JATKUVA INTEGRAATIO

1) Johdanto (3 sivua)
Fowleria:. \cite{Fow06}
Vaihtoehtoisesti myös Shorea: \cite{Sho10}
Historia: Grady Booch 1991, XP 90-luvun lopulla, Fowler 2000. \cite{HTH+16} \cite{SB13}
Tutkimus: Ei tutkittu tarpeeksi ja lisää vois. \cite{HTH+16} \cite{SB13}
Projektien eroja, syy-seuraussuhteita \cite{SB13}, yritysten eroja \cite{MSB17}.

2) Jatkuva integraatio (2+2+1=5 sivua)
Jatk. int. on ketterä. \cite{PRC17}
Jatk. int. on mentestystarina, jonka suosio on kasvussa. \cite{HTH+16}
Jatk. int. ei floppaa. \cite{HTH+16}
Fig 2, Fig 3, Tab 6. \cite{HTH+16}
Avoimen lähdekoodin projekteissa isoin este jatk. int. käyttöönotolle on osaamisen puute. Yliopistoille hommia. \cite{HTH+16}
Suositut a. lähdek. projektit käyttävät jatk. int:a. \cite{HTH+16}
Hyödyt suuremmat kuin haitat. \cite{HTH+16}
Toisaalta ei niin vakuuttuneita, hyötylistan ei odoteta kasvavan. \cite{SB13}
Toteutus-tilastoja. \cite{MSB17}
Buildin rikkoutuessa devaajat korjaavat sen heti. \cite{LCL+16}

2.1) <Hyvät ja huonot puolet> (2)
Hyvä: kommunikaatio sisäinen ja ulkoinen. \cite{SB13}
Hyvä: projektin ennustettavuus. \cite{SB13}
Hyvä: rinnakkainen devaus. \cite{SB13}
Hyvä: Bugit löytyy ajoissa. \cite{PRC17} \cite{HTH+16}
Hyvä: Automaatio. \cite{PRC17}
Hyvä: Laatu paranee. \cite{PRC17}
Hyvä: Nopeuttaa. \cite{PRC17}
Hyvä: Eri alustoilla testaaminen? \cite{PRC17}
Hyvä: Pull requt hyväksytään nopeammin. \cite{HTH+16}
Huono: devaus- ja testausvälineet eivät yhteensopivia. \cite{LCL+16}
Huono: yksikkötestejä kirjoitetaan käsin, käsitys testitoiminnoista blurrautuu, integraatiotestaus ei riittävää. \cite{LCL+16}
Huono: testejä pitää päivittää. \cite{LCL+16}
Huono: Työnjako, arkkitehtuuri, vaikeet testijutut <mitä muuta siinä oli> \cite{MSB17}
Taulukko esteistä. \cite{MSB17}
Huono: Configuraatiot ci-palvelimelle ja riippuvuuksien kanssa pelaaminen vaikeita etenkin uusille. \cite{PRC17}
Huono: Testeihin luotetaan ihan liikaa, vaikka ne voi olla huonoja tai epädeterministisiä. \cite{PRC17}
Huono: Devaajilla pitää olla itsekuria. \cite{PRC17}
Huono: Testikulttuuri. \cite{PRC17}
Huono: Testien automatisointi sekä palvelut ja välineet maksaa. \cite{PRC17}
Devaajat välttävät rikkomasta buildia. \cite{HTH+16}

2.2) <Testitoiminnot> (1-2)
Devaajat välttävät rikkomasta buildia2. \cite{HTH+16} \cite{SB13}
Helpompi merge ja troubleshooting. \cite{SB13}
Testaamista vaikeuttaa, jos ei ole kontaktia asiakkaisiin. \cite{SB13}
Ketterällä testaamisella ja jatk. int. korrelaatio, muttei luultavasti kausalisaatiota. \cite{SB13}
Toisaalta nää väittää, että testaaminen on edellytys jatk. int:lle. \cite{LCL+16}
Buildin ja laadun kommunikoiminen on ristiriitaista. \cite{SB13}
Buildien rikkoontumiseen yleensä teknisiä ja sosiaalisiä syitä, kuten riittämätön testaaminen ja kiire. \cite{PRC17}
Ci-build-testaus-devaus-järjestelmien pitää jutella keskenään sujuvasti. \cite{LCL+16}
Iso testikattavuus ei välttämättä ole epäketterää. \cite{LCL+16}
Turvallisuuskriittisissä ohjelmistoissa ankara testaaminen on tärkeää, muutokset ovat erilaisia kuin esim. webbisoftassa. \cite{LCL+16}

3) Regressiotestaaminen jatkuvassa integraatiossa (2+5 sivua)

Olemassa olevaa tekstiä. Muokkaa rakennetta ja lisää sitaatit. \cite{ERP14} \cite{LIH17} \cite{MGN+17}
Kombinatorinen testaus<ei jatkoon> \cite{LCL+16}

2.1 <Pullonkaula> (2)
Olemassa olevaa tekstiä. Muokkaa rakennetta ja lisää sitaatit.
Lisää siitä, miksi regressiotestaaminen on niin iso pullonkaula, että siitä kannattaa kirjoittaa kokonainen luku. \cite{ERP14} \cite{LIH17} \cite{MGN+17}

2.1 <Mikä rikkoo buildin> (2)
Buildien rikkoontumiseen yleensä teknisiä ja sosiaalisiä syitä, kuten riittämätön testaaminen ja kiire. \cite{PRC17}
Mitkä tekijät liittyvät build-statuskaavion tilasiirtymiin. Miten testit feilaavat, miten sen voi ottaa huomioon testikokonaisuuden muodostamisessa. \cite{LIH17}
Ohjelmistokehittäjille voisi räätälöidä viestejä sen mukaan, mitkä tekijät hänen työskentelyssään vaikuttavat buildin rikkoutumiseen. Esim. ohjelmointikieli, kuinka monta kokkia, henkilökohtainen rikkomisprosentti, tekeekö uutta koodia jne. \cite{MGN+17}

Kaavioita \cite{LIH17} \cite{MGN+17}

2.2 Testaaminen Googlella (1-2)
Google on poikkeuksellinen paikka. Siellä on helkkaristi jengiä ympäri maailmaa ja jatkuvasti kasvava jatkuvan integraation määrä on mahdoton haaste jopa Googlen resursseille. Googlen pre- ja post-submit-testaaminen tapahtuu testipalvelimella, jonne ohjelmistokehittäjä lähettää "changelistin", jossa on kyseiseen ajoon liittyvät "testikohteet". Testipalvelin priorisoi tyylillään. Lisäksi yleensä kolmen vartin välein ajetaan jono, eli vältetään samojen testikohteiden ajamista lyhyen ajan sisään. Tällä on hyviä ja huonoja puolia. \cite{ERP14} \cite{MGN+17}
Kaavioita \cite{ERP14} \cite{MGN+17}
Ehkä vähän ehdotuksesta tehokkaammaksi priorisointi- ja valintamenetelmäksi. Tai sitten siitä voisi ottaa vain yleisesti sitä, mitkä lainalaisuudet pätee ison testisysteemin tehostamisessa. \cite{ERP14}

4) Yhteenveto (2 sivua)

= 18 sivua