Sissejuhatus tarkvaraarendusse

Martti Raavel

martti.raavel@tlu.ee

Esimene loeng

- Sissejuhatus
- Tarkvara
- Tarkvaraarendus
- Tarkvaraarenduse elutsükkel
- Git
- Github
- .gitignore
- Markdown
- Github issue

Sissejuhatus

- Aine ülesehitus
- Kodused tööd
- Hindamine

Aine ülesehitus

- Hindeline
- Loengud kohapeal
- 4 loengut + eksam
- Materjalid Github-is
- Salvestused
- Kodused tööd

Kodused tööd

- Kohustuslikud
- **Tähtajaks!**
- Vigade leidmine materjalidest annab lisaboonust
- Materjalide täiendamine annab lisaboonust

Hindamine

- Hindamisele pääsemise eeldus on aktiivne osalemine loengutes ja **koduste tööde õigeaegne esitamine**
- Eksam
  - kodused tööd
  - Githubi kasutamine õppeainetes
  - Eneseanalüüs

Mis on tarkvara?

Tarkvara on juhiste või programmide kogum, mis on loodud arvutisüsteemis konkreetsete ülesannete või funktsioonide täitmiseks. See on programmide, andmete ja juhiste kogum, mis ütleb arvutile, mida ja kuidas teha.

Tarkvara liigid

- operatsioonisüsteemid;
- rakendustarkvara;
- programmeerimiskeeled;
- utiliiditarkvara;
- jne.

Tarkvara eesmärk

- andmete haldamine;
- dokumentide loomine;
- graafika kujundamine;
- mängude mängimine;
- jne.

Avatud vs suletud lähtekoodiga tarkvara

- Mis neil vahet on?
- Kumb lähenemine on parem?

Kust tarkvara saab?

- Osta
- Ise teha
- Lasta teha

Plussid/miinused?

Mis on tarkvaraarendus?

Tarkvaraarendus on tarkvararakenduste kavandamise, loomise, testimise ja hooldamise protsess. See hõlmab programmeerimiskeelte, tarkvaraarendustööriistade ja parimate tavade kasutamist, et luua tarkvara, mis vastab konkreetsetele nõuetele ja lahendab konkreetseid probleeme.

Tarkvaraarenduse elutsükkel

Tarkvaraarenduse elutsükkel (Software Development Life Cycle - SDLC) on protsess, mida tarkvaratööstus kasutab kvaliteetse tarkvara kavandamiseks, arendamiseks ja testimiseks. SDLC eesmärk on toota kvaliteetset tarkvara, mis vastab või ületab klientide ootusi, jõuab lõpule ettenähtud aja ja kuluprognooside jooksul.

Tarkvaraarenduse elutsükkel

- Planeerimine
- Nõuete määramine
- Disain
- Programmeerimine
- Testimine
- Evitamine ja hooldus

Planeerimine

- Mida?
- Miks?
- Kelle jaoks?

Nõuete määramine

- Täpsemalt mida vaja
- Prioritiseerimine
- Soov ei võrdu vajadus
- Kliendi kinnitus

Disain

- Arhitektuur
- UX
- UI

Programmeerimine

- Mis operatsioonisüsteemile?
- Mis keeles?

Testimine

- Manuaalne testimine
- Automaattestimine

Töösse andmine

- CI/CD
- Koolitus
- Dokumentatsioon

Hooldus

- Monitoorimine
- Vigade parandus

Kuidas kirjutatud koodi kaitsta ja hallata?

- Palju faile
- Palju muudatusi
- Rohkem kui üks arendaja/osapool

Versioonikontroll

Versioonikontroll on süsteem, mis jälgib ja haldab aja jooksul failides tehtud muudatusi.

Miks versioonikontroll?

- Koostöö
- Ajalugu ja jälgimine
- Muudatuste tagasivõtmine
- Hargnemine ja ühendamine

Git

Git on hajutatud versioonihaldussüsteem, mis on loodud tarkvara arendamise käigus lähtekoodi muutuste jälgimiseks.

Git-i sõnavara

- Repositoorium (*Repository*)
- Kloonimine (*Clone*)
- Tõmbamine (*Pull*)
- Haru (*Branch*)
- Kinnitus? (*Commit*)
- Tõukamine (*Push*)
- Tõmbetaotlus (*Pull request*)
- Ühendamine (*Merge*)

Git-i töövoog

Git-i töövoog on parimate tavade ja juhiste kogum *Git*-i kasutamiseks koodimuudatuste haldamisel. *Git*-i töövooge on palju, kuid kõige levinumat neist nimetatakse *feature branch flow*-ks.

Feature branch flow

- Loo uus haru (*branch*)
- Kirjuta koodi
- Testi kood
- *Commit*i muudatused
- Push
- Pull request
- Merge
- Kustuta haru

Harud

- main (master)
- dev
- test
- alamharud

Github

GitHub on veebipõhine platvorm, mida kasutatakse versioonikontrolliks ja koostööks tarkvara arendamiseks. Github pakub Giti versioonikontrollisüsteemi kasutavate tarkvaraarendusprojektide hostimisteenust.

Gtihubi funktsioonid

- Hoidla majutus
- Koostöö tööriistad
- Juurdepääsukontroll
- Integratsioonid
- Sotsiaalsed funktsioonid

Githubi kasutamine

- git CLI (*Command Line Interface*)
- Graafilise kasutajaliidesega tööriist
  - Github Desktop

Markdown

Markdown on märgendikeel, mis võimaldab kasutajatel kirjutada lihtteksti ja vormindada seda lihtsa süntaksiga, et luua dokumente, mida on lihtne lugeda ja kirjutada.

> Markdown'i kirjutamise lihtsustamiseks on hea kasutada VS Code-i laiendit `Markdown All in One`

Githubi kasutamise harjutamine 1

- Loo oma organisatsiooni alla repositoorium nimega SJTA
- Klooni see oma arvutisse
- Lisa sinna `README.md` fail ja lisa sinna pealkiri `Sissejuhatus tarkvaraarendusse`
- Täida loodud fail mingi sisuga
- Tee *commit*
- Push

Githubi kasutamise harjutamine 2

- Lisa oma organisatsiooni uus liige, kes hakkab edaspidi Sinu kodustele töödele `Code Review` -d tegema
- Tee SJTA repositooriumisse uus haru nimega `test`
- Lisa repositooriumisse uus fail nimega `test.md` ja lisa sinna pealkiri `Test`
- Tee `commit`
- Tee `push`
- Tee `Pull Request` ja lisa ülevaatajaks varem lisatud uus organisatsiooni liige
- Kui ülevaataja nõuab muudatusi, siis tee muudatused, `commit` -i, `push` -i ja küsi uuesti ülevaatust
- Tee `Review` Sinule määratud `Pull Request` -ile
  - Vastavalt vajadusele `Request changes` või `Approve`
  - Peale `Approve` lisa uueks `Reviewer` iks `mrtrvl`

Kodune töö 1

- Loo oma organisatsioonile `Member` ja `Public` vaated kasutades 2. Githubi harjutuses näidatud töövoogu
    - Member vaatesse lisa minimaalselt oma nimi ja pilt
- Loo oma organisatsiooni alla repositoorium nimega `Programmeerimine-I` ja lisa sinna `README.md` fail
    - Sellesse repositooriumisse tuleb edaspidi lisada kõik `Programmeerimine I` loengus kirjutatud kood ja kodused tööd
- Loo oma organisatsiooni alla repositoorium nimega `Veebirakendused` ja lisa sinna `README.md` fail
    - Sellesse repositooriumisse tuleb edaspidi lisada kõik `Veebirakendused ja nende loomine` loengus kirjutatud kood ja kodused tööd

Kodused tööd üldiselt

Kodused tööd sisaldavad üldiselt Gihtubi kasutamist erinevates õppeainetes

- Programmeerimine I loengutes kirjutatud kood
- Programmeerimine I kodused tööd
- Veebirakendused ja nende loomine õppeaine kood
- jms