Latvijas Lauksaimniecības universitāte
Informācijas tehnoloģiju fakultāte

Apstiprināta
Datoru sistēmu katedras
akadēmiskā personāla sēdē
2018. gada ___. _____

Profesionālās kvalifikācijas prakse

Professional Qualification Practice

Jelgavā

Programma

Studiju kursa kodi LLU IS reģistrā InfTP009, InfTP010
26 KP (1040 h): lekc.0 h, lab.d. 0 h, ieskaite
Izstrādāja: Andrejs Paura Datoru sistēmu katedras vieslektors
Tatjana Rubina Datoru sistēmu katedras viesdocents
Obligātais kurss Profesionālās augstākās izglītības bakalaura studiju programmā „Informācijas tehnoloģijas ilgtspējīgai attīstībai".
Kursa apguve tiek plānota 2 semestru laikā, katrā no tiem secīgi iekļaujot vienu daļu:
1.daļa: 16 KP(640h), ieskaite
2.daļa: 10 KP(400h), ieskaite

Anotācija

1.daļa

Prakses 1.daļas pamatmērķis ir programmēšanas inženiera pienākumu praktiska veikšana reālos programmizstrādes apstākļos pieredzējuša programmētāja uzraudzībā.

2.daļa

Prakses 2.daļā students turpina apgūt programmēšanas inženiera pienākumus, bet students izvēloties prakses vietu, īpašu uzmanību pievērš savam studiju nobeiguma kvalifikācijas darbam. Tāpēc students prakses vietas izvēli un praksē veicamo darbu saturu cenšas saskaņot ar kvalifikācijas darba sastāvā noteiktajiem uzdevumiem.

Annotation

1st part

The aim of first part is to discharge practically programming engineer duties in real circumstances under supervision of an experienced programmer.

2ndpart

During the second part students keep performing the tasks of programming engineer. Choosing a place for practice, special attention is paid to the final paper of the studies. This is why students try to integrate the content of work during the practice and tasks of the bachelor paper.

Studiju kursa apgūšanas mērķis

Prakses pamatmērķis ir programmēšanas inženiera pienākumu praktiska veikšana reālos programmizstrādes apstākļos pieredzējuša programmētāja uzraudzībā.

Studiju kursā sasniedzamie studiju rezultāti (zināšanas, prasmes un kompetence)

Prakses rezultātā studenti iegūst:

- zināšanas par programmēšanas inženiera pienākumu praktisku veikšanu reālos programmizstrādes apstākļos;
- prasmes izvēlēties uzdevuma risināšanai adekvātas metodes un līdzekļus, sagatavot programmatūras prasību specifikāciju, izstrādāt projektējumu un kodēt, atkļūdot un testēt programmas;
- kompetences, strādājot grupā vai veicot darbu patstāvīgi, izmantot mūsdienīgas programmatūras izstrādes metodes un labo programmēšanas stilu, kā arī pārliecināt citus un argumentēt savu viedokli.

The aim of study course

The objective of the Practice is to work as software engineer in real software development conditions under the supervision of an experienced programmer.

Learning Outcomes (knowledge, skills and competence)

As a result of practice students get

- knowledge about discharge of programming engineer duties in actual conditions of software development;
- skills in suitable task solving methods' and tools' choosing, in developing software requirements' specification, in design description, in programs' coding, debugging and testing;
- competence, working in teams or independently, in up-to-date software development methods' and good programming style using, as well as in persuading the others and arguing their own point of view

Studiju kursa saistība ar citiem studiju kursiem

Lai varētu veiksmīgi realizēt praksi, iepriekš jābūt apgūtiem šādiem studiju kursiem: Programminženierijas metodes (DatZ2057, DatZ2058), Sistēmanalīze un modelēšana (InfT2041), Programmēšanas pamati (DatZ1009, DatZ1010), Datu bāzu tehnoloģijas (DatZ2004, DatZ2005), Programmatūras testēšana (InfT4006), Lielās datu bāzes (DatZ3003, DatZ3004).

Patstāvīgā darba veidi un to apjoms

Pēc prakses pabeigšanas students sagatavo Prakses atskaiti (apjoms vismaz 3. lpp.), kurā apraksta programmatūru, informācijas sistēmu vai informācijas tehnoloģiju izstrādes, uzturēšanas vai pakalpojumu sniegšanas darbus, ar kuriem bija saistīts prakses uzdevums, raksturo savu līdzdalību kopējā procesā un konkrēti paveikto uzdevumu un sagatavo īsu prezentāciju (apm. 5 min), ar kuru notiek uzstāšanās prakses aizstāvēšanas seminārā.

Studiju rezultātu kontrole

Prakses laikā regulāri (reizi nedēļā) jāaizpilda prakses dienasgrāmata interneta vietnē http://prakses.itf.llu.lv, kurā īsi apraksta kārtējā nedēļā paveikto.

Pēc katras prakses daļas notiek uzstāšanās prakses aizstāvēšanas seminārā. Seminārā piedalās visi praksi pabeigušie studenti, studiju programmu direktori, ITF prakses vadītāji un prakses metodiskie vadītāji.

Katras kursa daļas prakses rezultāti tiek novērtēti kā ieskaite. Ieskaite par atbilstošu kursa daļu ir nokārtota, ja students ir regulāri aizpildījis prakses dienasgrāmatu, sagatavojis pārskatu un sekmīgi to aizstāvējis prakses seminārā.

Nokavēto nodarbību atstrādāšanas prasības un kārtība

Neīstenotās prakses vai nepilnā apjomā īstenotās prakses atstrādāšana tiek realizēta saskaņā ar LLU normatīvajiem dokumentiem par akadēmisko parādu kārtošanu.

Programmas izvērsts saturs

Ievads. Prakses vietu izvēlas un praksi īsteno saskaņā ar „ *Informācijas tehnoloģiju fakultātes bakalaura studiju prakses nolikumu"*. Kad students ir izvēlējies prakses vietu, tad praksi īsteno

saskaņā ar prakses līgumu, kuru noslēdz starp Latvijas Lauksaimniecības universitāti, prakses vietas iestādi un studentu.

Praksē students iepazīstas ar programmētāja darba specifiskajām prasībām, lai, turpinot studijas pēc prakses, lielāku uzmanību varētu pievērst praksē atklātajām nepilnībām zināšanās. Prakses laikā students izvēlas kvalifikācijas darba tēmu, praksē veicamo darbu saturu un iespēju robežās cenšas saskaņot ar kvalifikācijas darba sastāvā noteiktajiem uzdevumiem.

Praktikanta pienākumi un uzdevumi

1. **Kodēšanas darbu** izpildes laikā veic sekojošus uzdevumus:
   1.1. Lasīt un saprast programmatūras projektējuma aprakstus.
   1.2. Analizēt ieejas un izejas datus.
   1.3. Konfigurēt izstrādes vidi.
   1.4. Rakstīt programmas kodu saskaņā ar projektējumu un kodēšanas vadlīnijām.
   1.5. Konstruēt algoritmus.
   1.6. Lasīt un analizēt svešus programmu tekstus.
   1.7. Veidot lietotāja saskarni.
   1.8. Atkļūdot programmas un veikt vienību testēšanu.
   1.9. Analizēt programmas izpildes laiku un to optimizēt.
   1.10. Dokumentēt kodu.
   1.11. Veidot programmatūras instalāciju.
   1.12. Veidot iebūvēto palīdzības sistēmu.
2. **Projektēšanas darbu** izpildes laikā veic sekojošus uzdevumus:
   2.1. Lasīt un saprast programmatūras prasību specifikācijas.
   2.2. Iepazīties ar programmatūras projektējuma apraksta standartiem.
   2.3. Veidot un aprakstīt programmatūras arhitektūru.
   2.4. Analizēt dažādus tehniskos risinājumus un izvēlēties piemērotāko.
   2.5. Veidot datu konceptuālo un fizisko modeli.
   2.6. Veidot realizācijas modeli (klašu un/vai funkciju hierarhiju).
   2.7. Konstruēt un aprakstīt algoritmus.
   2.8. Projektēt lietotāja saskarnes.
   2.9. Sagatavot programmatūras projektējuma apraksta dokumentu.
   2.10. Apstrādāt izmaiņu pieprasījumus un problēmu ziņojumus.
3. **Uzturēšanas darbu** izpildes laikā veic sekojošus uzdevumus:
   3.1. Lasīt un saprast uzturamās sistēmas dokumentāciju un kodu.
   3.2. Apstrādāt izmaiņu pieprasījumus un problēmu ziņojumus.
   3.3. Veikt izmaiņu ietekmes analīzi.
   3.4. Veikt izmaiņas programmatūrā.
   3.5. Veikt uzturamās programmatūras konfigurācijas pārvaldību.
   3.6. Sistematizēt uzturēšanas gaitā uzkrāto atbalsta informāciju.
   3.7. Konsultēt programmatūras lietotājus.
4. **Programmatūras ieviešanas darbu** izpildes laikā veikt sekojošus uzdevumus:
   4.1. Veikt vides sagatavošanu programmatūras uzstādīšanai.
   4.2. Veikt datu pārnešanu.
   4.3. Izpildīt programmatūras uzstādīšanu un parametrizēšanu.
   4.4. Iepazīties ar lietotāja dokumentāciju.
   4.5. Sniegt konsultācijas programmatūras ieviešanas laikā.
5. **Programmatūras testēšanas darbu** izpildes laikā veikt sekojošus uzdevumus:
   5.1. Sagatavot testēšanas plānu.
   5.2. Sagatavot testēšanas specifikāciju.
   5.3. Analizēt programmas kodu.
   5.4. Sagatavot testpiemērus.
   5.5. Sagatavot testēšanas vidi.

5.6. Izpildīt testpiemērus.
5.7. Pierakstīt testēšanas gaitu un rakstīt problēmu ziņojumus.
5.8. Analizēt kļūdu avotus (prasības specifikācija, projektējuma apraksta, u.c.).
5.9. Reproducēt lietotāja konstatētās kļūdas.
5.10. Sagatavot testēšanas pārskata dokumentu.

6. **Prasību specificēšanas darbu** izpildes laikā veikt sekojošus uzdevumus:
   6.1. Iepazīties ar esošo pasūtītāja programmatūru.
   6.2. Analizēt prasību realizācijas iespējas.
7. **Lietotāja dokumentācijas gatavošanas** laikā veikt sekojošus uzdevumus:
   7.1. Iepazīties ar lietotāja dokumentācijas standartiem.
   7.2. Iepazīties ar lietotāja darījumu terminoloģiju.
   7.3. Rakstīt un noformēt lietotāja dokumentācijas tekstu.
   7.4. Saskaņot lietotāja dokumentāciju ar iebūvēto palīdzības sistēmu (Help).
8. **Programmatūras projekta plānošanas darbu** izpildes laikā veikt sekojošus uzdevumus:
   8.1. Prognozēt darba uzdevuma darbietilpību un izpildes laiku.
   8.2. Veikt individuālā darba plānošanu un kontroli.
   8.3. Piedalīties projekta gaitas izpildes apspriešanā.
   8.4. Izstrādāt programmēšanas vadlīnijas.

Mācību un metodiskās literatūras saraksts

Pamatliteratūra

1. Prakšu nolikums (Pielikums Senāta 12.11.2014. lēmumam Nr. 8 – 130). Jelgava: LLU 2014.
2. Informācijas tehnoloģiju fakultātes bakalaura studiju prakses nolikums. Jelgava: ITF 2011.
3. Programmēšanas inženiera profesijas standarts, apstiprināts ar Izglītības un zinātnes ministrijas 2009.gada 17.jūnija rīkojumu Nr. 5
4. Pressman R.S. Software Engineering. A Practitioner's Approach. European Adaptation. 7th edition, adapted by Darrel Ince. 2010 (LLUB)
5. Coronel C. Database principles: fundamentals of desing, implementation and management. Andover: Cengage Larning, 2013.
6. ANSI/IEEE Std 1008(R1993), IEEE Standard for Software Unit Testing (Standartizācijas biroja lasītava)
7. ANSI/IEEE Std 829-1998, IEEE Standard for Software Test Documentation (Standartizācijas biroja lasītava)
8. BS 7925-2.Standard for Software Component Testing. [tiešsaiste]. Pieejams: http://www.testingstandards.co.uk/Component%20Testing.pdf [Skatīts 2018.03.12.]

Papildliteratūra

1. IEEE Computer Society SWEBOK v3.0 Guide to the Software Engineering Body of Knowledge/ Project Management Institute 2014. [tiešsaiste]. Pieejams: www.swebok.org. [Skatīts 2018.03.12.]
2. ISO/IEC 12207. 2008. Systems and software engineering. Software life cycle processes (LVS Standartu lasītava)
3. ISO/IEC 26514:2008. Systems and software engineering — Requirements for designers and developers of user documentation, 2008. (LVS Standartu lasītava)
4. Troelsen A., C# 2010 and the.NET Platform. Apress, 2010.
5. Oppel A.SQL: a beginners guide New York: McGraw-Hill, 2016. 533 p.

6. Feuerstein S. Oracle PL/SQL Programming, Fifth Edition. Sebastopol: O'Reilly Media, 2009.

Informācija par programmas izstrādāšanu, izvērtēšanu un saskaņošanu

Programmu izstrādāja

_____2018.g. / *A. Paura* /

_____2018.g. / *T.Rubina* /

Saskaņots:

Datoru sistēmu katedras vadītāja

_____2018g. /*T.Rubina*/

Informācijas tehnoloģiju fakultātes Metodiskās komisijas vadītāja

_____2018g. /*L. Zvirgzdiņa*/

Informācijas tehnoloģiju fakultātes dekāns

_____2018.g. /*G. Vītols*/

Andrejs Paura 63005701

Tatjana Rubina 63005701