# Teststrategi for ytelsestesting

## IKT-testing i Helse Nord

**Ansvarlig:**

Faglig ansvarlig for innhold og revisjon, Testseksjonen TestiT, Avd. for Tjenesteproduksjon HN IKT

| Endring | Versjon | Rolle / Organisasjon | Revidert |
|---|---|---|---|
| Revisjon av Strategi for ytelsestesting v.1.0 | v.1.1 | Testseksjonen, HN IKT | 2018 |

**Godkjenning**

Dette dokumentet er godkjent av:

| Versjon | Godkjent dato | Navn | Rolle / Organisasjon |
|---|---|---|---|
| | | | |

HELSE NORD IKT

# INNHOLD

HELSE NORD IKT

# Terminologi

| Term/konsept | Beskrivelse |
|---|---|
| **Baseline** | Å skape en *baseline* er prosessen med å kjøre et sett med tester for å fange ytelsesmetriske data for det formål å vurdere effekten av de påfølgende ytelsesforbedrende endringer i systemet, komponenten eller applikasjonen. |
| **Belastning** | *Belastning* er stimulansen et system, applikasjon, eller komponent eksponeres for, for å simulere et scenario. En belastning består av det totale antallet brukere, samtidige aktive brukere, datamengder og et transaksjonsvolum. |
| **Cache/buffer** | En *cache* eller buffer er en er en komponent som lagrer data slik at fremtidige forespørsler om den dataen kan serveres raskere; dataen som er lagret i en buffer kan være resultatet av en tidligere beregning, eller en kopi av data som er lagret et annet sted. |
| **Kontroller / HP LoadRunner Controller** | HP LoadRunner Controller (kontroller) sender ytelsesteststcript til HP Load Generator (lastgenerator) for eksekvering. I kontrolleren settes et scenario opp som gjenspeiler lastmodellen. *Ytelsesdata* som måles, spesifiseres i kontrolleren og gir mulighet for sanntids innsamling av ytelsesdata under eksekvering. |
| **Enhetstesting** | En *enhetstest* er hvilken som helst type ytelsestest som er rettet mot en arkitektonisk komponent i applikasjonen. Servere, databaser, nettverk, brannmurer og lagringsenheter er vanlige komponenter som testes. |
| **Gjennomstrømning / Throughput** | *Gjennomstrømning (throughput)* er antall enheter av arbeid som kan behandles per tidsenhet; f.eks. forespørsler per sekund, samtaler per dag, treff per sekund, rapporter per år etc. |
| **Kapasitet** | *Kapasiteten* til et system er den totale belastningen det kan håndtere uten å bryte forhåndsbestemte akseptansekriterier for ytelse. |
| **Lastgenerator / Injector / Load Host** | En *lastgenerator* er en maskin (server eller klient, fysisk eller virtuell) som ikke brukes til noe annet enn å generere belastning. Det er ofte installert en last agent f.eks. HP Load Generator, som eksekverer ytelsesscriptene samt samler inn ytelsesdata. |
| **Latens / Latency** | *Latens* er et mål på respons som representerer tiden det tar å fullføre en eksekvering av en forespørsel. Latens kan også representere summen av flere latenser. |
| **Peak hour** | *Peak hour* er den tiden på døgnet når trafikken og dermed belastningen er som størst på systemet, applikasjonen og komponentene. |
| **Ramp-up /-down** | En gradvis øking eller reduksjon av belastning. |

| Term/konsept | Beskrivelse |
|---|---|
| **Responstid / Svartid** | Responstiden er tiden det tar fra en forespørsel er sendt fra en klient til klienten mottar et svar. Måles normalt i millisekund. |
| **Ressursutnyttelsesnivåer** | Informasjon om ressursforbruket for forskjellige system komponentene f.eks. CPU- eller minneforbruk. |
| **Scenario** | Et *scenario* er en sekvens av steg i applikasjonen som skal testes. Et scenario kan representere et brukersenario eller en virksomhetsprosess f.eks. søke i en produktkatalog, legge et produkt i en handlevogn eller plassere en ordre. |
| **Smoke test** | En *smoke test* er den første kjøring av en ytelsestest for å verifisere at applikasjonen eller systemet er klart for å ytelsestestes. |
| **Ytelsesdata / Performance Counters** | *Ytelsesdata* fra målinger kan brukes til å gi informasjon om hvor godt et operativsystem, en applikasjon, tjeneste eller driver presterer. Et operativsystem, nettverk og enheter er typiske kilder for ytelsesdata som igjen kan benyttes til å gi brukerne en grafisk visning av hvor godt systemet yter. Det kan f.eks. være CPU- eller minneforbruk. |
| **Transaksjon** | En *transaksjon* en eksekvering av en brukerfunksjon i systemer eller applikasjonen. I ytelsestestverktøyet LoadRunner spesifiseres start- og sluttpunktet for en transaksjon. Rapporteringen foregår ofte per transaksjon. |
| **Ytelse** | *Ytelse* viser til informasjon om programmets responstid, gjennomstrømming (throughput) og ressursutnyttelsesnivåer. |
| **Ytelsestest** | En *ytelsestest* er en teknisk undersøkelse gjort for å bestemme eller validere produktets egenskaper hastighet, skalerbarhet og/eller stabilitet under test. Ytelsestesting er den overordnede kategorien som inneholder alle andre underkategorier av ytelsestesting. |

**Tabell 1: Terminologi**

# 1. INNLEDNING

Dette dokument beskriver Helse Nord IKTs ytelsestestprosess og testtilnærming samt de tjenestene som Helse Nord IKT tilbyr av ytelsestesting. Dokumentet er underordnet Helse Nord IKTs til enhver tid gjeldende testpolicy og teststrategi. Ytelsestestprosessen med tilhørende testtilnærming skal benyttes i samband med alle leveranser av ytelsestesttjenester i Helse Nord IKT. Dette dokument er et levende dokument som vurderes fortløpende og tilpasses de endringer som skjer i Helse Nord IKT.

## 1.1 Referanse til styrende dokumenter

| Dokument | Beskrivelse | Lagret |
|---|---|---|
| Testpolicy | Helse Nord IKTs testpolicy. | |
| Teststrategi | Helse Nord IKTs teststrategi. | |

**Tabell 2: Styrende dokumenter**

# 2. HENSIKT

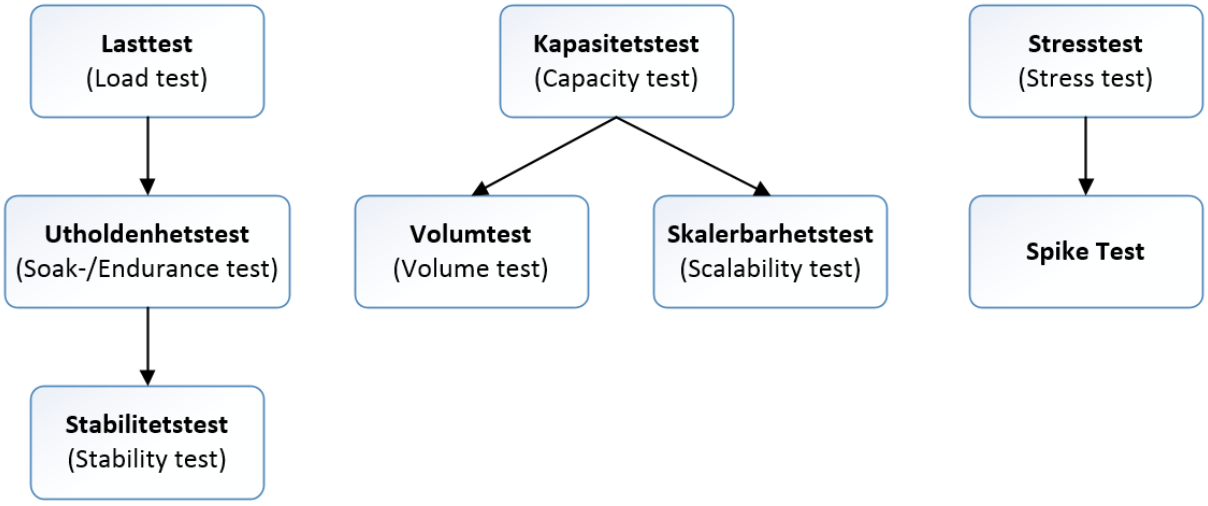
Ytelsestesting utføres vanligvis for å bidra til å identifisere flaskehalser i et system, etablere en baseline for fremtidig testing, støtte ytelsesrelaterte forbedringstiltak, verifisere oppnåelse av krav til ytelse og/eller samle andre ytelsesrelatertdata for å hjelpe interessenter ta informerte beslutninger relaterte til den generelle kvaliteten på applikasjonen som blir testet. Noen mer spesifikke grunner for å gjennomføre ytelsestesting inkluderer:

- Vurdere produksjonssettingsmodenheten:
  - Forutse eller beregne ytelsesegenskapene til en applikasjon, system eller komponent i produksjon og vurdere om eventuelle bekymringer vedrørende ytelse skal adresseres eller ikke. Disse forutsigelsene er også verdifull for interessenter som tar beslutninger om en applikasjon, system eller komponent er klar for produksjonssetting, i stand til å håndtere fremtidig vekst, eller om det krever en ytelsesforbedring/hardware oppgradering før produksjonssetting
  - Fremlegge data som indikerer sannsynligheten for brukeres misnøye med ytelsesegenskapene til systemet
  - Fremlegge data for å støtte estimeringen av inntektstap eller skadet merkevare på grunn av skalerbarhet, stabilitetsproblemer eller brukeres misnøye med ytelsesegenskapene til systemet.
- Vurdere infrastrukturen:
  - Sammenligning av ulike systemkonfigurasjoner for å finne ut hva som fungerer best for både applikasjonen og virksomheten
  - Kontrollere at applikasjonen viser de ønskede ytelsesegenskapene innenfor budsjetterte ressursbegrensninger
- Vurdere en programvares ytelsesegenskaper:
  - Fastslå en programvares ønskede ytelsesegenskaper før og etter endringer i programvaren
  - Fremlegge sammenligninger mellom programmets nåværende og ønskede ytelsesegenskaper
- Effektivisering av arbeidet med ytelsesforbedrende tiltak:

- Analysere applikasjonen, systemets eller komponentens atferd ved ulike belastningsnivåer
- Identifisere flaskehalser i systemet eller applikasjonen
- Fremlegge informasjon vedrørende hastighet, skalerbarhet og stabilitet for et system før produksjonssetting. Dette gir grunnlag for ta informerte beslutninger om systemet skal endres eller konfigureres annerledes for å eventuelt øke ytelsen

# 3. YTELSESTJENESTER

I dette avsnittet beskrives de ytelsestesttjenester som Helse Nord IKT leverer. De typer av ytelsestester som kan utføres er oppdelt i tre hovedkategorier: lasttest, kapasitetstest og stresstest. Se skissen i Figur 1 for å få et overblikk over de vanligste typene av ytelsestest.




**Figur 1: Typer av ytelsestester**

## 3.1 Lasttest

Lasttest fokuserer på å bestemme eller validere egenskapene til et system eller en applikasjon når det utsettes for den belastning og volum som er forventet i ordinærproduksjon. Lasttesting gjennomføres ofte for å bekrefte at applikasjonen oppnår ytelsestestkriteriene som er spesifisert i avtaler i form av krav eller kriterier spesifisert i prosjekter. En lasttest måler ytelsen i form av responstider, gjennomstrømming og ressursutnyttelsesgrad.

Lasttester kan blant annet brukes til:

- Bestemme gjennomstrømningshastigheten som kreves for å supportere maksimum forventet belastning
- Oppdage samtidighetsproblemer og funksjonelle problemer som kan oppstå under last. Data som samles inn ved lasttesting kan også brukes til skalering- og kapasitetsplanleggingsaktiviteter eller beregne antallet brukere som systemet kan håndtere før ytelsen degraderer

- Skalering- og kapasitetsplaneringsaktiviteter eller beregne antallet brukere som systemet kan håndtere før ytelsen degraderer

## 3.2 Utholdenhetstest og Stabilitetstest

Utholdenhetstest er en underkategori av lasttest. En utholdenhetstest er en type ytelsestest fokusert på å bestemme eller validere egenskapene til produktet under test når det utsettes for belastningsmodeller og volumer forventet under produksjonsvirksomhet over en lengre periode. Utholdenhetstester kan blant annet brukes for å regne ut gjennomsnittstid mellom feil (*Mean Time Between Failure)* og gjennomsnittstid til feil (*Mean Time To Failure)*. Stabilitetstest er her en underkategori der en tester om systemet kontinuerlig fungerer tilfredsstillende over en gitt akseptabel periode.

## 3.3 Kapasitetstest

En kapasitetstest utfyller lasttesting ved å bestemme serverens svikt punkt, mens lasttesting monitorer resultater på ulike nivåer av last- og trafikkmønster.

Kapasitetstesting utføres i forbindelse med kapasitetsplanlegging, som brukes til å planlegge for fremtidig vekst, f.eks. en økt brukermasse eller økt volum av data. Det kan f.eks. være nødvendig å vite hvor mye ekstra ressurser (prosessor-, minne- eller diskkapasitet) som kreves for å imøtekomme fremtidige belastninger.

Kapasitetstesting er et hjelpemiddel til å identifisere en skaleringsstrategi for å avgjøre opp- eller utskalering.

### 3.3.1 Volumtest

Volumtest er en underkategori av kapasitetstest. Volumtest utføres for å verifisere at det ikke oppstår problemer med en realistisk eller maksimal mengde data. Volumtesting er nødvendig da funksjonelle tester normalt ikke bruker store mengder data, snarere det motsatte. Typiske problem er fulle eller nesten fulle disker, databaser, filer eller buffere som ikke er tilstrekkelig dimensjonert og tidsavbrudd på grunn av lang responstid.

## 3.4 Skalerbarhetstest

Skalerbarhetstest er en underkategori av Kapasitetstest. Skalerbarhetstest er en inkrementell økning av last for å finne ut om systemet skalerer proporsjonalt eller uproporsjonalt etter hvert som lasten øker. En typisk proporsjonal skalering er at hvis responstiden er ett sekund med 10 brukere, så er responstiden to sekunder med 20 brukere og tre sekunder responstid med 30 brukere. En uproporsjonal skalering kan typisk være en responstid på 1 sekund med 10 brukere, 2 sekunder med 50 brukere og 2 sekunder med 100 brukere. Dette gjøres for å finne ut om en må skalere opp et system etter hvert som belastningen øker over tid.

## 3.5 Stresstest

Stresstest er en test som normalt kjøres over kort tid for å sjekke om det oppstår ytelsesutfordringer ved belastning over maksimumbelastning på et system. Hensikten med testen er å oppdage avvik som kun oppstår ved høy belastning. Det kan for eksempel være

synkroniseringsproblemer, minnelekkasjer eller korrupte data. I noen tilfeller kjøres testen frem til systemet krasjer eller ytelsen forringer, dette for å måle flaskehalser.

### 3.5.1 Spike test

Spike test er en underkategori av stresstest fokusert på å bestemme eller validere ytelsen til et system, en applikasjon eller en komponent når denne utsettes for belastningsmodeller og/eller lastvolumer som under korte perioder gjentatte ganger øker utover forventet maksimalbelastning.

Spike testing gjøres ved å observere oppførselen til systemet når belastningen som genereres når antallet brukere plutselig øker betydelig. Målet er å finne ut om ytelse vil bli redusert, om systemet svikter eller om det vil være i stand til å håndtere dramatiske endringer i belastningen.

# 4. TESTTILNÆRMING

I Helse Nord IKT benyttes en risikobasert tilnærming for alle testaktiviteter. Dette avsnittet beskriver hvordan dette benyttes på ytelsestest. Utfra et risikobasertperspektiv hjelper dette avsnittet til med å velge hvilken type ytelsestest som er mest hensiktsmessig å utføre. Her følger en beskrivelse av hvilke risikoer de forskjellige ytelsestesttypene adresserer:

**Lasttest**

- Kan systemet håndtere den alminnelige forventede belastningen og maksimal forventet belastning?
- Hvor mye data kan databasen/filserveren håndtere?
- Er nettverkskapasiteten god nok?

**Utholdenhetstest**

- Er ytelsen konsistent over tid?
- Finnes det langsomt voksende problem som vi ikke har oppdaget enda?
- Finnes det utenforliggende forstyrrelser som vi ikke har regnet med?

**Stresstest**

- Hva skjer om belastningen i produksjon overstiger antatt belastning?
- Hvilke type feil bør vi forberede oss for?
- Hvilke indikasjoner skal vi se etter for å kunne agere før feil oppstår?

**Spike test**

- Hva skjer om belastningen i produksjon overstiger antatt belastning?
- Hvilke type feil bør vi forberede oss for?
- Hvilke indikasjoner skal vi se etter for å kunne agere før feil oppstår?

**Kapasitetstest**

- Klarer systemkapasiteten å håndtere belastningen under både normal og maksimalbelastning?

Denne tabellen kan benyttes ved valg av type ytelsestest fra et risikobasertperspektiv:

| Risk | Ytelsestesttyper | | | | | |
|---|---|---|---|---|---|---|
| | Kapasitets-test | Komponent-test | Utholdenhets-test | Last-test | Spike-test | Stress-test |
| **Hastighetsrelaterte risker** | | | | | | |
| Brukertilfredshet | | | X | X | | X |
| Negative responstid trender | | X | X | X | | |
| Konfigurasjon | | | X | X | | X |
| Konsistens | | X | X | X | | |
| **Skalerbarhetsrelaterte risker** | | | | | | |
| Kapasitet | X | X | X | X | | |
| Volum | X | X | X | X | | |
| Framtidig voks | X | X | | X | | |
| Ressursforbruk | X | X | X | X | X | X |
| Hardware/miljø | X | X | X | X | | X |
| **Stabilitetsrelatertrisker** | | | | | | |
| Pålitelighet | | X | X | X | X | X |
| Robusthet | | X | X | X | X | X |
| Hardware/miljø | | X | X | X | X | X |
| Lekkasje (minne- /CPU-lekkasje) | | X | X | X | | |
| Gjenopprettingsevne | | X | | | X | X |
| Integrasjoner | | X | X | X | | X |

**Tabell 3: Typer av ytelsestest fra et risikobasertperspektiv**

# 5. VERKTØY, RETNINGSLINJER OG TESTMILJØ

## 5.1 Testing internt i testmiljø og testing fra lokasjoner

Av de 3 hovedtypene ytelsestesting beskrevet i kapittel 2 ser vi at det i utgangspunktet bare er Lasttest som bør kjøres fra lokasjoner inn til sentrale system, og det under visse forutsetninger. En Lasttest skal i utgangspunktet ikke overstige belastningen som er beregnet og forventet. Forventet belastning vil da være basert på de ytelseskrav satt i avtaler og

Lasttesten skal representere denne. Det forutsettes her at nettverkstrafikk må overvåkes nøye og at en Lasttest blir terminert øyeblikkelig hvis den har en negativ innvirkning på eksisterende systemer. En slik negativ innvirkning vil da være tegn på at noe unormalt har skjedd og må avklares før systemet kan gå i produksjon. Denne øvelsen kan også benyttes for å teste oppgraderinger.

Normalt er det ikke noen grunn til at Stresstesting og Kapasitetstesting skal kjøres fra en lokasjon inn til et sentralt system. Ved Stresstest kjøres unormalt høy belastning lavere enn bristepunktet for å avdekke avvik. Ved en Kapasitetstest økes belastning til et system svikter og vi finner bristepunktet. Slike ytelsestester egner seg ikke over lange avstander og nett da eventuell treghet i nettverk vil kunne maskere andre flaskehalser i et system.

Stresstesting og Kapasitetstesting gir en voldsom belasting til et system inklusive nettverk. For å få et mest korrekt bilde av hvordan et system oppfører seg under ekstreme forhold, så hører disse to kategoriene av ytelsestesting til i et miljø der stor belastning ikke vil berøre produksjon og en kan etablere en kontrollert og forutsigbar setting med konfigurasjon og overvåking.

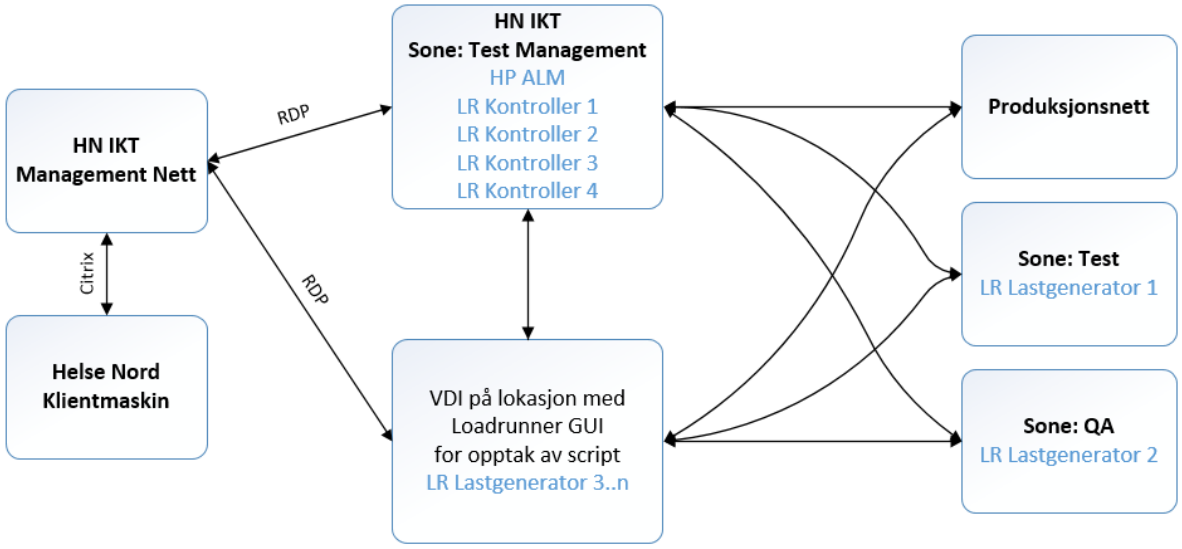
En vellykket ytelsestest før innføring eller oppgradering av et system vil da være å avdekke og fjerne flaskehalser i systemets interne komponenter med Stresstesting og Kapasitetstesting. Deretter kjøres en Lasttest fra lokasjoner for å verifisere at nettverk heller ikke er en flaskehals og at systemet fungerer som en helhet fra alle lokasjoner.

Før en ytelsestest foretas fra en lokasjon inn til sentralt system skal det gjennomføres en ROS der spesielt basistjenester og funksjonell forvaltning for systemet vil være sentrale. I tillegg skal det varsles i henhold til Helse Nord IKTs rutiner. Dette innebærer minimum:

- Legge ytelsestesten inn i vedlikeholdskalenderen
- Varsling på intranett dagen før og eller samme dag
- Varsle brukerstøtte og eventuelt IC for berørte tjenester

## 5.2 Overordnet beskrivelse av ytelsestesting i testmiljøene

Helse Nord skal i hovedsak bruke HP LoadRunner til ytelsestesting. Hvis oppgavene krever det, skal vi vurdere alternative verktøy og vi setter ingen begrensninger enten det gjelder verktøy bygget på åpen kildekode eller kommersielle verktøy.




**Figur 2: Oversikt over LoadRunner i Helse**

Figur 2 viser oversikt over hvordan LoadRunner er installert og konfigurert i Helse Nord. LoadRunner består i hovedsak av tre deler:

- **LoadRunner VuGen**. GUI-verktøy der en kan utvikle testskript basert på ren programmering eller en kombinasjon der en gjør opptak av arbeidsprosesser i et system som igjen genererer kode for eksekvering av tester. Koden fra slike opptak er kun en start, og koden må videre manipuleres og endres for gi en størst mulig grad av naturlig variasjon og tilfeldighet. Uten en slik variasjon og tilfeldighet vil en ikke kunne simulere en reell bruk og belastning på et system
- **LoadRunner Controller**. En kontroller er et GUI-verktøy der en henter inn testskript og orkestrerer en ytelsestest med antall virtuelle brukere, den belastning en ønsker å kjøre, varighet etc.
- **LoadRunner Load Generator**. Lastgenerator er en tjeneste som mottar arbeidsordre fra en kontroller for så å generere last mot systemet som skal ytelsestestes

Vi har i dag plassert ut på alle lokasjoner en vertsmaskin med VMware med VDI-maskiner for testing. Disse er virtuelle maskiner som tankes og vedlikeholdes av den lokasjonens Altiris og er for alle formål å betrakte som lokale klientmaskiner, men i motsetning til normale klientmaskiner kan disse nå både vårt Testsone og QA-sone. På denne måten kan en enkelt opprette en ny VDI-maskin på hvilken som helst lokasjon, installere VuGen og gjøre jobben med å lage testskript.

Vertsmaskinene kan i tillegg huse en virtuell Lastgenerator som kan brukes for å kjøre ytelsestesting fra lokasjonene. I slike tilfeller kan flere eller alle VDI-maskiner kjøres ned for å sikre nok ressurser til Lastgeneratoren.

# 6. YTELSESTESTPROSESSEN

I dette avsnitt beskrives Helse Nord IKT sin ytelsestestprosess med underliggende aktiviteter. Flere av aktivitetene i prosessen er iterative på den måten at det kan være behov for å gå tilbake til den samme aktiviteten flere ganger under ytelsestestgjennomføringen. Prosessen er også beskrevet i et prosessdiagram i Vedlegg E.


**Figur 3: Ytelsesprosessen**

## 6.1 Identifisere testomgivelsene

Det første steget i ytelsesprosessen omfatter aktiviteter knyttet til å identifisere omgivelsene for ytelsestesten. Det er viktig at testen begynner med disse aktivitetene da informasjonen samlet fra disse er grunnlag for de neste stegene prosessen. Det er vanlig at aktivitetene i dette steget må gjentas flere ganger under ytelsestesten når ytelsestestteamet lærer seg mer om applikasjonen, brukerne, funksjoner eller risikoer relatert til ytelse knyttet til disse.

Å identifisere testomgivelsene handler om å identifisere test- og produksjonsmiljøet samt de verktøy og ressurser som er tilgjengelig for testteamet. Å ha en grundig forståelse av testomgivelsene muliggjør å effektivere testdesign, planlegging og identifikasjon av utfordringer knyttet testen tidlig.

Spesifikke aktiviteter i dette steget inkluderer, men er ikke begrenset til aktivitetene under. De er listet i en numerisk rekkefølge, men ofte foregår de samtidig.

### 1 Etablere et ytelsestestteam

Det er essensielt å etablere et ytelsestestteam som skal utføre testen. Relevante deltakere i teamet kan være; systemforvaltere, databaseadministratorer, nettverksspesialister, systemansvarlige/superbrukere, arkitekter, testledere og prosjektledere.

### 2 Identifisere systemfunksjonene og/eller virksomhetsprosessene

Det er upraktisk og ikke realistisk å simulere alle mulige systemfunksjoner i en ytelsestest. Det er derfor nødvendig å bestemme de systemfunksjonene og/eller virksomhetsprosessene som er viktigst å ytelsesteste. Disse beskrives senere i scenarier som er nødvendige for i neste steg kunne etablere kriteriene for ytelsestesten samt for ytelsestesteren å forstå hensikten med systemet.

Når systemfunksjonene og/eller virksomhetsprosessene skal identifiseres bør følgende ligge til grunn:

- Kontraktsmessig forpliktende scenarier
- De systemfunksjonene eller virksomhetsprosessene som:
  - Er mest brukt
  - Er driftskritiske
  - Er ytelseskrevende
  - Der interessenter har uttrykt bekymringer

Grunnlag for å bestemme systemfunksjonene og/eller virksomhetsprosessene inkluderer blant annet intervjuer med interessenter, gjennomgang av kontrakter, designdokumenter, prosjektplaner, kravspesifikasjoner og brukerhistorier.

**3 Avklare testmiljø**

Under ideelle forhold, hvis målet er å definere ytelsesegenskapene til en applikasjon i produksjon, er testmiljøet en eksakt kopi av produksjonsmiljøet pluss lastgenerering- og ressursovervåkingsverktøy. Eksakte kopier av produksjonsmiljøer er uvanlig, så den viktigste faktoren blir da å forstå likheter og forskjeller mellom test- og produksjonsmiljøer.

**4 Identifisere testdatabehov**

For å simulere reelle forhold kan det kreves en spesifikk mengde og type data. Det er vanlig forekommende at testdata ikke kan gjenbrukes i ytelsestester grunnet caching på forskjellige nivåer. Det er derfor ofte behov for en stor mengde data. Det kan være utfordrende å etablere testdata, det er derfor viktig å begynne kartlegge behovet og hvis mulig etterspørre testdata, så tidlig som mulig.

**5 Identifisere arkitekturen**

Det er avgjørende at ytelsestesteren kjenner til både den logiske og den fysiske arkitekturen. Den logiske arkitekturen i denne sammenhengen er hvordan systemkomponentene (fysiske og virtuelle) henger sammen og hvordan de interagerer med hverandre. Den fysiske arkitekturen er hvordan aktuell hardware henger sammen. I arbeidet med å identifisere disse må ytelsestestere benytte seg av arkitekter, systemadministratorer og andre relevante tekniske ressurser for både test- og produksjonsmiljøet.

**6 Identifisere kritiske systemkomponenter**

Det er viktig for ytelsestesteren å vite om noen av systemkomponentene har kjente ytelsesutfordringer eller om det finnes integrasjonspunkter som er utenfor dennes kontroll under ytelsestesten.

## 6.2 Analysering og innsamling av ytelseskriterier

Det andre steget i ytelsesprosessen omfatter aktiviteter knyttet til innsamling av kriterier for ytelsestesten. Kriterier kan uttrykke seg i form av mål og/eller krav. Disse kriterier er mer spesifikke enn den overordnede hensikten med testen. I motsetning til mål er krav forhandlingsbare grunnet kontraktsforpliktelser og/eller virksomhetsbehov. Med utgangspunkt i de scenarier eller virksomhetsprosesser som er identifisert i det første steget, så etableres krav eller mål for disse. Eksempler på slike krav eller mål er hva brukere og eventuelt andre interessenter regner som god ytelse.

Andre ressurser som kan benyttes for å identifisere mål eller krav til ytelse er virksomhetskrav, brukerforventninger, kontraktsforpliktelser, industristandarder, SLA, baselines og optimaliseringsmål.

Normalt uttrykkes god ytelse i termene responstid, gjennomstrømming eller ressursforbruk. Kriterier er ofte beskrevet på en måte som ikke alltid er testbart fordi de ikke inneholder

kvantifiserbare verdier, så ytelsestestteamet må derfor kvantifisere kriteriene til noe som er testbart. Eksempler på kvantifiserbare kriterier er

- **Responstid:** Produktkatalogen skal vises på skjermen på mindre enn 3 sekunder.
- **Gjennomstrømming:** Systemet skal håndtere 25 bookinger per sekund.
- **Ressursforbruk**: Prosessorforbruket skal ikke overskride 75 %.

## 6.3 Testplanlegging og- design

Planlegging og utforming av ytelsestester innebærer å dokumentere utvalgte scenarioer, identifisere brukerspredningen, lage en lastmodell, etablere testdata, etablere en ytelsestestplan og utvikle testscriptene.

### 1. Dokumentere utvalgte scenarioer

De systemfunksjonene og/eller virksomhetsprosessene som ble identifisert i det første steget må nå dokumenteres. Dette kan gjøres ved å finne ut hvordan enkelte brukere faktisk gjennomfører oppgavene eller aktivitetene for systemfunksjonene og/eller virksomhetsprosessene. Det finnes flere metoder for å gjøre dette som å studere brukermanualer, observere brukere eller prøve å utføre de selv. Malen i Vedlegg C kan benyttes til å dokumentere scenarioene.

### 2. Identifisere brukerspredningen

I de fleste typer av ytelsestester velges ut en eller flere Peak Hour. Basert på hvordan bruksmønstret ser ut kan det finnes mer enn én Peak Hour. I en lasttest er det denne eller de timene som skal simuleres. Andre type tester tar utgangspunkt i denne timen når beregninger av ønsket last skal gjøres.

Ved å identifisere Peak Hour kan man finne ut to viktige ting; spredningen/forholdet mellom scenarioene og antallet aktive brukere i perioden. Dette er viktig for å simulere et virkelig scenario. Spredningen mellom scenarioene er hvor ofte brukerne utfører hvert scenario i forhold til de andre scenarioene, med andre ord det antall ganger som det totale antallet brukere utfører hver enkelt aktivitet.

Peak Hour kan identifiseres ved å hente data fra databaser eller logger, intervjue interessenter, innhente data fra en pilotgruppe eller som siste utvei bruke egen intuisjon.

### 3. Lage en lastmodell

Når brukerspredningen og antallet aktive brukere i tidsrommet som skal simuleres er identifisert må ventetider beregnes. Under en brukersesjon kan en brukere befinne seg i et antall forskjellige stadier f.eks. pålogging, søking, booking, avlogging osv. Alle brukere har forskjellige måter å interagere med applikasjonen og de bruker ulik tid på å navigere mellom de forskjellige funksjonene. Det er en avgjørende faktor å beregne ventetider for å skape en realistisk test. Konsekvensene av å se vekk fra ventetider i testplanleggingen er en urealistisk belastning som kan føre til skadelige unøyaktigheter som ofte fører til dårlige beslutninger.

Malen i Vedlegg B er laget for å automatisk regne ut ventetider for hvert enkelt scenario. Ved å fylle ut lengden på ytelsestesten (eksekveringstid), antallet aktive brukere i perioden og antallet eksekveringer per scenario regnes ventetiden ut automatisk.

### 4. Etablere testdata

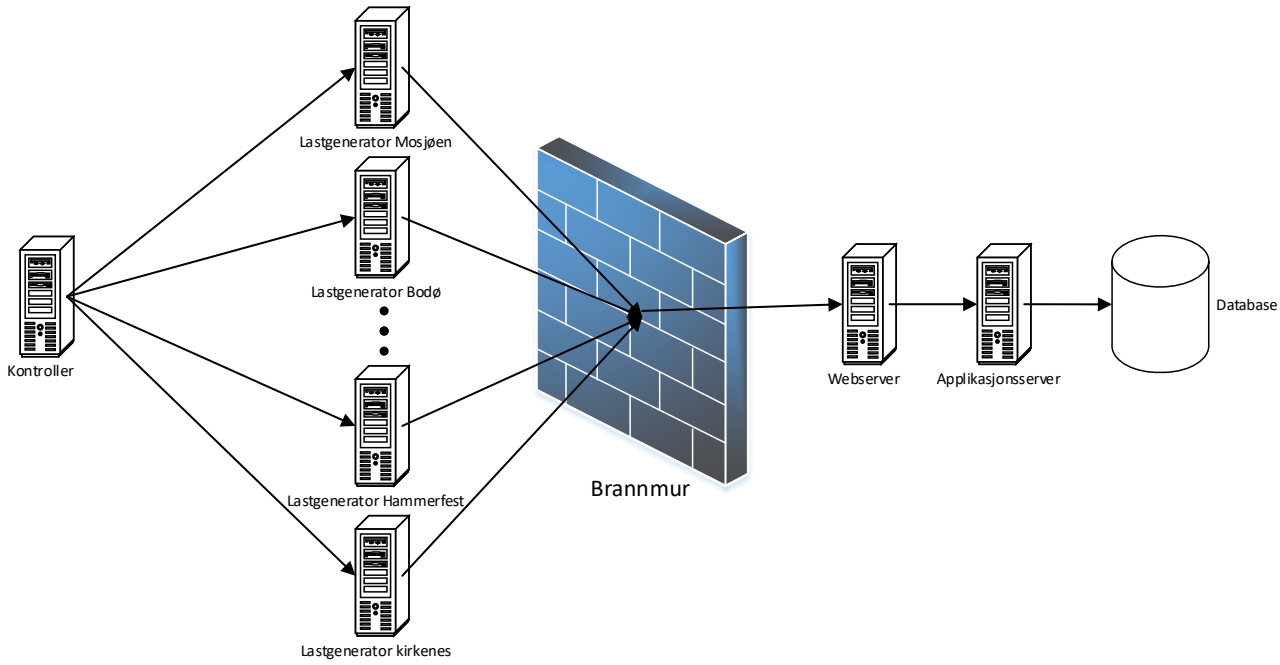
Testdata må nå etableres i det miljø der testen skal foregå. Det kan av og til gjøres ved å generere dette selv gjennom scripting, men i mange tilfeller er det behov for bistand fra forskjellige ressurser som for eksempel databaseadministratorer.

### 5. Konfigurere testmiljøet

Før testen kan eksekveres må testmiljø, verktøy og andre ressurser for testgjennomføringen klargjøres. Verifiser først at testmiljøet der testen skal foregå er tilgjengelig i perioden for testen, og at det er konfigurert i henhold til det som ble bestemt i det første steget i prosessen.

#### Lastgeneratorer

I noen tilfeller er det av ulike grunner ønskelig å generere lasten fra forskjellige geografiske plasser f.eks. for å identifisere forskjeller i svartider mellom lokasjoner. Det må da settes opp maskiner lastgeneratorer som kan generere lasten fra disse plassene. Dette er beskrevet i kapittel 4.




**Figur 4: Testmiljø oppsett med geografisk spredning av lastgeneratorer**

Lastgeneratorene må også konfigureres. Dersom HP LoadRunner benyttes må agenten til LoadRunner installeres og settes opp til å kjøres med en servicebruker. Det kan også være behov av brannmursåpninger eller installasjon av forskjellig programvare.

Dersom en stor belastning skal genereres, kan det være nødvendig å verifisere at lastgeneratorene tåler belastningen som er planlagt. Generer belastningen og overvåk CPU- og minneforbruk, og hvis ressursforbruket er for høyt blir lastgeneratoren en flaskehals under ytelsestesten og resultatet blir misvisende. Det er også viktig å verifisere før testen at det ikke

er noen bakgrunnsprosesser på disse maskinene som kan påvirke ytelsestesten som f.eks. et antivirusprogram.

**Monitorering og datainnsamling**

Det er avgjørende å kunne monitorere og samle inn ytelsesdata på de komponentene som belastes under testen. I HP LoadRunner finnes det allerede innebygde verktøy for monitorering og innsamling av data, men ofte vil det være behov for å overvåke andre ting medføre installasjon av andre verktøy. I andre tilfeller kan leverandøren tilby overvåkingsverktøy. Slike verktøy må da konfigureres sammen med leverandør og andre relevante ressurser, f.eks. systemforvalter.

Hvilke ytelsesdata som skal samles inn, bestemmes i hovedsak av ytelseskriteriene som skal oppfylles. For å verifisere oppnåelse av ytelseskriteriene kan systemforvalter, databaseadministrator, nettverksspesialist og/eller andre tekniske ressurser ha ønsker om andre parametere som skal overvåkes under testen.

**6. Lage ytelsestestplan**

Resultatet av de tidligere aktivitetene i prosessen dokumenteres i en testplan. Helse Nord IKT sin mal for testplan skal benyttes. Den skal godkjennes av bestiller av oppdraget før igangsetting av ytelsestesten. Testplanen skal blant annet inneholde; hensikt med testen, mål og kriterier, kort beskrivelse av scenarioene og lastmodell.

**7. Utvikle testscript**

Denne aktiviteten foregår ofte parallelt med de andre aktivitetene i prosessen da det ofte er vanskelig å estimere hvor lang tid denne aktiviteten tar. I så stor grad som mulig skal ytelsestestverktøyet HP LoadRunner benyttes til utviklingen av testscriptene. Denne aktiviteten innefatter blant annet; parametrisering og korrelering av scriptene, etablering av transaksjoner og smoke test av scriptene. HP LoadRunner har innebygd kobling mot HP ALM og testscript skal i hovedsak lagres i prosjekt i HP ALM for versjonskontroll og gjenbruk.

Utvikling av testscript krever i noen tilfeller, spesielt når proprietære protokoller benyttes, et tett samarbeid med utviklere som er kjent med applikasjonen.

## 6.4 Eksekvering

I forbindelse med eksekveringen av testen er det en hel del aktiviteter som foregår.

**1. Konfigurere kontrolleren**

Et avgjørende moment for ytelsestesten er å konfigurere kontrolleren riktig for å simulere en realistisk belastning. Feilkonfigurering av kontrolleren vil innebære en feil belastningen og vil gi et misvisende testresultat. Oppsettet i kontrolleren skal gjenspille lastmodellen i forhold til hvilke scenarioer som skal kjøres, antall brukere, ventetider, brukerspredning og lengde på testen. Utover konfigurasjon av nevnte ting må også egenskaper for logging, ramp-up og ramp-down konfigureres.

Hvis ikke et separat produkt benyttes for monitorering må det innebygde monitoreringsverktøyet i HP LoadRunner konfigureres. En konfigurasjon av kontrolleren og monitorering utgjør et scenario og alle slike scenariene skal arkiveres i HP ALM for gjenbruk.

**2. Smoke test**

Det er lurt å kjøre en smoke test før testen begynner for å verifisere at ting fungerer som forventet. I noen tilfeller kan en smoke test kjøres for å fylle buffer før hovedtesten.

**3. Varsle interessenter**

I god tid føre eksekveringen av ytelsestesten bør relevante grupperinger varsles. Disse bør så varsles igjen samme dag eller litt før testeksekveringen.

**4. Observasjon**

Utover monitorering av applikasjonen, systemet eller komponentene som er utsatt for testen bør også testspesifikke ting observeres for å sikre en vellykket testgjennomføring. CPU- og minneforbruk på samtlige lastgeneratorer bør observeres under hele kjøringen for å sikre at disse ikke er overbelastet. Det kan også være hensiktsmessig å observere eventuelle feilsituasjoner som kan oppstå under kjøringen, at samtlige virtuelle brukere kjører normalt og at testen genererer en belastning.

Det er også fornuftig, hvis mulig, å bruke systemet under testen for senere kunne sammenligne brukeropplevelsen med testresultatet.

**5. Kjør testen flere ganger**

Det er ikke uvanlig at uforutsigbare ting eller tilfeldigheter som påvirker testresultatet oppstår under testen. Hvis det er mulig bør testen kjøres to ganger for å være sikker på at resultatet stemmer. Hvis resultatet av kjøringene ikke er enhetlige bør årsaken identifiseres før ytterligere kjøring starter.

**6. Validere testresultatet**

Det er lett å stole på at testen har blitt gjennomført riktig. Men før videre analyse og presentasjon av testresultatet gjøres, er det viktig å validere resultatet. Verifiser at brukerspredningen er riktig slik at det ikke oppstår en belastning som den vist i Figur 3. Verifiser at det er påført belastning på systemet f.eks. ved å studere loggfiler. Dobbeltsjekk at riktige testdata har blitt brukt.

**7. Baseline**

Dersom en skal gjøre endringer i et system i form av ytelsesforbedrende tiltak eller endringer og oppgradering av et system er det hensiktsmessig å lage en baselike. Testscenarioer, script, resultat og overvåkingsdata fra LoadRunner lagres i HP ALM som en baseline, slik at en i etterkant kan reteste for å se om det er en endring i ytelse.

## 6.5 Resultatanalyse og rapportering

Når ytelsestesten er gjennomført må testresultatet sammenstilles, analyseres og tilslutt rapporteres. Helse Nord IKT sin mal for ytelsesteststluttrapporter skal benyttes for rapportering av ytelsestestresultatet. Rådata, sammenstillinger og grafer arkiveres i HP ALM, men kan også tilsendes ved etterspørsel.

**1. Behandle data og lage grafer**

Dersom HP LoadRunner har blitt benyttet for gjennomføringen av testen er rådata fra lastgeneratorene tilgjengelig i analysekomponenten i LoadRunner. Før videre analyse av resultatet kan foretas må data behandles og grafer lages. En visualisering av testresultatet i form av grafer forenkler arbeidet med å identifisere eventuelle ytelsesutfordringer samt avlesing av oppnåelse av ytelseskriterier.

Behandlingen av data foregår i hovedsak gjennom filtrering. Det er for eksempel ønskelig å filtrere vekk ytelsesdata fra ramp-up og ramp-down periodene da disse kan være misvisende. Forskjellige interessenter er interessert i forskjellige type grafer og data og det innebærer at forskjellige filter og grafer må tilpasses ulike interessenter. Dette er derfor en iterativ prosess.

2. **Analyse av testresultatet**

Når datagrunnlaget og grafene er på plass kan disse analyseres. Dette er en aktivitet som er fornuftig å gjøre både individuelt og i et team. Forskjellige ressurser kan se ting på forskjellige måter. Resultatet sammenlignes her med ytelseskriteriene for å fastslå måloppnåelsen av disse.

For å oppdage ytelsesproblemer er det ofte nødvendig å analysere flere forskjellige grafer, og ofte kan det være hensiktsmessig å konsolidere flere forskjellige grafer. Det blir da lettere å peke ut mønster. Hvis testen er kjørt flere ganger for eksempel med forskjellig antall brukere kan også resultatet fra disse være fornuftig å konsolidere for å se hvordan ressursforbruk eller responstider endres med økt antall brukere.

Dersom ytelsestesten er kjørt på nytt for eksempel etter et ytelsesforbedringstiltak eller etter en oppgradering, bør data sammenstilles slik at de kan presenteres i en trend-graf. En trend-graf er en visualisering av hvilken effekt forandringene på applikasjonen, systemet eller komponenten har hatt på ytelsen. Det kan brukes til å finne ut om et ytelsesforbedrende tiltak har hatt positiv eller negativ effekt på ytelsen.

**3. Rapportering**

Når analysen er foretatt skal resultatet fra testen rapporteres. Husk at forskjellige målgrupper har forskjellige behov for rapportene som derfor må tilpasses målgruppen. Utover den generelle sluttrapporten som sendes ut til bestiller av oppdraget, så kan mindre og tilpassete rapporter lages og sendes andre interessenter.

Ved rapportering av responstider er ofte ytelseskriteriene formulert som gjennomsnittlig responstid. Problemet med gjennomsnittlig responstid i forbindelse med ytelsestest er at det veldig ofte forekommer uforutsigbare hendelser som gjør at responstiden i enkelte tilfeller blir unormalt høy. I de fleste tilfeller er det nærmest umulig å finne ut årsaken. Siden disse unormalt høye responstider ofte i stor grad påvirker den gjennomsnittlige responstiden er det vanlig å ta vekk tilfeldige ekstremverdier.

# 7. VEDLEGG

## 7.1 Vedlegg A: Eksempel på dokumentasjon av brukerscenarier

- Beskriv virksomhetsprosessen som skal ytelsestestes steg for steg.
- Testdata er kun nødvendig når dette må genereres.
- Transaksjonsnavn fylles ut av utvikler av ytelsestesten.

**MERK!** Dette er kun et eksempel, benytt malen som ligger her: (Sett inn link til malen)

### Virksomhetsprosess: Nytt avvik

| | Steg | Testdata nødvendig? | Transaksjonsnavn |
|---|---|---|---|
| 1 | Åpne Internet Explorer og gå til websiden: http://alm.hn.helsenord.no | Nei | |
| 2 | Log Inn | Ja | LogIn |
| 3 | Velg domene og prosjekt | Ja | |
| 4 | Åpne Defect Modulen | Nei | |
| 5 | Registrer ny Defect | Ja | |
| 6 | Klikk Lagra | Nei | NewDefect |
| 7 | | | |
| 8 | | | |
| 9 | | | |

## 7.2 Vedlegg B: Eksempel på lastmodell

HVA: Mal for å regne ut målverdiene samt konfigurasjonstall til oppsett av ytelsestesten.

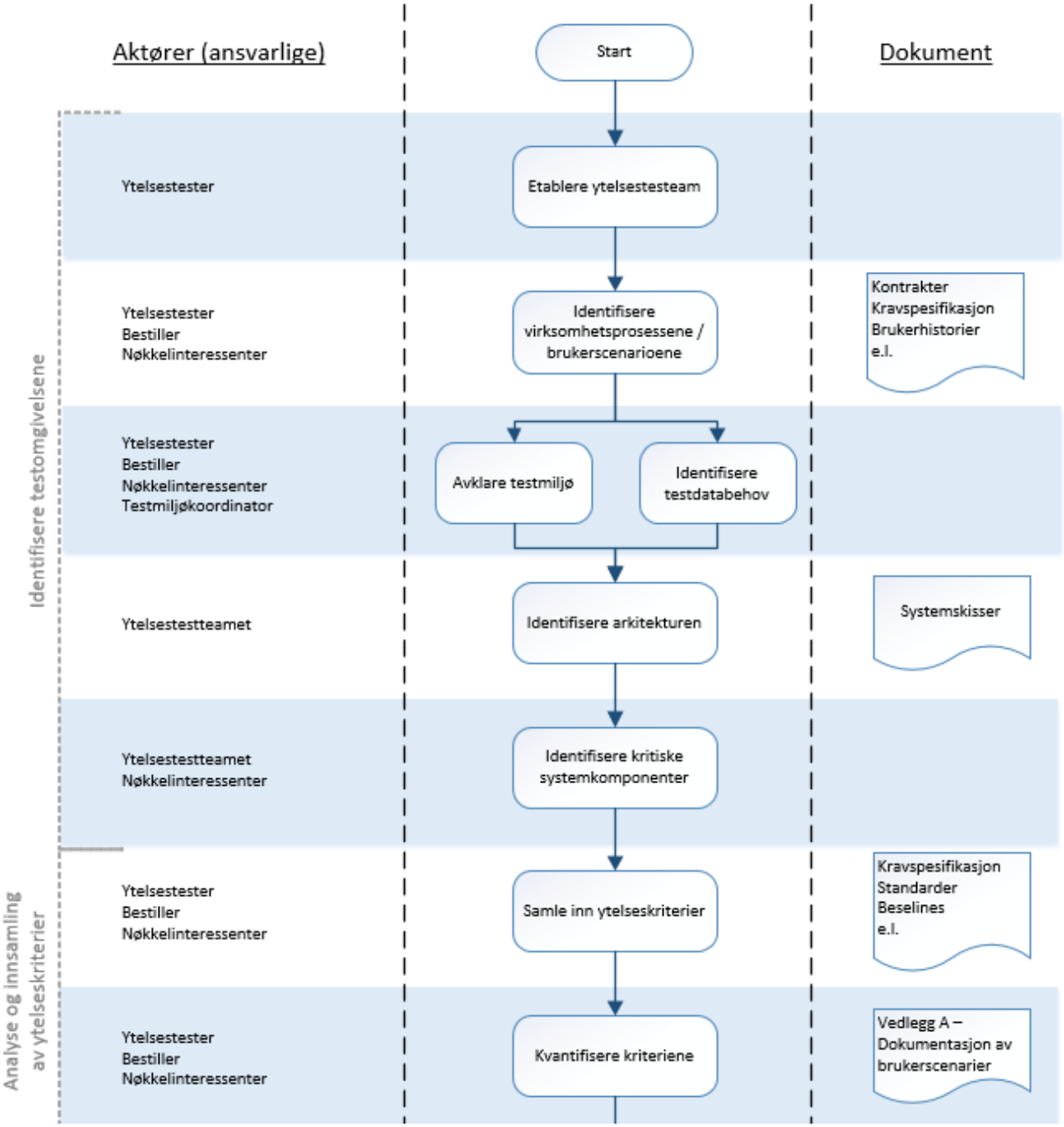
HVORFOR: For å sikre en god ytelsestest med ønsket belastning.

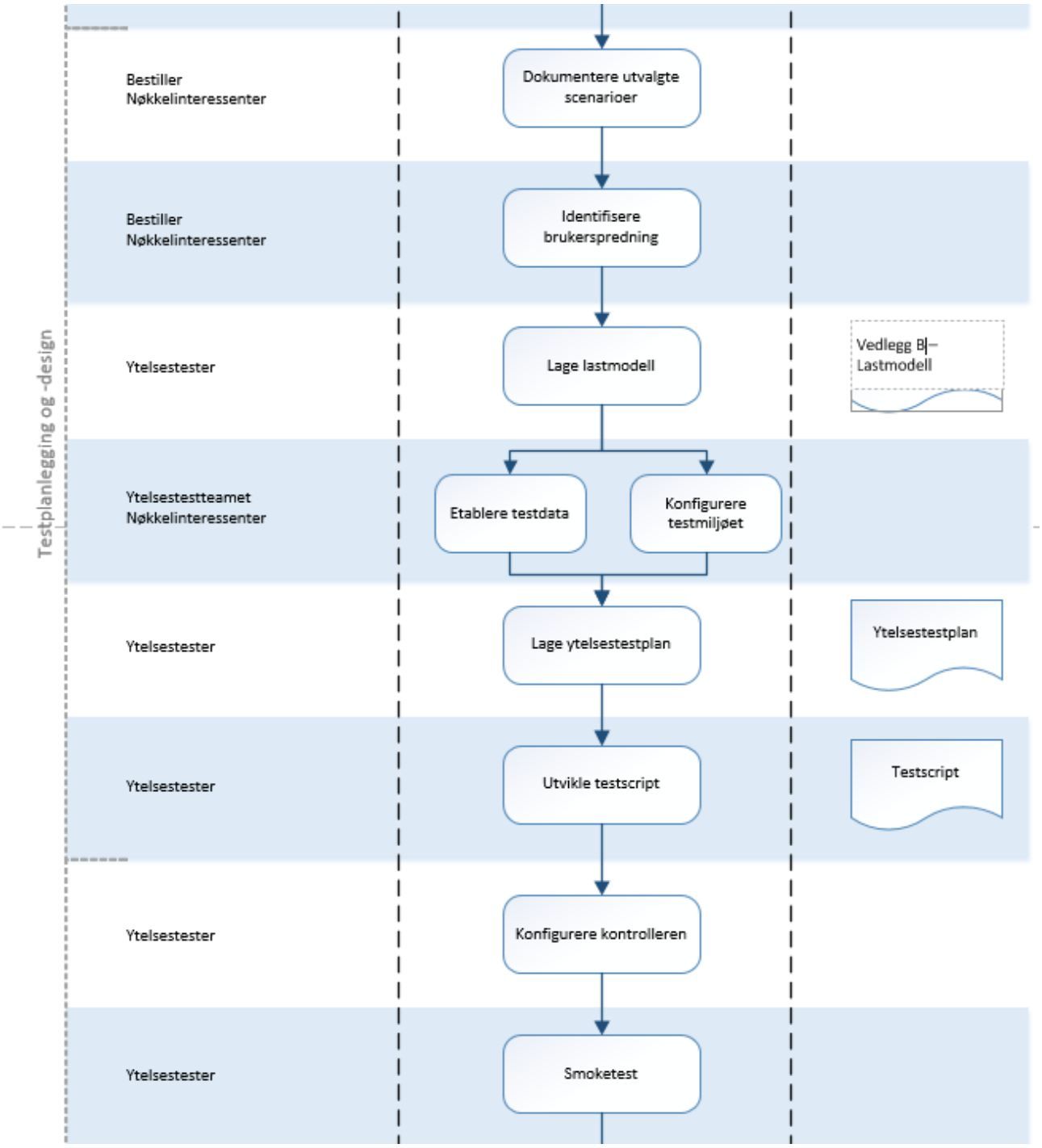
HVORDAN: Fyll ut applikasjonssamtidigheten dvs. antallet aktive brukere på systemet under peak-hour, hvilke virksomhetsprosesser som er de vanligste for disse og hvor mange gangere de er gjennomført i perioden. Kun grønne felter fylles ut.

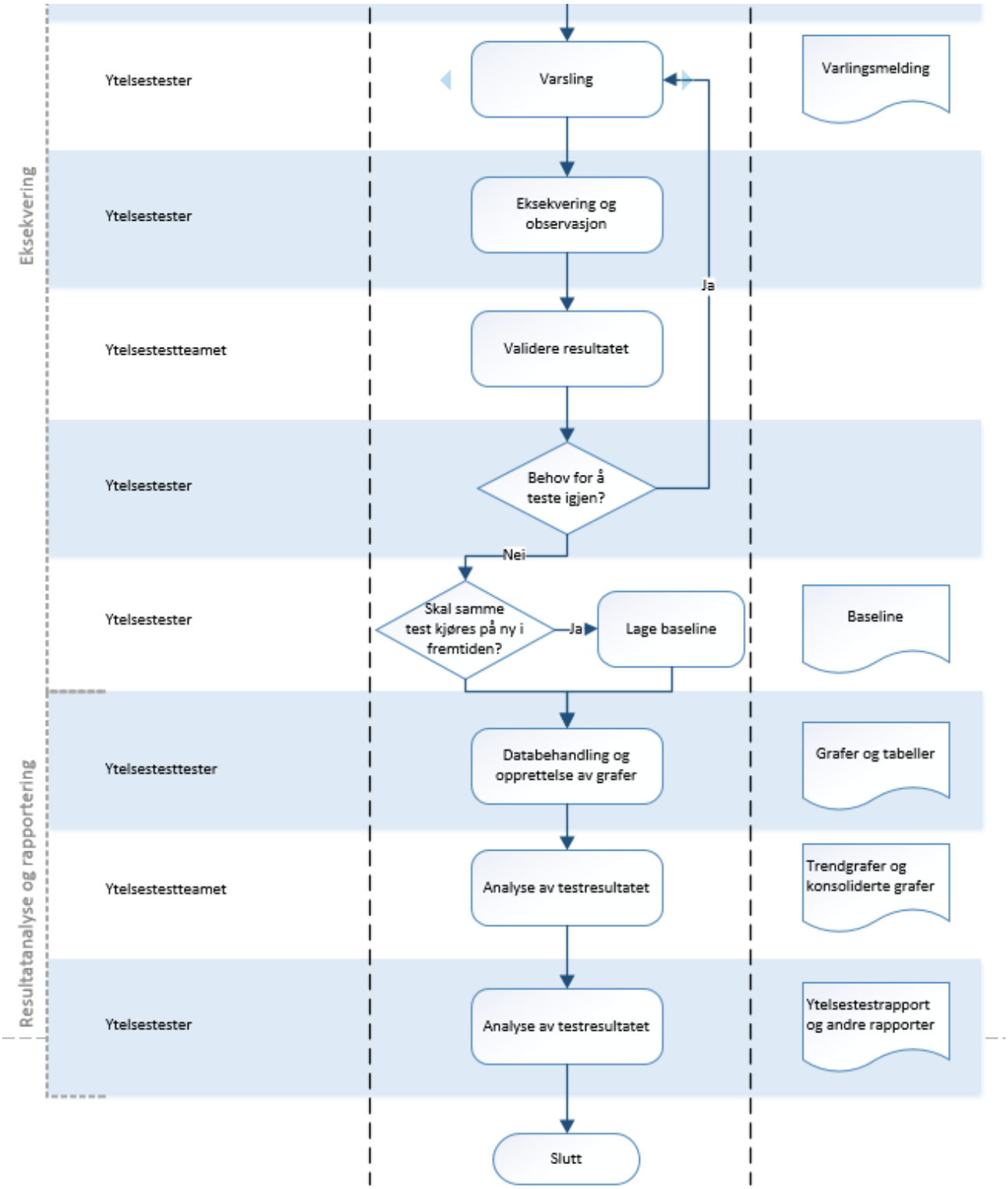
| | |
|---|---|
| **Aktive brukere:** | 50 |
| **Eksekveringstid (sek):** | 720 |

| | | | | | | Ventetid (sek) | |
|---|---|---|---|---|---|---|---|
| # | Scenario | Antall eksekveringer | % | Bruker-spredning | Iterasjoner | Fra | Til |
| 1 | Søke | 150 | 42,86 % | 21,4 | 7,0 | 1 | 205 |
| 2 | Booke | 200 | 57,14 % | 28,6 | 7,0 | 1 | 205 |
| 3 | | | 0,00 % | 0,0 | #DIV/0! | | #DIV/0! |
| 4 | | | 0,00 % | 0,0 | #DIV/0! | | #DIV/0! |
| | **SUM:** | **350** | | **50** | | | |

## 7.3 Vedlegg C: Prosessdiagram for ytelsestestprosessen




Testplanlegging og -design
Bestiller
Nøkkelinteressenter
Dokumentere utvalgte scenarioer
Bestiller
Nøkkelinteressenter
Identifisere brukerspredning
Ytelsestester
Lage lastmodell
Vedlegg B – Lastmodell
Ytelsestestteamet
Nøkkelinteressenter
Etablere testdata
Konfigurere testmiljøet
Ytelsestester
Lage ytelsestestplan
Ytelsestestplan
Ytelsestester
Utvikle testscript
Testscript
Ytelsestester
Konfigurere kontrolleren
Ytelsestester
Smoketest


Eksekvering
Resultatanalyse og rapportering
Ytelsestester
Ytelsestester
Ytelsestestteamet
Ytelsestester
Ytelsestester
Ytelsestesttester
Ytelsestestteamet
Ytelsestester
Varsling
Eksekvering og observasjon
Validere resultatet
Behov for å teste igjen?
Ja
Nei
Skal samme test kjøres på ny i fremtiden?
Ja
Lage baseline
Databehandling og opprettelse av grafer
Analyse av testresultatet
Analyse av testresultatet
Slutt
Varlingsmelding
Baseline
Grafer og tabeller
Trendgrafer og konsoliderte grafer
Ytelsestestrapport og andre rapporter