Aineopintojen harjoitustyö: Ohjelmistotekniikka (4 op)

Aloitusluento 1.11.-23

1

Kurssista

Kai Korpimies, Valtteri Kantanen, muut assarit

Pieni harjoitustyö Python-kielellä
Alussa muutama harjoitus, ei tenttiä

Esitietoina Ohjelmoinnin jatkokurssi, Tietokantojen perusteet, Tietokannat ja web-ohjelmointi tai vastaavat tiedot sekä Tietokone työvälineenä

Kurssimateriaali: https://ohjelmistotekniikka-hy.github.io/
- sama linkki löytyy kurssisivulta

2

Osaamistavoitteita

"Kurssin suoritettuasi
- tunnet ohjelmistotuotantoprosessin vaiheet,
- olet tietoinen vesiputousmallin ja ketterän ohjelmistotuotannon luonteesta,
- osaat soveltaa versionhallintaa osana ohjelmistokehitystä,
- osaat soveltaa UML-mallinnustekniikkaa vaatimusmäärittelyssä ja ohjelmiston suunnittelussa tarkoituksenmukaisella tavalla
- tunnet ohjelmiston testauksen eri vaiheet
- osaat soveltaa automatisoitua testausta yksinkertaisissa ohjelmistoprojekteissa
- tunnet tärkeimpiä ohjelmiston suunnitteluperiaatteita ja osaat soveltaa niitä yksinkertaisissa projekteissa"

3

Kurssipalautteesta

"Kevään palautteessa osaamistavoitteet, toteutustapa, materiaalit ja arviointimenetelmät saivat kukin arvosanakseen vähintään 4.2/5. Työmäärää pidettiin sopivana tai jonkin verran raskaanpuoleisena. Sanallisissa palautteissa mm. kaivattiin lisää ohjausta ja mainittiin viikoittaisten kriteerien selkeys.

Ohjausta pyritäänkin nyt mahdollisuuksien mukaan tehostamaan ja kiinnitämme myös huomiota työmäärään ja kriteerien selkeyteen."

4

Ohjelmistotuotanto

Tarvitaanko seuraavissa tilanteissa erilaisia käytäntöjä?

1. kirjoitat pienen ohjelman ohjelmointikurssin harjoituksissa
2. toteutat itse vähän isomman projektin
3. osallistut avoimen lähdekoodin projektiin, jossa on monta ohjelmoijaa tai
4. osallistut ohjelmistotuotantoon jossain ohjelmistotalossa

Monellako kokemusta tilanteista 2 - 4?

5

Ohjelmistotuotanto

Pohdittavaa mm.:
- järjestelmällisyys
- versioiden ja riippuvuuksien hallinta, koodin laadunvarmistus, testaus, automatisointi....
- yhteistyö, työnjako, toimivat käytännöt
- ohjelmiston ei-toiminnalliset vaatimukset (tehokkuus, turvallisuus, käytettävyys ym.)
- asiakkaan/käyttäjän näkökulma

6

Ohjelmistotuotannon vaiheet

Vaatimusmäärittely

Suunnittelu

Toteutus

Testaus

Ylläpito

7

Vaatimusmäärittely (requirements analysis)

Mitä ohjelmiston tai järjestelmän tulee tehdä
- esim. tarjottavat palvelut, suorituskyky, kustannukset
- ei niinkään vastauksia kysymykseen "miten"

Käyttäjän/asiakkaan näkökulma

Huomioidaan toimintaympäristö ja toteutusteknologia

8

Suunnittelu (system design)

Miten onnistuu? Järjestelmän rakenne, toiminta, tietosisältö?

Arkkitehtuurisuunnittelu keskittyy ohjelmiston yleisrakenteeseen:
- minkälaisista "suuremmista osista" ohjelmisto koostuu ja minkälaisia keskinäisiä suhteita näiden osien välillä on
- rajapinnat, riippuvuudet

Oliosuunnittelu tarkentaa:
- mitä luokkia, miten toimivat yhdessä?

Suunnittelun tulokset dokumentoidaan

9

Toteutus

Ohjelmointi etusijalla

Ei yleensä ole pelkkää mekaanista koodausta määrittelyn ja suunnittelun mukaisesti

Yleensä hyödynnetään monenlaisia välineitä, valmiita kirjastoja, sovelluskehyksiä...

10

Testaus

Tavoitteena on löytää virheet jotka estävät ohjelmistoa toimimasta määritysten mukaisesti.

Yksikkötestaus (unit testing): täyttävätkö yksittäiset luokat vaatimuksensa?

Integraatiotestaus: toimivatko luokat yhdessä oikein?

Järjestelmätestaus: toimiiko järjestelmä kokonaisuudessaan (ml. käyttöliittymä) vaatimusten mukaisesti?

11

Vesiputousmalli (waterfall model)

Edetään "top down" -tyyliin em. vaiheet yksi kerrallaan
- eri vaiheet ehkä eri tahojen vastuulla

Periaatteessa suoraviivaista, mutta:
- yleensä aiempiin vaiheisiin on pakko vielä palata -> lisätyötä!
  - käyttäjien toiveet eivät usein selviä heti alussa
  - muutenkaan ei ehkä vielä tiedetä tarpeeksi
  - toimintaympäristö voi muuttua
- testaus alkaa kovin myöhään

Miten näitä ongelmia voisi ratkaista?

12

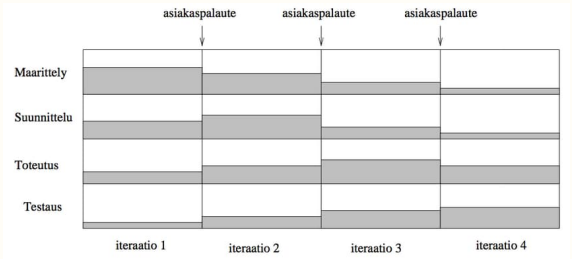
Ketterät (agile) menetelmät

Alussa vaatimukset pääpiirteissään ja arkkitehtuurin alustavaa hahmottelua, sitten ohjelmallista iterointia ja inkrementaalista etenemistä

- jokaisessa iteraatiossa suunnitellaan, toteutetaan ja testataan pieni osa ohjelmistoa
- asiakas voi kokeilla ohjelmistoa jokaisen iteraation jälkeen

Vaatimuksia ja arkkitehtuuria voidaan tarkentaa ja muuttaakin

13



Edetään kurssilla ketterästi...

14

Esimerkki: TodoApp

Kurssin "referenssisovellus"

- havainnollistaa, toimii mallina

https://github.com/ohjelmistotekniikka-hy/python-todo-app

15

TodoApp: vaatimusmäärittelyä

Sovelluksen avulla voi pitää kirjaa tekemättömistä töistä (todoista)

Kaksi käyttäjäroolia. Mitä toiminnallisuutta nämä tarvitsevat?

- tavalliset käyttäjät
- ylläpitäjät

16

Tavalliset käyttäjät voivat:

- luoda järjestelmään käyttäjätunnuksen
- kirjautua järjestelmään, jonka jälkeen he voivat:
- nähdä omat tekemättömät työnsä (todot)
- luoda uusia todoja
- merkitä todoja tehdyiksi, jolloin ne häviävät listalta

Ylläpitäjät voivat:

- tarkastella tilastoja
- poistaa käyttäjätunnuksia

17

Toimintaympäristö:

- toteutetaan Pythonilla ja käyttöliittymä Tkinterillä
- käyttäjä- ja todotiedot tallennetaan paikallisen koneen levylle
- ohjelmiston tulee toimia osaston Linux-työasemilla

Käyttöliittymä:

- vapaamuotoinen luonnos (piirros) valikoista ja niiden välisistä siirtymistä

Tehokkuuteen, kustannuksiin ym. ei oteta kantaa..

18

TodoApp: suunnittelusta

Kurssimateriaalissa esitellään muutamia ohjelmistotuotannossa hyödyllisiä periaatteita, joita TodoApp:issakin on noudatettu. Esimerkiksi:

- *käyttöliittymä on erotettu sovelluslogiikasta* (vrt. kerrosarkkitehtuurit)
- luokat, metodit ja funktiot toteuttavat ns. *single responsibility* -periaatetta: yhden komponentin ei tulisi tehdä liian montaa asiaa

19

TodoApp: yksikkötestaus

Yksikkötestauksessa luokille tehdään testiluokkia, jotka sisältävät testimetodeja

- tyypillisesti näitä metodeita kertyy paljon

TodoAPP:in testiluokat hakemistossa python-todo-app/src/tests/

- esim. luokka todo_service.py ja vast. testiluokka todo_service_test.py

Miksi yksikkötestausta testiluokkineen?

Graafista käyttöliittymää ei kurssilla tarvitse yksikkötestata

20

Kurssin aikataulu

viikko 1: palautuksen deadline ti 7.11.
viikko 2: palautuksen deadline ti 14.11.
viikko 3: palautuksen deadline ti 21.11.
viikko 4: palautuksen deadline ti 28.11.
viikko 5: palautuksen deadline ti 5.12.
viikko 6: palautuksen deadline ti 12.12.
viikko 7: loppupalautuksen deadline pe 22.12.

viikkoharjoitukset ("laskarit")

harjoitustyö

Deadlinet aina klo 23:59. *Myöhästyneitä palautuksia ei huomioida.*

Palautetta ja pisteytys aina vast. viikon loppuun mennessä (paitsi viimeisen deadlinen kohdalla)

21

Välineitä

Github: kaikki palautukset tänne. Myös kurssimateriaali täällä.

Labtool: pisteet ja palaute näkyvät täällä viikoittain

Moodle: ajankohtainen vasta kurssin lopulla. Suuria kielimalleja koskeva kysely, palautepisteen haku. Täällä näet lopulliset pisteesi tammikuussa.

Unittest yksikkötestaukseen

Docstring koodin kommentointiin

Poetry ja Invoke ohjelmiston hallinnointiin VS coden (tms.) ulkopuolella (kirjastojen lataus, koodin ja testien suorittaminen)

Pylint koodaustyylin laadunvarmistukseen

22

1. viikko

Kurssimateriaalissa harjoituksia, aiheena komentorivi, Git ja GitHub

Tämän tulisi olla pitkälti tuttua aiemmilta kursseilta (mm. Tietokone työvälineenä); jos näin, voit sivuuttaa

Lue komentorivi- ja versionhallintaharjoitusten tehtävänannot

Jos tuntuu yhtään vieraalta, tee harjoitukset käytännössä!

- tästä ei saa pisteitä, mutta *tarvitset näitä asioita jatkossa*

23

Jos et tee komentorivi- ja versionhallintaharjoituksia:

Kurssimateriaalissa on tehtävänanto, jossa pyydetään mm. luomaan GitHub-repositorio ja lisäämään sinne pari tiedostoa

- jos et osaa suorittaa tehtävänannossa pyydettyjä asioita, toimi siis komentorivi- ja versionhallintaharjoitusten ohjeiden mukaisesti

Tutustu jo alustavasti seuraavan viikon kurssimateriaaliin

- sivun loppupuolelta löydät ohjeita harjoitustyön aiheenvalintaan ja alustavan määrittelydokumentin kirjoittamiseen
- mikään ei estä jo käytännössä tekemästä 2. viikon harjoituksia ja/tai harjoitustyösi määrittelydokumenttia. Ne kuitenkin arvioidaan vasta kurssin toisen viikon jälkeen.

24

Harjoitustyön aiheesta

Millainen aihe itseäsi kiinnostaa?

Vältä kovin "eeppistä" vaatimusmäärittelyä
- perustoiminnallisuuden tulisi olla toteutettavissa nopeasti
- perustoiminnallisuutta tulisi voida laajentaa helposti
- voit sijoittaa joitain toimintoja kategoriaan "jos aikaa jää"

Liian suppeakaan aihe ei saisi olla
- TodoApp:ia vastaava laajuus riittää

25

Harjoitustyön aiheesta

Painotamme seuraavia:
- toimivuus ja varautuminen virhetilanteisiin
- ohjelman selkeä ja tarkoituksenmukainen rakenne ja toteutus, ml. luokkien vastuut
- laajennettavuus ja ylläpidettävyys

Esim. seuraavat eivät ole tällä kurssilla tärkeitä:
tehokkuus, grafiikka, tietoturva, tekoäly...

26

Harjoitustyöltä odotetaan mm.

Selkeää arkkitehtuuria

Sovelluslogiikasta erotettua käyttöliittymää
- mieluiten graafista

Tietojen tallennusta
- tiedostoihin tai vielä paremmin tietokantaan

Valitse aihe, jossa on riittävästi sovelluslogiikkaa: älä keskity liikaa käyttöliittymään tai tietojen tallennukseen

27

Rajauksia

Ohjelmointikieli on Python, GUI-toteutukseen suositellaan Tkinteriä tai Pygamea

Työn tulee toimia ja olla testattavissa osaston Linux-koneilla ja yliopiston Linux-virtuaalityöasemalla, Cubbli Linux -etätyöpöydällä
- ohjelmointiympäristöksi *suositellaankin* Linuxia, vaikkakin myös Windowsilla sinänsä tulisi onnistua

Ei web-sovelluksia

28

Esimerkkejä aiheista

Kortistointi, kirjanpito, sunnittelu:
kirjakortisto, budjetointi, opinnot, kuntoilu, laihdutus, sää, sähkönkulutus...

Opetus- ja harjoitteluohjelmat: kielet, matematiikka...

Simulointi: soittimet, erilaiset laskimet...

Editorit: tekstieditorit, HTML-editorit...

Omaan tieteenalaan ja opintoihin liittyvät ohjelmat

29

Esimerkkejä aiheista

Pelit: reaaliaikaisia tai vuoropohjaisia
- monenlaiset "arcade-tyyppiset" pelit
- lautapelit: tammi, kaksin pelattava shakki ym.
- korttipelit pasianssista pokeriin
- ristinolla, sudoku, muistipelit, tietovisat...

Tai sitten jotain ihan muuta.

30

Dokumentaatio

Seuraa materiaalin ohjeita

Voit myös katsoa TodoApp:in dokumentaatiota: (vähintään) sen tasoista odotetaan
- ei suoraa kopiointia

Koodin muuttujat, luokat ja metodit kirjoitetaan englanniksi, dokumentaatio suomeksi tai englanniksi

31

2. viikko

Harjoituksia: Poetry, testaus, testikattavuus

Harjoitustyön määrittelydokumentti

Aloita harjoitustyön tuntikirjanpito viimeistään nyt

32

3. viikko

Harjoituksia luokka- ja sekvenssikaavioista

Harjoitustyössä mm.
- toteutetaan toiminnallisuutta
- otetaan Poetry käyttöön
- aloitetaan testaus ja generoidaan testikattavuusraportti
- varmistutaan, että toimii osaston koneilla (virtuaalityöasemassa)

33

UML-kaavioista

Unified Modeling Language (1996): joukko ohjelmistojen rakennetta ja käyttäytymistä kuvaavia kaavioita
- näistä tällä kurssilla luokka-, pakkaus- ja sekvenssikaaviot
- havainnollistamiseen suunnittelussa ja myöhemmin
- laajalti ymmärrettyjä
- UML ei ole sidoksissa ohjelmointikieliin tai prosessimalleihin

Kaavioita voi piirtää monin eri tavoin; yksi kätevä tapa on *Mermaid-syntaksi*

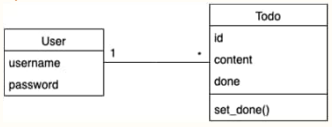
34

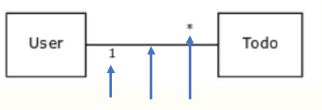
Luokkakaaviot kuvaavat ohjelmiston luokkia ja niiden välisiä yhteyksiä

```
class User:
    def __init__(self, username, password):
        self.username = username
        self.password = password
import uuid

class Todo:
    def __init__(self, content, done=False, user=None, todo_id=None):
        self.content = content
        self.done = done
        self.user = user
        self.id = todo_id

    def set_done(self):
        self.done = True
```





Kaaviossa tärkeitä luokkien suhteet, ml. osallistumisrajoitteet
Metodeita voi kuvata koodin yhteydessä (Docstring)

35

Kurssimateriaalissa vielä mm. riippuvuudet, perintä, pakkauskaaviot...

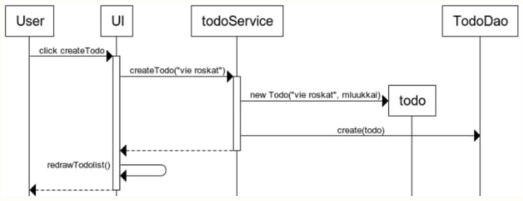
Näillä voidaan kuvata ohjelmiston rakennetta

Miten kuvaisit sen toimintaa?

36

Sekvenssikaaviot: toimenpidesarjan suoritus ajassa

Mikä olio kutsuu minkäkin mitäkin palvelua, missä järjestyksessä kutsutaan ja millä ehdoilla



- aika etenee ylhäältä alas
- metodikutsut kuvataan nuolilla
- metodien suoritusta kuvataan yleensä "aktivaatiopalkeilla"
- paluuarvot merkitään katkoviivoilla

(UML tarjoaa muitakin käyttäytymiskaavioita)

37

Viikot 4 - 7

Edistetään harjoitustyötä ohjeiden mukaisesti...

38

Arviointi

Viikkodeadlinet 16 p. (osa viikkopisteistä tulee laskareista)

Koodikatselmointi 2 p.

Dokumentaatio 13 p.

Automatisoitu testaus 5 p. *loppupalautus*

Lopullinen ohjelma 24 p. (laajuus, ominaisuudet ja koodin laatu)

Yht. max. 60 p. + 1 lisäpiste palautteesta.

Arvosana 1: 30 p., arvosana 5: n. 55 p.

Läpipääsyyn vaaditaan lisäksi vähintään 10 p. lopullisesta ohjelmasta.

39

Lopullinen ohjelma tarkemmin:

Käyttöliittymä 4 p.

- hyvin yksinkertainen tekstikäyttöliittymä 0 p. ... laaja GUI 4 p.

Tiedon pysyväistallennus 4 p.

- tiedosto 1 - 2 p.; tietokanta/Internet 3 - 4 p.

Ohjelman lajuus 4 p.

- TodoApp laajuudeltaan 2 p.

Sovelluslogiikan kompleksisuus 3 p.

Ulkoisten kirjastojen hyödyntäminen 1 p.

Release 1 p.

Koodin laatu 5 p.

Virheiden käsittely 2 p.

40

Muutama varoitus...

1. Muista, että harjoitustyösi täytyy toimia osaston Linux-koneilla
2. Muista, että aika kuluu nopeasti
3. Älä plagioi
4. Älä riko tekijänoikeuksia
5. Ole varovainen ChatGPT:n ja vastaavien kanssa

41

Osaston Linux-koneet

Ympäristöissä on eroja; ei riitä, että työsi toimii omalla koneellasi!

Harjoitustyösi pitää pystyä joka viikko suorittamaan, kääntämään ja testaamaan komentoriviltä käsin osaston Linux-koneilla

- muuten työtäsi ei tarkasteta ja menetät viikon/loppupalautuksen pisteet

Voit testata (ja miksei toteuttaakin) ohjelmistosi toimintaa yliopiston Linux-virtuaalityöasemalla

42

Aika kuluu nopeasti

Seitsemän viikkoa ei ole paljoa

Kaikkea ei kannata jättää deadlineja edeltäviin iltoihin!

Mieti jo etukäteen, missä kohtaa harjoitustyötä saattaa tulla ongelmia

Työ on saatava valmiiksi kurssin aikana ja sitä on toteutettava tasaisesti, muuten kurssi katsotaan keskeytetyksi

- samaa ohjelmaa ei voi jatkaa seuraavalla kurssilla, vaan työ on aloitettava uudella aiheella alusta

43

Älä plagioi

Verkosta tai kavereilta löytyneiden vastausten kopiointi ja niiden palauttaminen omina aikaansaannoksina on kiellettyä

Yliopistolla uudet ohjeet, plagiointiin suhtaudutaan varsin ankarasti

- vilpillä voi olla vakavia seurauksia
- uhkana koko kurssisuorituksen hylkääminen tai jopa opiskelijan määräaikainen erottaminen

https://studies.helsinki.fi/ohjeet/artikkeli/mita-ovat-vilppi-ja-plagiointi

44

Älä riko tekijänoikeuksia

...tai muita immateriaalioikeuksia

Muista, että mitä tahansa verkosta löytynyttä ei saa käyttää omassa työssä

Tarkista lisenssit. Sallivatko ne käytön?

Koskee monenlaista materiaalia: koodia, kuvia, tekstiä...

Harjoitustyöt lähtökohtaisesti julkisia GitHubissa. On omalla vastuullasi, ettet riko tekijänoikeuksia!

45

ChatGPT

...ja yleensäkin suuret kielimallit ja "vastaavat tekoälypohjaiset välineet" (Bing Chat, Google Bard, GitHub Copilot yms.)

ChatGPT yms. sopivat (varauksin ja omalla vastuulla) mm.

- virheidenjäljitykseen
- itse kirjoitetun koodin paranteluun
- dokumentaation luonnosteluun ja hiomiseen
- tiedonhakuun ja neuvojen kysymiseen
- ohjelmakoodin selittämiseen...

Entä harjoitustyön koodin ja dokumentaation generointi?

46

Kurssisivulla: "sallittu ohjelmakoodin generointiin"

...mutta muista, ettei ole mitään takeita, että generoitu koodi todella tekee mitä odotetaan. (Kielimalleihin liittyy muitakin ongelmia, esim. avoimia tekijänoikeuskysymyksiä.)

Monilla kursseilla generoinnin täyskielto

47

Kurssisivulla: "sallittu ohjelmakoodin generointiin"

...mutta muista, ettei ole mitään takeita, että generoitu koodi todella tekee mitä odotetaan. (Kielimalleihin liittyy muitakin ongelmia, esim. avoimia tekijänoikeuskysymyksiä.)

Monilla kursseilla generoinnin täyskielto

POIKKEUS: *yksikkötestejä* ei saa generoida
Lisäksi:

- dokumentaatiota, ml. kaaviot, ei saa sellaisenaan copy-pastettaa ChatGPT:stä tai vastaavista
- viikkoharjoituksissa generoitu vastaus = 0 p.

(Erittäin ahkeran ohjelmakoodin generoinnin täytynee jossain kohtaa vaikuttaa meilläkin jonkin verran pisteytykseen... Tämä raja ei kuitenkaan tule ihan pian vastaan.)

48

Kurssisivulla: "kielimallien käytöstä pitää raportoida"

Jos liität työhösi generoitua koodia sellaisenaan tai vähäisin muutoksin, lisää kommentit

- kommentit generoidun koodinpätkän alkuun ja loppuun
- muista, että myös generoidun koodin esittäminen omana on plagiointia

Lisäksi lopussa on pieni selvitys ChatGPT:n ja vast. tekoälypohjaisten välineiden käytöstä

- kirjaattehan matkan varrella muistiin, että miten mahdollisesti olette näitä käyttäneet ja millaisin kokemuksin
- osa dokumentaatiota, mutta toteutetaan kyselylomakkeella
- tuloksia on tarkoitus myös käyttää pienimuotoisessa tutkimuksessa

49

Ongelmia?

Viikkopalautuksista tulee *lyhyttä* palautetta, mutta *varsinaista tukea* on tarjolla:

1. pajassa (kävele sisään paja-aikoina)
2. Discord-kanavalla (kirjoita kysymys)

Lisäksi Labtooliin voi kirjoittaa kommentteja, jos viikottaiset pisteet arveluttavat

Pajassa saat parhaiten apua!

50