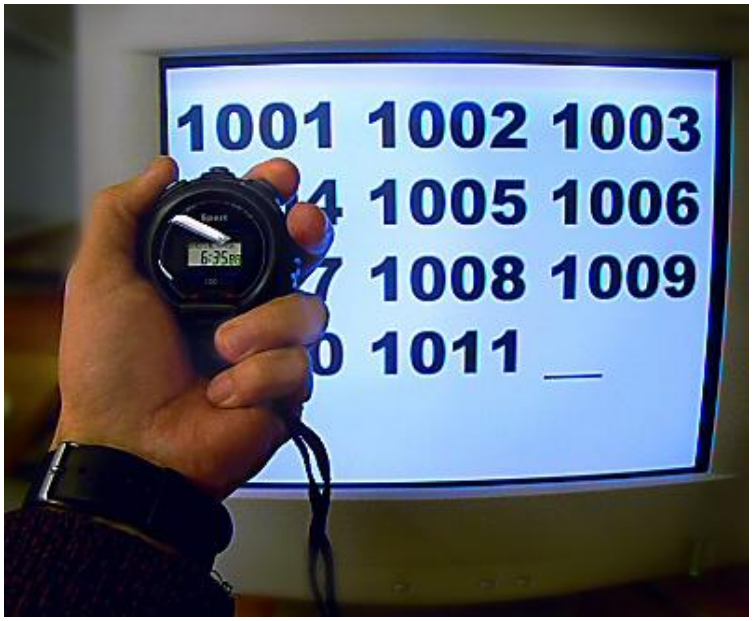
Suorituskykytestaus / kuormitustestaus



Kalvosarja esittelee suorituskyky- / kuormitustestauksen keskeisiä periaatteita ja käytännön menettelyjä. Kontekstina on lähinnä organisaation tietojärjestelmän testaus.

UPDATE 2010

Matti Vuori, www.mattivuori.net

Sisällysluettelo 1/4

Sisällysluettelo 2/4

Sisällysluettelo 3/4

Sisällysluettelo 4/4

Yleiskuvaus

- Selvitetään järjestelmän kyky suoriutua tehtävästään vaaditussa ajassa kuormitettuna.
- Testataan, että ohjelmisto täyttää sen suorituskykyvaatimukset:
  - Käyttäjämäärät.
  - Datamäärät.
  - Vasteajat.
  - Jne...
- Järjestelmänäkökulma – mahdollisimman paljon ulkoisia mittauksia järjestelmän läpi. Mutta myös mittauksia konepellin alla.

Suorituskykytestauksen syyt ja edut

- Suorituskyky on kriittinen menestystekijä esimerkiksi verkkopalveluilla, joille odotetaan suuria käyttäjämääriä.
- Sovellusten nopeus on tärkeä käyttäjätyytyväisyystekijä.
- Monissa työprosesseissa nopeus tärkeää tai kriittistä.
- Systemaattinen suorituskykytestaus paljastaa järjestelmän todellisen suorituskyvyn ja sen, miten järjestelmä skaalautuu lähitulevaisuudessa uusille käyttäjämäärätasoille.
- Se on siis testityyppi ja samalla keino ”ostaa aikaa”.

Konkreettiset tavoitteet

- Suorituskykyvaatimusten verifiointi.
  - Suorituskyvyn selvittäminen.
  - Kuormituksen siedon selvittäminen.
- Varmistaa, että käyttöönotto onnistuu suunnitellusti – että palvelu ylipäätään toimii.
- Pullonkaulojen tunnistaminen.
- Selvittää laitteistotarve – riittävyys nyt ja lähitulevaisuudessa.

Suorituskykytestauksen alalajit 1/3

- Kuormitustestaus (load testing)
  - Selvitetään palvelun kuormitettavuusominaisuudet – maksimikapasiteetti, lyhyen ajan käyttäytyminen kuormitettuna jne.
  - Tunnistetaan ongelmat ja mahdollisuuksien mukaan niiden syyt.
  - Suunnitelmalähtöistä, perustuu vaatimusmäärittelyyn.

Suorituskykytestauksen alalajit 2/3

- Stabiliteettitestaus (stability testing)
  - Selvitetään palvelun toiminta pitkän käyttöjakson aikana sopivalla kuormalla.
  - Monitoroidaan palvelua ja tunnistetaan ongelmia toiminnassa, logeissa.
  - Kesto vaihtelee 15 min ... 24 h ... viikko ... kuukausi.
  - Mitataan resurssien käyttöä, vasteaikojen kehittymistä jne...
  - Suunnitelmalähtöistä, perustuu vaatimusmäärittelyyn.

Suorituskykytestauksen alalajit 3/3

- Isolaatiotesti (isolation testing)
  - Korkean tason testit paljastavat ongelmien olemassaolon, mutta eivät niiden syitä.
  - Syiden tunnistamiseen tarvitaan testejä, jossa ongelmaan liittyvät ilmiöt eristetään ja pyritään löytämään todellinen syy ja myös analysoimaan eri tekijöiden vuorovaikutuksia.
  - Luonteeltaan tutkivaa, ketterää testausta.
- Testausten jälkeen: Käytönvalvonta
  - Testauksessa syntyvän monitorointiymmärryksen avulla on hyvä kehittää tuotantokäyttöön otettavan palvelun monitorointia ja täydentää kenties vielä lopullisesti verifioimattomia suorituskykymalleja.

Testaus vastaa konkreettisiin asiakkaan tiedontarpeisiin

- Täyttääkö toteutettava järjestelmä loppukäyttäjien vaatimukset?
- Pärjätäänkö ruuhkahuipuissa?
- Täyttääkö toteutettava järjestelmä vaatimukset suunnitellulla laitteistolla?
- Tarjoaako järjestelmä riittävästi kasvunvaraa seuraavan vuoden tarpeisiin?
- Mikä on lisäkapasiteetin tarve?
- Montako palvelinta tarvittaisiin tarvittavan kapasiteetin toimittamiseksi?

Loppukäyttäjän näkökulma oleellinen

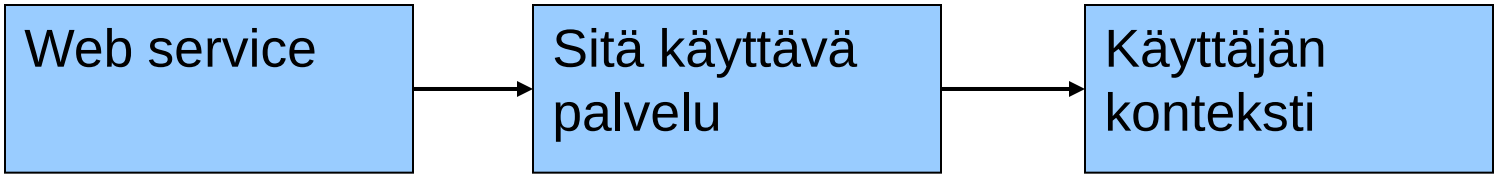
- Suorituskyvystä nauttii aina loppukäyttäjä.
- Loppukäyttäjälle on olennaista koettu palvelutaso.
  - Vasteajat.
  - Yhteyden saaminen palveluun.
  - Palvelun ylikuormittumisesta aiheutuvat virheet.
  - Palvelun toimiminen aina, kun yksilö niin haluaa.
- => Ymmärrettävä palvelun käyttö
  - Palvelun tyyppi.
  - Ketkä sitä käyttävät ja millaisissa tilanteissa.
  - Käyttäjien tarpeet – tavoitteet, preferenssit, paineet, kriittiset asiat.

Ymmärrettävä palvelun käyttäjät ja käyttötavat

- Palvelun tyyppi.
- Ketkä sitä käyttävät ja millaisissa tilanteissa.
- Käyttötarkoitus.
- Käyttötapaukset.
- Käyttöympäristö.
  - Ml. tietoliikenneyhteydet.
- Käyttäjien tarpeet
  - Tavoitteet.
  - Preferenssit.
  - Paineet (aikapaineet).
  - Kriittiset asiat.

Kokonaisuuksien huomioon ottaminen

- Palvelupohjaisissa järjestelmissä näkökulma voi jäädä oman palvelun suorituskykyyn.
  - Miten hyvin omalta palvelimelta saadaan dataa ja se välitetään eteenpäin.
- Oleellista on tunnistaa ne tavat, joilla loppukäyttäjien palvelua tarjotaan.
  - Miten loppukäyttäjät käyttävät dataa? Mitkä ovat siinä kontekstissa suorituskykyvaatimukset?



Palvelun tarjoajan näkökulma

- Käyttäjiä on monta.
- Eri käyttäjäryhmillä on erilainen prioriteetti.
  - Erilainen tärkeys asiakkaana. Kassavirta, asema.
- Käyttäjäryhmät käyttävät eri palveluja.
- Eri palveluilta edellytettävä palvelutaso on erilainen.
- Tärkeimmät palvelut kriittisiä.

Tulevaisuusnäkökulma

- Testauksessa on tärkeää tuottaa tietoa tulevaisuudesta.
- On ennakoitava järjestelmän tulevaa käyttöä.
  - Käyttötapaukset tulevaisuudessa.
  - Muutokset.
- Siksi ei testattavissa asioissa pidä luottaa liikaa historiatietoihin (esimerkiksi palvelimen logitiedostot), vaan on analysoitava tulevia tilanteita.

Ajoittuminen ohjelmistoprojektissa 1/2

- Tarpeen tehdä kaikissa ohjelmistoprojektin vaiheissa.
- Kriittiset suorituskykytekijät (esim. uudet tietoliikennetekniikat) testataan testipenkissä jo konseptin verifiointivaiheessa.
  - Oleellista eristää mahdolliset ongelma-alueet siten, että ne voidaan testata.
  - Esimerkiksi pelkän tietokannan testaus ei edellytä koko testausjärjestelmää.
  - Tällöin voidaan arkkitehtuuria muuttaa testien perusteella, tai muuttaa järjestelmävaatimuksia (lisää rautaa...).
- Varhainen testaus auttaa ymmärtämään monimutkaisen arkkitehtuurin suorituskykyyn vaikuttavia tekijöitä ja eri komponenttien käyttäytymistä.
- Näkökulma on yhtä paljon laadullinen kuin määrällinen.

Ajoittuminen ohjelmistoprojektissa 2/2

- Järjestelmätestauksessa tehdään lopulliset suorituskykytestit.
  - Järjestelmätestien lopussa, kun isoimmat bugit on saatu pois.
  - On oltava varmuus siitä, että viat johtuvat kuormasta, eikä ”tavallisista” bugeista.
- Esim. pankkialalla on usein pakollinen vaatimus, että kuormitustestit on ajettu ennen tuotantoon siirtoa ja että kaikki osapuolet hyväksyvät ne.

Ohjelmistohankintojen osana

- On tärkeää todentaa, että uusi järjestelmä oikeasti pystyy tarvittavaan kapasiteettiin.
  - Toimittajan lupauksiin ei pidä luottaa.
  - Toimittajan on varmistettava, että lopputulos on asiakkaalle riittävän ripeä.
- Kuormitustestaus on usein osa asiakkaan tekemää hyväksymistestausta
- Julkishallinnon hankkeissa koko ajan yleisempi
  - Mainittu myös valtionhallinnon JHS-suosituksen 129 liitteessä 3).

Muutosten testauksessa

- Tietojärjestelmien muutoksen testauksessa on kuormitustestaus usein yksi testausvaihe.
- Esim. pankkialalla on usein pakollinen vaatimus, että kuormitustestit on ajettu uudelle versiolle ennen tuotantoon siirtoa ja että kaikki osapuolet hyväksyvät ne.
  - Tämä merkitsee sitä, että kaikki muutetut versiot käyvät läpi tietyn prosessin.
  - Olipa muutos miten pieni tahansa, kuormitustestaus on osa uuden version käyttöönottoprosessia.

Kunnonvalvonnassa

- Säännöllisesti tehtävä kuormitustestaus on osa tietojärjestelmän kunnonvalvontaa.
  - Varmistaa, että suorituskyky on riittävä.
  - Auttaa tunnistamaan ongelmia tietokannoissa ja palvelimilla.
- Kuormitustestausta käytetään myös tuotanto-ongelmien diagnosointiin
  - Simuloidaan ongelmatilannetta.
  - Tämä onnistuu organisaatiossa, jossa kuormitustestaus on säännönmukaista ja järjestelyt valmiina.

Lähestymistapa

- Automatisoitu testaus testausohjelmilla.
  - Luodaan kuormaa.
  - Mitataan vasteaikoja.
- Virtuaalisten käyttäjien hyödyntäminen todellisuutta simuloivassa testausjärjestelyssä.
  - Suurten organisaatioiden sisäisissä projekteissa voidaan käyttää myös todellisia käyttäjiä.
- Testauksen peruslähtökohta on suunnitelmalähtöinen, mutta tarvitaan myös tutkivan testauksen elementti.

Suhde muihin testeihin

- Toiminnallisuustestaus testaa toimintojen toimimisen oikein, sekä suorituskyvyn normaaleissa käyttötilanteissa.
  - Läpäisty toiminnallisuustestaus on edellytys suorituskykytestaukselle.
- Myös käytettävyystestaus tarkastelee järjestelmän vasteaikoja:
  - Kattaen käyttäjän toimet laajemmin.
  - Ottaen vähemmän systemaattisesti huomioon järjestelmän.
- Tietoturvatestauksen kannalta on oleellista todentaa hallittu toiminta valtavissa käyttövoluumin purskeissa (palvelunestohyökkäykset). Tämä testaus tapahtuu kuormitustestauksessa.

Tuotantoympäristön ymmärtäminen oleellista

- Miten ympäristö toimii?
- Mitä on konepellin alla?
  - Järjestelmän elementit.
  - Laitteisto, käyttöjärjestelmät, ohjelmistot, versiot.
- Mistä suorituskyky riippuu. Mitkä ovat pullonkaulat?
- Mitä muita prosesseja ympäristössä tapahtuu? Mitä muuta tietoliikennettä tapahtuu?
- Millaisella teholla tuotantojärjestelmää voi kuormittaa – koskaan ei voida käyttää täydellä teholla. Tämä heijastuu mitoitukseen, testaukseen ja tulosten tulkintaan.

Tekijät

- Suorituskykytestaukseen erikoistuneet testaajat.
- Kiinteä yhteistyö ohjelmistokehityksen kanssa.
- Yleiset osaamistarpeet:
  - Testausosaaminen.
  - Analysointitaito. Kyky eristää ongelmia ja analysoida suorituskykyyn vaikuttavia tekijöitä.
  - Ymmärrys järjestelmä- ja ohjelmistoarkkitehtuureista.
  - Tietoliikennetekniikan ja tietoverkkojen ymmärrys.
- Tapauskohtaisesti
  - Testattavan sovellustyypin ymmärrys.
  - Käyttöjärjestelmä- ja laitteisto-osaaminen.
  - Tietokantaosaaminen.
  - Ohjelmointiosaaminen.

Ympäristö

- Integrointi- tai järjestelmätestausympäristö.
- Asiakkaan testausjärjestelmä.
- Testipenkki.
- Ympäristö riippuu testauksen tavoitteista ja siitä, miten testattava asia kyetään eristämään kokonaisuudesta.
- Puuttuville järjestelmän osille rakennetaan korvike – usein prototyyppi.

Perusprosessi

- Tunnistetaan alustavasti palvelun pullonkaulat.
- Suunnitellaan testaus, ml. testausjärjestelmän määrittely.
- Nauhoitetaan tai ohjelmoidaan jokin operaatio.
- Toistetaan se sopivalla määrällä virtuaalisia käyttäjiä.
- Mitataan aikoja ja järjestelmää suorituksen aikana.
- Analysoidaan tulokset.

Kuormitustestausohjelmat 1/3

- Ohjelmia, joilla järjestelmää kuormitetaan esimerkiksi suurella määrällä tietokantakyselyjä.
- Ohjelmistot, jotka simuloivat suuria määriä käyttäjiä.
- Nykyisin keskeinen sovellusalue selainohjelmien testaus.
- Ohjelmia on erilaisia – valinta tarpeiden mukaan.
- Esimerkkejä yleisesti käytetyistä ohjelmista:
  - Mercury LoadRunner. Kallis.
  - Java-pohjainen JMeter on yleinen avoimen lähdekoodin sovellus. Helppo jakaa kaikille testaajille, asiakkaille ja alihankkijoille.
  - Compuware QALoad.
  - Microsoftin ilmainen Web Application Stress Tool, joka lähinnä nauhoittaa http-käskyjä. Ohjelman käyttö on vähentynyt parempien ohjelmien tullessa markkinoille.

Kuormitustestausohjelmat 2/3

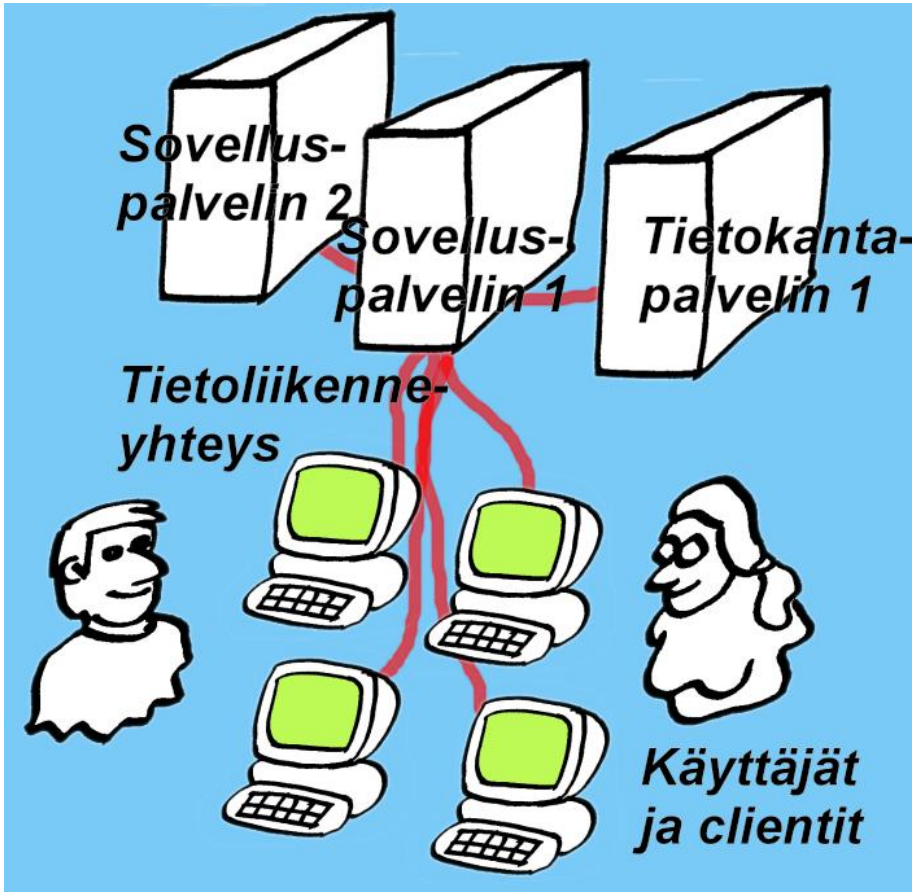
- eValid. Perustuu Internet Explorer -yhteensopivaan WWW-selainohjelmistoon, jonka nauhoittama oliomalliin perustuva skriptikoodi on testaajan täysin hyödynnettävissä.
- Apache Bench on käytössä Apache http-palvelimen suorituskyvyn testaukseen – oleellinen komponentti monissa järjestelmissä.

Kuormitustestausohjelmat 3/3

- Java-sovellusten koodipohjaisessa testauksessa käytetään usein esimerkiksi httpUnitin päälle rakennettua testausohjelmaa.
  - Ideana on se, että yksikkötestausohjelmalla JUnit huolehditaan testien hallinnasta ja suorittamisesta ja httpUnit:n avulla suoritetaan http-kyselyt ja tulosten vertailu.
  - Keskeinen etu on rajoittamattomat mahdollisuudet testauksen
- Palvelimen päässä kuormitustestauksessa käytetään erilaisia monitorointiohjelmia, joiden avulla kapasiteetin tarve osataan yhdistää kuormitustilanteisiin.
- Pitkä lista ohjelmistoja: (Huom! Linkki ei toimi 05/2007) http://www.performancetester.com/component/option,com_weblinks/Itemid,4/catid,68/

Testauksen kohteen stereotyyppi

- Erilaisia palvelimia.
- Toisissa sovelluksen / palvelun osia, toisissa tietokantoja. Sisäisesti hajautettu.
- Client-sovelluksia, selainpäätteitä.
- Komponenttien välisiä tietoliikenneyhteyksiä (verkko).
- Useita kanavia.
  - Erilaisia käyttäjäryhmiä ja niissä suuri määrä käyttäjiä – ja käyttötapoja ja tarpeita.



Kolmitasoarkkitehtuuri oleellinen

- Esityskerros
  - Clientin käyttöliittymä, WWW-palvelin
- Liiketoimintakerros
  - Sovelluspalvelin
- Datakerros
  - Tietokantapalvelin
  - Tietokanta.

Erilaisia testauksen konkreettisia kohteita

- Arkkitehtuurin kerroksia ohitetaan testausjärjestelyissä suunnitelmallisesti.
- Kohteena voi olla ohjelmisto, laite tai tietokanta – yhdessä tai eri testeissä.
- Omat testit mahdollistavat testien riippumattomuuden, helpot testijärjestelyt sekä helpon ja aikaisen ajoituksen.
- Kokonaisuuden testaus tärkeää lopputesteissä.

Missä pullonkaulat usein ovat

- WWW-sovelluksissa:
  - Useimmin sovelluspalvelimessa.
  - Tietokantapalvelimessa lähes yhtä usein. (In-memory -tietokannat ja klusterointi parantaneet aiemmin pääosin levypohjaisten tietokantojen suorituskykyä valtavasti.)
  - Verkossa vähemmän.
  - Vähiten WWW-palvelimessa.
- Palvelun käynnistyksen yhteydessä.
- Kaikissa epäjatkuvuuskohdissa.
- Suurilla syötteillä.
- Kaikki asetetut rajat – rajattomuus on huono ratkaisu, mutta minne osataan laittaa rajat, esim. sallittujen yhteyksien määrä.

Palvelun testattavuus

- Varsinkin kehittämisen aikana tehtävässä testauksessa on oleellista palvelun testattavuus:
  - Perusarkkitehtuurin toteutus.
  - Rajapintojen toteutus standarditekniikoilla (esim. http-kutsut).
  - Modulaarinen ja kerrosarkkitehtuuri.
  - Käyttöliittymien ohittamismahdollisuus testiohjelmilla.
  - Tarpeeton skriptikielten käyttö, mm. JavaScript, tuottaa usein ongelmia.
  - Testiympäristön olemassaolo ja mahdollisuus käyttää.
- Palvelun keskeneräisyys vaikuttaa suorituskykyyn vahvasti.
  - Debug-versiot. Esimerkiksi debug-tulostukset voivat hidastaa sovellusta merkittävästi.
  - Virittämätön tietokanta.
- On ymmärrettävä tarkoin, mitä ollaan testaamassa; mikä on testattavan järjestelmän tila ja ominaisuudet.

Bugit pois ennen testausta

- Ennen kuormitustestausta on varmistuttava ohjelmiston toimivuudesta.
  - Testaus on usein operaatio, jota valmistellaan vahvasti ja joka sitoo monien ihmisten panoksen – siksi sen pitää onnistua ilman teknisiä ongelmia.
  - Ohjelmiston testattavien ominaisuuksien on läpäistävä toiminnallisuustestit.
- Kuormitustestit on testattava ennen testipäivää.
- Kuitenkin on valittava sellaisia testejä ja testiaineistoja, jotka eivät ole herkkiä ”rikkomaan” ohjelmistoa.
  - Siis mielellään suhteellisen yksinkertaisia.

Vaatimusmäärittely tuottaa testattavat vaatimukset 1/5

- Olennaisimpia ovat palvelun liiketoiminnalliset tavoitteet: millaista kapasiteettia tarvitaan.
  - Ennen suorituskykytestausta on tarkoin määritettävä nämä tavoitteet ja projektin reunaehdoista syntyvät huolenaiheet – esim. pärjätäänkö suunnitellulla laitteistolla?
- Kasvunvara otettava huomioon.
  - Olisi järjetöntä muuttaa palvelua tiheään sen käyttäjämäärien vähitellen kasvaessa.
  - Hyvä palvelu toimii jonkin matkaa eteenpäin peruslaitteistolla.
  - Skaalautuvuus oleellista. Jos nyt edellytetään 100 samanaikaista käyttäjää, miten palvelu saadaan tukemaan 1000:tta samanaikaista käyttäjää. Miten palvelu voidaan esimerkiksi hajauttaa usealle palvelimelle tai päivittää palvelimen kapasiteettia (vaihtaa palvelinkone tai vaikkapa ostaa siihen jo asennettuja prosessoreita käyttöön.)

Vaatimusmäärittely tuottaa testattavat vaatimukset 2/5

- Palvelun käyttöprofiili.
  - Kun asiakasryhmät ja niiden tarpeet ja tavat on tunnistettu, kyetään määrittelemään palvelun profiili mm. sen suhteen, miten kapasiteettitarve ajoittuu ajallisesti.
  - Ruuhkahuiput päivän aikana, viikon aikana jne... Tunnistettava ajankohdat, jolloin on paljon kuormitusta. Esimerkiksi puhelinjärjestelmissä ja työajankirjausjärjestelmissä perjantai klo 16.
  - Näitä kannattaa selvitellä vastaavien palvelujen historiasta, jos mahdollista.
  - Odotettavissa olevat tilanteet – esimerkiksi kiinnostavan uutisen julkistuksen jälkeen voi sivuston kävijämäärä olla 10 x normaalia suurempi.

Vaatimusmäärittely tuottaa testattavat vaatimukset 3/5

- Yksittäisen käyttötapauksen kuormitusprofiili. Esimerkiksi tiedostojen latauksen kokohajonta – on tärkeää selvittää toimivuus suurilla tiedostoilla / dokumenteilla (worst case -tilanteet). Samoin erityisen raskaat tietokantahaut ovat hyviä testitapauksia.
- Edellytettävä luotettavuus / käyttövarmuustaso.

Vaatimusmäärittely tuottaa testattavat vaatimukset 4/5

- Edellytettävät vasteajat.
  - Käyttötapauksittain, skenaarioittain, käyttötehtävittäin (riippuen ohjelmistoprojektin tavasta määrittää käyttäjien tavoitteellisia asioita).
  - Joko määritetty tai perustuvat tuoteryhmän standardivaatimuksiin – eli testaajien pitää tarvittaessa määrittää. Kaikkea ei tarvitse olla vaatimusmäärittelyssä.
  - Halutulla käyttäjämäärällä.
  - Aina ei vasteajoilla ole suurempaa merkitystä – kunhan palvelu ylipäätään toimii.
- Haluttu kapasiteetti.
  - Käyttäjämäärät, joita palvellaan.
  - Medialatausten määrä.

Vaatimusmäärittely tuottaa testattavat vaatimukset 5/5

- Järjestelmävaatimukset.
  - Pärjättävä tietyllä kokoonpanolla.

- => Kvantifiointi testausta varten.
  - Vasteaika tietyllä käyttäjämäärällä, tietyssä käyttötapauksessa.
  - Montaako käyttäjää kyetään palvelemaan tietyllä palvelutasolla (vasteajalla).
  - Montako latausta onnistuu tietyllä palvelutasolla.

Miten päästä käsiksi mitattaviin järjestelmän osiin? 1/2

- Aina ei ole järkevää tehdä end-to-end -testausta.
- Analysoidaan testattava järjestelmä ja tunnistetaan sen pullonkaulat.
- Käyttöliittymät voi ohittaa tai niitä voi simuloida ohjelmallisesti.
- Jos järjestelmät on tehty testattaviksi, niissä on hyviä rajapintoja, joilla käytetään palveluja.
- Yhä useammin löytyy http-rajapinta, jota voidaan kuormittaa muutama kerros ohittaen.
  - Esimerkiksi mobiilijärjestelmien back-end:ssä suoraan käsiteltävä rajapinta.

Miten päästä käsiksi mitattaviin järjestelmän osiin? 2/2

- Rajapintoja ei yleensä ole dokumentoitu, vaan ne täytyy selvittää
  - Käyttöliittymästä voidaan tunnistaa http-kutsuja.
  - Käyttöliittymän toimintaa voidaan nauhoittaa testausohjelman skripteiksi – ja hyödyntää tietoja myös muissa ohjelmissa.

Testaussuunnittelu 1/3

- Vaatimusmäärittelyn päälle tarvitaan aina syvällistä analysointia käyttäjien toiminnasta ja järjestelmästä.
- Jos ohjelmiston vaatimusmäärittely ei kerro vasteaikoja tai määritä palvelun profiilia, ne on määritettävä testaussuunnittelun aikana.
- Suorituskykymalli kuvaa ymmärryksen järjestelmän kapasiteettiin vaikuttavista tekijöistä.
- Testiympäristön ja tuotantoympäristön välille tehdään usein vertailuyhtälö: montako käyttäjää vastaa montaako toisessa järjestelmässä. Tämä perustuu esim. erilaisiin prosessoreihin yms.
- Automaattiset testitapaukset tehdään tärkeimmistä toiminnoista.

Testaussuunnittelu 2/3

- Uusien testien verifioinnissa vanhat testit toimivat referenssinä – jos on vanhoja testejä.
- Testitapauksia ajetaan arvioitujen käyttöprofiilien suhteessa – siis, miten "käyttäjämassat" tulevat toimimaan tuotantoympäristössä.
- a) Testit ajetaan usein eri kuormitustasoilla ja piirretään vasteista graafeja.
  - Järjestelmän stabiiliuden vuoksi käyttäjämääriä lisätään usein vaiheittain – pienistä määristä suuriin.
- b) Tehdään testaus vaadituilla maksimikuormilla ja todetaan suorituskyky. Kuorman nosto vähitellen.
- c) Etsitään huippukuormatilanteet, joissa järjestelmä ei enää täytä suorituskykyvaatimuksia.

Testaussuunnittelu 3/3

- Kuormaa pitää joskus jakaa tulemaan eri IP-osoitteista. Tällöin tarvitaan ohjelmisto, joka osaa käyttää useaa päätettä (esim. LoadRunnerin agenttityöasemat ja JMeterin Remote Testinging, ks. http://jakarta.apache.org/jmeter/usermanual/remote-test.html).
- Samalla, kun verifioidaan suorituskyky, on järkevää tunnistaa pullonkaula – onko se tietoverkossa, palvelimen prosessorissa, muistin määrässä vai missä?
- Suorituskykytestaukseen voi kannattaa liittää jonkinasteista pitkäaikaistestausta, jolla saadaan kiinni esimerkiksi useiden päivien aikana tapahtuvat muistivuodot – joille järjestelmä on aina altis nimenomaan kuormitettaessa.
- Suunnittelutiimi vaihtelee. Ideaalitilanteessa suunnitteluun osallistuvat järjestelmän kaikkien elementtien vastuuhenkilöt.

Suunnitteludokumentit

- Huomioon ottaminen yleissuunnitelmissa: ohjelmistoprojektin projektisuunnitelma, laadunvarmistussuunnitelma, testauksen pääsuunnitelma.
- Yleensä aina tehtävä oma testaussuunnitelma.
- Katselmointi tärkeää, koska jo testauksen konkreettisen kohteen ja testausjärjestelyjen valinta on kriittistä – eroaa tässä muista testityypeistä.
- Kaikkien järjestelmän pääelementtien vastuuhenkilöiden on hyvä osallistua katselmointiin.

Resursointi

- Testityypin monimutkaisuudesta johtuen on vaikea antaa nyrkkisääntöjä.
- Testauksen suunnitteluun ja kohteen ymmärtämiseen menee aikaa.
  - On helppo luoda testijärjestelyjä, jotka tuottavat täysin virheellisiä tietoja.
  - Luotettavien tietojen tuottaminen on työläämpää.
- Työmäärä riippuu myös järjestelmän testattavuudesta.
- Isohkoilla järjestelmillä riittävästi kalenteriaikaa – esim. 1 kk.
- Kokonaisresurssit itseltä ja asiakkaalta vaihtelevat hyvin paljon.
  - Testaus 2006 -tapahtumassa kerrottiin esimerkki Postilta, jossa suorituskykytestaukseen osallistui kolmena viikonloppuna 40 henkeä. vaikka käytössä oli LoadRunnerilla automatisoitu testaus.

Testispeksi 1/2

- Verifioivat testitapaukset:
  - Käyttötapaukset, joille määritetään maksimi vasteaika.
  - Testitapaukset, jossa käyttötapauksia suoritetaan samanaikaisesti N kappaletta.
  - Hyväksymisehtona vasteajan toteutuminen.
  - Ennakkoehtona tietty järjestelmän tila, joka on stabiloitu muutamalla testiajolla.

Testispeksi 2/2

- Mittaavat testitapaukset:
  - Käyttötapauksia suoritetaan samanaikaisesti N kappaletta, jossa N = yhdestä haluttuun maksimikapasiteettiin. Mitataan vasteajat ja niiden keskiarvo, minimi ja maksimi.
  - Kasvatetaan määrää, kunnes palvelin selvästi hyytyy.
- Testauksen dynaaminen ohjaus:
  - Testejä toistetaan muutama kerta.
  - Tarkkaillaan järjestelmänä käyttäytymistä ja analysoidaan käyttäytymisen syitä ja vaikutuksia testituloksiin.
  - Uudistetaan tarpeen mukaan testiympäristön konfiguraatiota.

Vasteaikojen ymmärtäminen

1. Käyttäjä käynnistää pyynnön	
2. Käyttäjä päättää pyynnön	Syöteaika
3. Pyyntö siirretään järjestelmälle	Tietoliikenneviive
4. Järjestelmä aloittaa suorituksen	2-4: Järjestelmän reaktioaika
5. Järjestelmä aloittaa vastauksen	Järjestelmän suoritusaika
6. Vastaus siirretään käyttäjälle	Tietoliikenneviive 2-5: Vasteaika (jos vastauksesta hyötyä alusta asti ja jos sitä)
7. Järjestelmä saa vastauksen valmiiksi	2-7: Vasteaika
8. Käyttäjä tulkitsee vastauksen	-8: Toiminnon kesto

Testausympäristö 1/3

- Tilanteissa, joissa suorituskykytestausta tehdään, on usein käytössä asiakkaan testausympäristö.
- Verkkopalveluita joudutaan usein testaamaan asiakkaan tiloissa sisäverkossa.
- Selvitettävä tarkoin, etteivät testien vaikutukset vuoda tuotantotietokantoihin tai muihin järjestelmiin.
- Palvelinten monitoroinnissa tarvitaan paikallisen IT-osaston tukea.
- Koska testiympäristöä kuormitetaan, on sovittava testausajankohdista, sallituista toimista ja siitä, että paikalla on henkilöstöä tarpeen mukaan nostamassa palvelimet pystyyn.

Testausympäristö 2/3

- Ympäristön vastaavuus tuotantoympäristön kanssa tärkeää.
  - Tunnistettava erot.
  - Analysoitava niiden vaikutukset.
- Tunnistettava tuotantoympäristössä oleva muu toiminta ja otettava se huomioon – muu tietoliikenne, muut prosessit, niiden käyttäytyminen.
- Tiedettävä muu testiympäristössä tapahtuva toiminta.
- Huomion kiinnittäminen tietoverkkoon. Cachet, proxyt yms. voivat vääristää vakavasti tuloksia.
- Käyttäjiä simuloitaessa otettava huomioon, mistä ne kytkeytyvät; miten todelliset tietovirrat kulkevat.

Testausympäristö 3/3

- Tietokannan koko oltava tuotantoa vastaava.
- Tietokannan "puhtaus" voi vaikuttaa suorituskykyyn. Tietokanta, joka on juuri täytetty hyvin käyttäytyvällä testiaineistolla toimii eri tavalla kuin ajan myötä kehittynyt tietokanta.
- Käyttäjien, oikeuksien luominen ym. yms. tehdään usein konfigurointitietokannan (taulun) avulla.
- On ymmärrettävä sovelluksen toiminta. Voidaanko testeissä esimerkiksi käyttää samaa käyttäjätunnusta vai pitääkö niitä olla käytössä vaikkapa tuhat kappaletta?
- Järjestettävä hyvät logit testauksenaikaisten virheiden dokumentointiin.

Testauksen tilaajan yleisiä tehtäviä

- Testausympäristö.
  - Tarjoaminen käyttöön.
  - Sopivien olosuhteiden luominen – ei muita testauksia jne.
  - Operaattori koko testauksen ajan auttamaan ongelmatilanteissa.
  - Palvelimen monitoroinnin järjestäminen.
- Testaajille pääsy tiloihin, kulku, kahvit jne.
- Testaajan tietoliikenneyhteydet.
- Järjestelmän elementtien ja rajapintojen kuvaus.
- Käyttäjätunnukset testausta varten.
- Mahdollisesti testiaineisto.

Testiaineisto / tietokannat

- Toisin kuin toiminnallisuustestauksessa, testiaineiston pitää olla varmatoimista
  - Testaus ei saa keskeytyä bugeihin, joita vaativa aineisto paljastaa.
- Otettava huomioon tyypilliset käytettävät aineistot ja worst case -tilanteet (vaikkapa suuret kuvatiedostot).
- Ymmärrettävä tietokannan käyttäytyminen.
  - Jos syötetään tietoja ja haetaan samaa tietoa, se löytyy välimuistista.
  - Indeksointi tietokantaa täytettäessä.

Suorituskykytestauksen testitapauslista (esimerkki)

ID	Kuvaus	Testauksen kohde	Kuorma	Priori-teetti
10	Kuormitustesti yhdellä käyttäjällä (testisetin savutesti)	Suorituskyky-	1 käyttäjä	1
15	Pieni käyttäjämäärä	Suorituskyky-	5 käyttäjää	1
20	Suunnitellun käyttäjämäärän verifiointi	Suorituskyky-	120 käyttäjää	
25	Pullonkaulatesti	Kuormitus	Haetaan rajat	2
35	Datamäärätesti	Määrätesti	Suurimman datamäärän verifiointi	

Mittarit – välituloksia / standardi ISO 9126 1/4

Mittari	Tarkoitus	Kommentteja
Vasteajat	Mikä on tietyn toiminnon toteuttamiseen kuluva aika? Miten pitkä aika kuluu ennen kuin järjestelmä vastaa tiettyyn operaatioon?	Viittaa yksittäisen käyttö/työtehtävän vaatimaan aikaan.
Vasteajat (keskimääräinen vasteaika)	Mikä on käyttäjän keskimääräinen odotusaika komennon syöttämisen jälkeen, määritellyssä järjestelmän kuormitus- ja käyttötilanteessa?	
Vasteajat (worst case -vasteajan suhde)	Mikä on absoluuttinen aikaraja, joka edellytetään toiminnon toteuttamiseen? Voiko käyttäjä huonoimmassa tapauksessa saada vastineen määritetyssä ajassa? Voiko käyttäjä huonoimmassa tapauksessa saada vastineen vielä siedettävässä ajassa?	

Mittarit – välituloksia / standardi ISO 9126 2/4

Mittari	Tarkoitus	Kommentteja
Kapasiteetti (throughput)	Montako tehtävää kyetään suorittamaan tietyssä ajassa?	Tämä on työn tuottavuuteen liittyvä mittari käyttäjänkin kannalta, mutta myös teknisen systeemin suorituskykymittari. (Vrt. automaattiset tietokantatoiminnot, transaktioiden määrä.)
Kapasiteetti (keskimäärin)	Montaako samanaikaista tehtävää järjestelmä kykenee suorittamaan tietyssä ajassa?	
Kapasiteetti (worst case)	Mikä on järjestelmän absoluuttinen raja samanaikaisissa tehtävissä?	Esimerkiksi samanaikaisten käyttäjien määrä.
Toimintoihin kuluva aika	Mikä on käyttäjän odotusaika komennon syöttämisen jälkeen ennen kuin hän pystyy käynnistämään uusia toimintoja (jotka liittyvät samaan asiaan)?	Kauanko siis järjestelmä prosessoi tehtävää.

Mittarit – välituloksia / standardi ISO 9126 3/4

Mittari	Tarkoitus	Kommentteja
Toimintoihin kuluva aika (keskimää-rin)	Mikä on käyttäjän keskimääräinen odotusaika komennon syöttämisen jälkeen ennen kuin hän pystyy käynnistämään uusia toimintoja, tietyissä järjestelmän kuormitustilanteissa?	Tämä on sama mittari, mutta yksittäisten toimintojen sijaan tutkii keskiarvoja ja ottaa käytännön olosuhteet huomioon.
Toimintoihin kuluva aika (worst case)	Mikä on toimintoon kuluvan ajan ehdoton yläraja? Mikä aika kuluu toimintoon pahimmassa tapauksessa?	
Odotusaika	Mikä osuus käyttäjän ajasta kuluu järjestelmän vastaamisen odottamiseen?	Tällaiset odotusajat ovat turhaa aikaa ja kuluttavat organisaation resursseja. Tämä on selvä tuottavuusmittari.

Mittarit – välituloksia / standardi ISO 9126 4/4

Mittari	Tarkoitus	Kommentteja
Suurin muistin käyttö	Mikä on toiminnossa tarvittava suurin muistin määrä?	Määrittää muistivaatimukset.
Tiedonsiirto-kapasiteetin tarve	Mikä on ehdoton yläraja tiedonsiirtoon, mitä toiminto tarvitsee?	
Tiedonsiirto-kapasiteetin käyttö	Pystyykö järjestelmä toimimaan odotetun tiedonsiirtokapasiteetin puitteissa?	Olennainen testattava asia! Odotettu kapasiteetti riippuu käyttäjistä, vrt. modeemikäyttäjät.

Suorituskykytestauksen tarkistuslista ennen testisessiota

- Asiat pitää olla kunnossa, kun asiakkaan luona aloitetaan testaus.
- Nämä tarkistettava:
  - Kaikki testijärjestelmän elementit.
  - Tunnukset ja oikeudet.
  - Testidata.
  - Testijärjestelmän eristys. Se ei saa vaarantaa muita palveluja. Ulkopuolisten WWW-sivujenkin kuormitus voi olla laitonta tietoliikenteen häirintää.
  - Monitorointi.
  - Loggaus.
  - Testausohjelmat.
  - Skriptit.
  - Henkilöt.
  - Roolit ja vastuut.

Testipäivän kulku

- Tiimin kokoontuminen ja päivän tapahtumien läpikäynti.
- Testausvalmiuden tarkastus.
- Testiympäristön valmistelu.
- Testiympäristön toimivuuden testaus.
- Testit.
- Logien keruu.
- Testijärjestelmän siivous.
- Päivän päätöspalaveri.

Jatkuva monitorointi testauksen aikana 1/2

- Jatkuvat aikasarjat sen mukaan, mitä aikasarjoja saadaan aikaiseksi.
- Useiden aikasarjojen samanaikainen esitys:
  - Käyttäjämäärät.
  - Tietoliikenteen määrä.
  - Tapahtumia sekunnissa.
  - Prosessorin (prosessorien) käyttö.
  - Muistin käyttö.
  - Levyjärjestelmän käyttö.

Jatkuva monitorointi testauksen aikana 2/2

- Virheilmoitusten seuranta – käyttäjille ei saa tulla esimerkiksi koodia tai suoria palvelinohjelmiston virheilmoituksia. Ideaalitilanteessa kaikkien järjestelmän elementtien vastuuhenkilöt valvovat testiä.
- Poikkeamien raportointi testauksen vetäjälle / koordinaattorille.
- Usein testin aikana tehdään manuaalista testausta, jolla pyritään tunnistamaan sellaisia ongelmia, jotka eivät näy monitoreissa ja logeissa.

Tulosten tulkinta

- Suorituskykytietoja on aina tulkittava, koska testaustilanne ei ole todellinen.
- Ymmärrettävä, mitä data kertoo?
- Johtopäätökset järjestelmän suorituskyvystä ovat paljon helpompia, mutta johtopäätökset pullonkauloista eivät.
- Kaiken testauksen perusasiat:
  - Testijärjestelyjen tuottamien ilmiöiden ja virheiden tunnistaminen ja erottaminen.

Testausraportti

- Normaali testiraportti.
- Tämä testityyppi perustuu muita enemmän asiakkaan konkreettisiin tiedontarpeisiin, joihin vastaukset ovat lyhyitä: kyllä, ei, 57...
- Keskeistä on yhteenveto, jossa vastataan voimakkaasti asiakkaan selkeisiin tiedontarpeisiin.
- Tuloksena voi olla testitulosten ohella skaalautuvuusstrategian määrittely – millä tavalla päästään seuraavalle suorituskykytasolle.
- Graafit auttavat visualisoimaan suorituskykyä. Esimerkiksi vasteajat suhteessa käyttäjämääriin.
- Tulosten analysointi tärkeää, ettei tule käytännön yllätyksiä siirrettäessä järjestelmä tuotantoon.
- Testausraportin käsittely vaihtelee. Joissakin organisaatioissa se on osa tuotantoon siirron prosessia ja raporttidokumenttiin otetaan vastuuhenkilöiden hyväksyntä.

Haasteita

- Ympäristöjärjestelyt.
- Asiakkaan kanssa sovittava, että löytyy henkilöitä auttamaan testauksessa.
- Testien luominen, jotka todella simuloivat todellisuutta.
- Kohteen ymmärtäminen ja heikoimman lenkin tunnistaminen.
- Testeissä todettujen pullonkaulojen ymmärtäminen ja ongelmien jäljittäminen.

Tyypillisiä testauksen virheitä

- Lähtökohdaksi ei ole tunnistettu käyttäjien tarpeita.
- Järjestelmää kehitettäessä ei ole otettu huomioon sen testattavuutta.
- Generoitu kuorma liian yksipuolista.
- Pullonkauloja ei ole tunnistettu – verkko, tietokanta, sovellus, liittymät.
- Testauksen aikaisia virheitä ja häiriöitä ei ole huomattu (monitorointi, logit, testiskriptien laatu).
- Ympäristötekijöitä ei ole otettu huomioon.

Uudelleenkäytettävyys

- Kuormitustestien on hyvä olla uudelleenkäyttöisiä, että niitä voidaan ajaa uudestaan muutosten jälkeen.
- Olennaista:
  - Testien yleinen standardointi, ohjeistus, lomakkeet.
  - Dokumentointi ja kommentointi.
  - Nimeämiskäytännöt.
  - Ylläpitojärjestelyt.
  - Referenssitestit, joiden avulla saadaan kuva uusien testien toimivuudesta.

Testauksen aloittaminen organisaatiossa

- Prosessit.
  - Milloin testataan erilaisten prosessien puitteissa.
  - Roolit ja vastuut.
  - Hyväksymismenettelyt.
- Osaajat.
  - Erikoisosaajien koulutus.
  - Roolien ja vastuiden määrittäminen.
- Infrastruktuuri.
  - Testausohjelmistot.
  - Testiympäristöt.