly.
5. Analysis of the Test Process - the test process should be analyzed to ensure:
   - The effectiveness and efficiency of test processes.
   - The test objectives are applicable, reasonable, adequate, feasible, and affordable.
   - The test program meets the test objectives.
   - The correct test program is being applied to the project.
   - The test methodology, including the processes, infrastructure, tools, methods, and planned work products and reviews, is adequate to ensure that the test program is conducted correctly.
   - The test work products are adequate to meet the test objectives.
   - Test progress, performance, processes, and process adherence are assessed to determine the adequacy of the test program.
   - Adequate, not excessive, testing is performed.
6. Continuous Improvement ? identifying and making improvements to the test process using formal process improvement processes.
7. Adapting the Test Environment to Different Software Development Methodologies ? the test environment must be established to properly test the methodologies used to build software systems such as waterfall, web-based, object oriented, agile, etc.
8. Competency of the Software Testers ? management must provide the training necessary to assure that their software testers are competent in the processes and tools included in the test environment.

Test tools

1. Tool Development and/or Acquisition ? understand the processes for acquiring and using test tools, methods, and understand the skills needed for test development, execution, tracking, and analysis tools. (Both manual and automated tools including test management tools).
2. Tool Usage ? understanding of how tools are used for:
   - automated regression testing tools
   - defect management tools
   - performance/load testing tools
   - manual tools such as checklists, test scripts, and decision tables; traceability tools

- code coverage
- test case management tools
- common tools to aid in testing such as an excel spreadsheet.

Management Support for Effective Testing

1. Management must create a "tone" that encourages software testers to do their work in an efficient and effective manner. This is accomplished through test policies, management support of those policies, open communication between management and testers, and enforcing compliance to policies and processes.
2. Test processes must align with organizational goals, user business objectives, release cycles and different developmental methodologies.

Tillegg E
Spesifikasjon av testmiljø

Specification of test environment

Informasjon om denne malen.
Display with "hidden taxt" visible!
Specification of test environment.

Document ID:
Saved: **Last updated:**
Author:
Version number:
Approval participants:

	Approved by	Date

Figur E.1: Skjema for beskrivelse av testmiljø side 1 (Schaefer 2005a)

Specification of test environment

1 Objective with this document

-
-

2 Access to the test environment

Test environment must be ready the:
Test environment is free from:
Test environment accessibility requirements outside normal working hours:

3 Application Configurations to be tested

4 Drawing of test environment infrastructure

Figur E.2: Skjema for beskrivelse av testmiljø (Schaefer 2005a)

Specification of test environment

5 Server Software

Resource description	Database server	Mainframe
Applications with version numbers		
OS and version number		
Middleware with version numbers		
Database and version number		
Development tools and version numbers		

6 Client Software

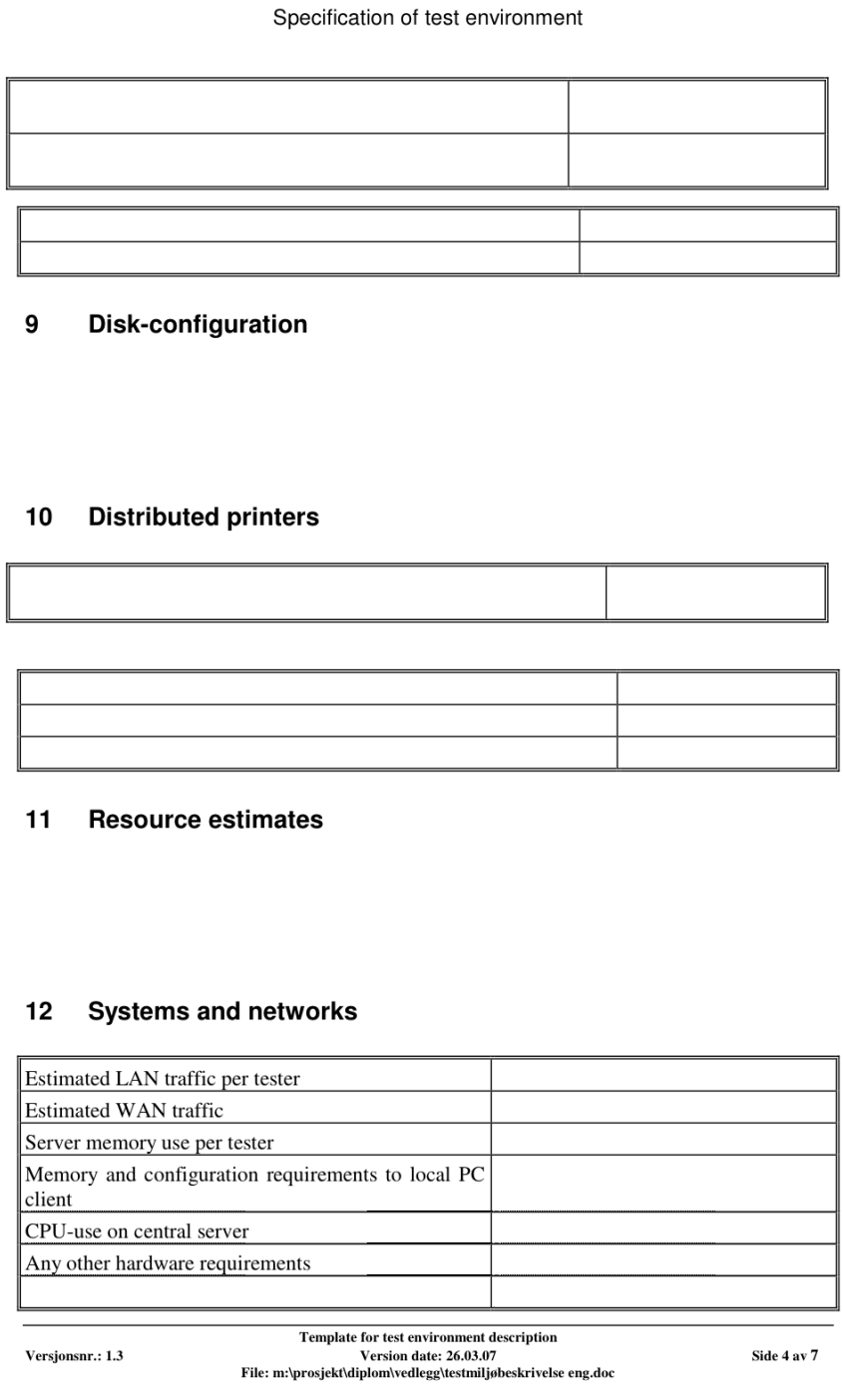
Resource description	PC server	PC client
Apøplications with version numbers		
OS with version number		
MiddleWare with version numbers		
Database and version number		
Development tools and version numbers		

7 Special Test lab infrastructure

Simulators
Test tools
Special test hardware
Cables

8 Licenses

Figur E.3: Skjema for beskrivelse av testmiljø (Schaefer 2005a)



Specification of test environment

9 Disk-configuration

10 Distributed printers

11 Resource estimates

12 Systems and networks

Estimated LAN traffic per tester	
Estimated WAN traffic	
Server memory use per tester	
Memory and configuration requirements to local PC client	
CPU-use on central server	
Any other hardware requirements	

Template for test environment description

Versjonsnr.: 1.3 **Version date: 26.03.07** **Side 4 av 7**

File: m:\prosjekt\diplom\vedlegg\testmiljøbeskrivelse eng.doc

Figur E.4: Skjema for beskrivelse av testmiljø (Schaefer 2005a)

13 External memory requirements (disks, tapes, …)

Resource description	Database-server	PC-server	PC-client	Batch
Requirements to disk capacity for other than databases				
Requirements for disk capacity for databases				

14 Transactions

Transaction	Transaction amount	How will this change during the test, during future versions?

15 Dependencies on other applications under test

Description of dependencies	**Responsible person**

16 Administration during testing

16.1 Stop and start of application on central server

How to start the application normally	
How to stop the application	
How to start the application in unnormal situa-	

Figur E.5: Skjema for beskrivelse av testmiljø (Schaefer 2005a)

tions, for example after a crash	

16.2 Stop and start of database under testing

How to load the database with starting test data	
How to re-initialize the database	

16.3 Stop and start of network under testing

How to initialize the network	
How to re-initialize the network	
How to decouple the network	

17 Data description

17.1 DB2

- **DDL - for definition**
- **PLANs – to bind**
- **Who shall have access**
- **How much memory?**
- **Special backup and recovery requirements**

17.2 IMS

- **DBD-source plus infor about how much memory**
- **PSB-source**
- **MPP´s**
- **BMP´s**
- **Transaction codes**
- **programs**
- **special requirements for backup and recovery**

Figur E.6: Skjema for beskrivelse av testmiljø (Schaefer 2005a)

17.3 CICS

- **files-VSAMSRCE**
- **transaction codes**
- **programs**
- **special requirements for backup and recovery**

17.4 IDMS

17.5 Oracle

17.6 Sybase

17.7 Others

18 Backup, establishment of start points andrestart points

19 Reference documentation

Doc. Reference	Description of title and content

Figur E.7: Skjema for beskrivelse av testmiljø (Schaefer 2005a)

Tillegg F
Testmiljø i TMM

Test Environment

Introduction
The *purpose* of 'Test Environment'[5] is to establish and maintain an environment in which it is possible to execute the specified tests in a manageable and repeatable way.

A test environment is needed to obtain test results under conditions which are as close as possible to the 'real-life' situation. This is especially true for higher level testing. Furthermore, at any test level the reproducibility of test results should not be endangered by undesired or unknown changes in the test environment.

Specification of the test environment is carried out early in the projects. The specification is reviewed to ensure their correctness, suitability, feasibility and its representativeness towards the 'real-life' environment. Early specification has the advantage that there is more time to order and/or develop any specific simulators, files, stubs or drivers.

Availability of a test environment encompasses a number of issues which need to be dealt with: Is it necessary for testing to have an environment per test level? A separate test environment can be very expensive. It must therefore be decided how to use them as efficiently as possible. Maybe it is possible to have the same environment shared between testers and developers. But then strict management and control is necessary as both testing and development activities are done in the same environment. When poorly managed, this situation can cause many problems ranging from conflicting reservations to people finding the environment in an unknown or undesired state when starting one's activities.

Throughout the project the test environment is subject to changes due to for example hardware changes, incremental test environment development and changes in the test object. Thorough (configuration) management on the test environment is needed to cope with these changes.

Scope
The process area 'Test Environment' addresses all activities for specifying, managing (including configuration management), and ensuring availability of an adequate test environment for both lower and higher test levels.

Goals

Goal 1	Test environments are specified early, and their availability is ensured on- time in projects.
Goal 2	For higher test levels the test environment is as much as possible "real-life".
Goal 3	Test environments are managed and controlled using documented procedures.

Common features

Commitment to perform

Policy 1	The projects follow a documented organisational policy for specifying, managing and controlling the test environment. This policy typically specifies:

[5] Although the process area 'Test Environment' has not been part of the initial TMM framework as published by IIT, many authors have since then added 'Test Environment' as a fourth process area at TMM level 2.

Figur F.1: Testmiljø i TMM side 1 (van Veenendaal 2006)

1 A summary report is prepared and distributed to the affected groups

2 Specification, management and control of test environments is performed according to documented procedures

3 Responsibilities with respect to test environment are assigned

4 Test environment specification shall be done early in the life cycle and is part of the test planning process

5 Higher levels tests will be carried out in an environment that is as much as possible 'real-life'

6 Lower level tests, e.g. unit and integration testing, shall apply stubs and drivers for testing

Ability to perform

Resources 1

Adequate resources and funding are provided for specifying, managing and controlling the test environment.

1 Experienced individuals, who have expertise and technical knowledge are available to support the specification and implementation of the test environment

2 Configuration management tooling is available for managing (changes to) the test environment

3 Adequate time and resources are provided to the project to implement an adequate test environment

4 Adequate time and resources are provided to engineers to develop stubs and drivers needed for lower level testing

Responsibility 1

A group (or person) is designated to be responsible for managing and controlling the test environment.

1 The responsibility typically covers (Pol *et al*, 2002):

- ❑ Support regarding test environment specification
- ❑ Implementation of test environment
- ❑ Configuration management of the test environment
- ❑ Solving technical problems related to the test environment
- ❑ Ensuring that tests are reproducible with respect to the test environment
- ❑ Support and consultancy on test environment-related procedures and technical issues
- ❑ Ensuring the availability of the test environment

Process 1

The test planning procedure addresses the specification of the test environment.

1 The test planning procedure typically consists of the following tasks:

- ❑ Defining test environment requirements
- ❑ Defining a specification of test environment
- ❑ Identifying risks of non-conformance to test environment requirements
- ❑ Defining tasks and responsibilities
- ❑ Estimating of test environment related costs, effort and schedule

Figur F.2: Testmiljø i TMM side 2 (van Veenendaal 2006)

Process 2

The procedures for managing and controlling the test environment are specified and documented.

1 The procedures typically address of the following issues:

- Back-up and restore
- Change management
- Configuration management

Process 3

A procedure for ensuring the availability of the test environment is specified and documented.

1 The procedure typically consists of the following parts:

- Making reservations for the test environment
- Aligning time slots between testing and development
- Shutting-down the environment correctly after usage (e.g. information on how to update the environment's change log, bring it back in a known state and removing test files)

Activities performed

Activity 1

The test environment is specified early in the project according to a documented procedure.

1 The test environment specification typically includes:

- Network components
- Software components (e.g. operating systems, firmware)
- Simulators, stubs and drivers
- Identification of supporting documentation (e.g. user guides, technical guides, installation manuals)
- Tools to support the development of stubs and drivers

2 The test environment specification is reviewed by:

- Project management
- Technical experts
- Other affected groups

3 The specification is typically reviewed on:

- Technical correctness
- Suitability of the environment for test purposes
- 'Real-life' environment representativeness
- Technical and economical feasibility
- Feasibility of on-time delivery

Activity 2

Higher test levels are performed in a test environment that is as much as possible 'real-life.

1 The operational test environment is in compliance with the specified requirements

2 Risks of non-conformances to the test environment requirements are discussed with management

Activity 3

Management and control of the test environment is carried out according to a documented procedure.

1 Proposed changes to the test environment are reviewed by test management

Figur F.3: Testmiljø i TMM side 3 (van Veenendaal 2006)

Activity 4

The availability and usage of the test environment is coordinated according to a documented procedure.

Activity 5

Test environment incidents are reported according to a documented procedure.

1 Test environment incidents identified are documented and tracked to closure

2 Test environment incidents are managed and controlled

Note: refer to Process 3 of the Test Techniques and Methods process area for practices covering incident reporting and management.

Directing implementation

Measurement 1

Goal-oriented measurements are made and used to determine the status of the test environment activities.

1 Measurement are based upon the test environment process area and upon the organisational policy

2 Measurements focus the level of deployment, and the effectiveness and efficiency of the test environment activities

Examples of measurements include:
- Number of conflicting environment reservations
- Effort needed for maintenance, repair and updates
- Number of test case failures due to the test environment
- Average down-time of the test environment
- Number of test environment incidents reported
- Percentage of test environments available on time and according to specification

Verifying implementation

Adherence 1

The quality assurance group reviews and/or audits the activities and work products for test environment.

1 Results are reported to:

- Resource management
- Test management
- Project management

2 At a minimum, the reviews and/or audits verify that:

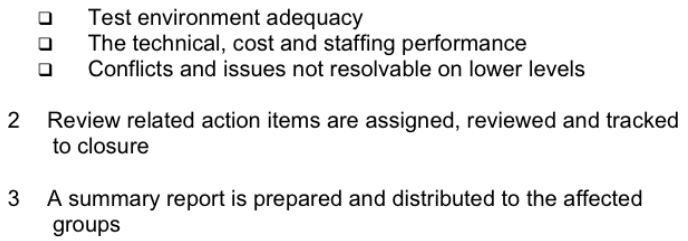
- A test environment specification is written early in the project according to a documented procedure
- The test environment is a much as possible 'real-life', especially for higher test levels
- The availability of the test environment is at an adequate level and carried out according to a documented procedure
- Test environment management and control is carried out according to a documented procedure

Review 1

The activities for test environment are reviewed with management on a periodic basis and event driven basis.

1 Issues typically addressed are:

Figur F.4: Testmiljø i TMM side 4 (van Veenendaal 2006)



Figur F.5: Testmiljø i TMM side 5 (van Veenendaal 2006)

Tillegg G
Akronym

ATR Activities, tasks, and responsabilities

BBS Bankenes betalingssentral

CPU Central Processing Unit

CSTE Certified Software Tester

DA Domstoladministrasjonen

EKG Elektrokardiogram

FAT Funksjonell akseptansetest

GUI Graphical User Interface

HEMIT Helse Midt-Norge IT

IDE Integrated Development Environment

ISTQB International Software Testing Qualifications Board

ITIL Information Technology Infrastructure Library

MS Microsoft

NTNU Norges teknisk-naturvitenskapelig universitet

QA Quality Assurance

RUP Rational Unified Process

SAT Produksjonsbasert akseptansetest

SOA Service-Oriented Architecture

Tcl Tool command language

TSL Test Script Language

TMM Testing Maturity Model

QA Quality Assurence

QTP QuickTest Professional