Co-funded by the
Tempus Programme
of the European Union

ARARAT

ԾՐԱԳՐԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐԻ ՃԱՐՏԱՐԱԳԻՏՈՒԹՅՈՒՆ ՈԼՈՐՏԻ ՈՐԱԿԱՎՈՐՈՒՄՆԵՐԻ ԱՋԳԱՅԻՆ ՇՐՋԱՆԱԿ

ԱՐԱՐԱՏ

«Համալսարան-գործատու»
հայկական համակարգող գործակալություն

www.ararattempus.org

With the support of the Tempus Programme of the European Union

«Համալսարան-գործատու» հայկական համակարգող գործակալություն

ԾՐԱԳՐԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐԻ ՃԱՐՏԱՐԱԳԻՏՈՒԹՅՈՒՆ ՈԼՈՐՏԻ ՈՐԱԿԱՎՈՐՈՒՄՆԵՐԻ ԱԶԳԱՅԻՆ ՇՐՋԱՆԱԿ

Մասնագիտական կրթության որակի ապահովման ազգային կենտրոն
2016

Co-funded by the
Tempus Programme
of the European Union

Այս ծրագիրը ֆինանսավորվել է Եվրոպական Հանձնաժողովի աջակցությամբ: Սույն հրատարակությունը արտացոլում է միայն հեղինակների տեսակետները, և Հանձնաժողովը պատասխանատվություն չի կրում դրանում պարունակվող տեղեկատվության օգտագործման համար:

Բովանդակություն

1. ՆԵՐԱԾՈՒԹՅՈՒՆ

Այս փաստաթուղթը ներկայացնում է ՏԵՄՊՈՒՍԻ ԱՐԱՐԱՏ («Համալսարան-գործատու» հայկական համակարգող գործակալություն) ծրագրի շրջանակներում մշակված Ծրագրային համակարգերի ճարտարագիտության Ոլորտային որակավորումների ազգային շրջանակը (ՈՈԱՇ):

ՈՈԱՇ-ի մշակման անհրաժեշտությունը պայմանավորված է հետևյալ խնդիրներով:

1. Որակավորումների վերաբերյալ կարծիքի ներառում որակավորումների պահանջներում:

ՀՀ-ն մշակել է Որակավումների ազգային շրջանակը (ՈԱՇ) և այժմ գտնվում է բուհական կրթությունը ՈԱՇ-ի պահանջներին համապատասխանացնելու փուլում: Որակավորումների ոլորտային շրջանակի մշակումը, որը կնկարագրի ՀՀ-ում Ծրագրային համակարգերի ճարտարագիտության ոլորտում աշխատաշուկայի կարիքները, հիմքեր կստեղծի համակարգի կառուցման համար: ՈԱՇ-ի պահանջները, որպես ուղենիշ ունենալով և դրանք արտահայտելով ոլորտին հատուկ պահանջների տեսքով, ՈՈԱՇ-ը օգտակար և կողմնորոշիչ գործիք կդառնա ուսումնական հաստատությունների և գործատուների միջև երկխոսություն ստեղծելու համար: ՈՈԱՇ-ը կօգնի ուսումնական հաստատություններին ավելի արդյունավետ կազմակերպել կրթությունը և աշխատաշուկայի պահանջներին համապատասխան կրթություն տրամադրել:

2. Գործատուի և կրթության միջև միասնական լեզվական դաշտի ձևավորում

Աշխատելով տարբեր համատեքստերում՝ գործատուն և կրթությունը շատ հաճախ սկսում են խոսել իրար համար անհասկանալի լեզվով: Այստեղ խոսքը գնում է բառերի և դրանց իմաստի

ընկալման տարբերությունների մասին: ՈՈԱԾ-ը պետք է թույլ տա երկու կողմերին հաղթահարել այդ դժվարությունը և խոսել միմյանց հասկանալի հասկացություններով:

3. Որակավորումների ճանաչում՝ ազգային և միջազգային մակարդակներում

Բոլոնյան հռչակագրի կարևոր ուղղություններից մեկը վերաբերում է որակավորումների փոխճանաչման խնդրին: Զևավորվող գլոբալ կրթական ցանցի և աշխատաշուկայի պայմաններում այս խնդիրը չի կորցնում իր արդիականությունը: ՈՈԱԾ-ի ներդրումը մեծ դեր ունի նաև ՀՀ-ում հավատարմագրման գործընթացի կազմակերպման ժամանակ՝ ուղղորդելով արտաքին գնահատման փորձագետներին գործընթացի բարելավմանը միտված փաստերի բացահայտմանը և դասավանդման գործընթացը հայկական իրականության համատեքստին համապատասխանեցնելու վերաբերյալ խորհրդատվությունների տրամադրմանը:

Հաճախ գրավիչ է թվում օգտագործել լավ մշակված և լայն տարածում գտած լավագույն փորձը: Սակայն, իրականացման ժամանակ մի շարք յուրահատկություններ պետք է հաշվի առնվեն, որոնք կախված են ազգային աշխատաշուկայի պահանջներից, ավանդույթներից ու առկա ռեսուրսներից:

ՈՈԱԾ-ը նախատեսված չէ փոխարինելու կրթական ծրագրերի հայտնի չափորոշիչները և ոլորտում ուսումնական պլանների մշակման ուղենիշները: Այնուամենայնիվ, այն արձագանքում է Ծրագրային համակարգերի ճատարագիտական կրթության մեջ առկա բացերին, որոնք դուրս են բերվել ազգային գործատուների հետ հարցազրույցների և քննարկումների ընթացքում: Այսպիսով, Ծրագրային համակարգերի ճարտարագիտության ՈՈԱԾ-ը հանդիսանում է կրթության հիմնական շահագրգիռ կողմերի՝ գործատուի և ակադեմիական դաշտի հետազոտության արդյունքը:

2. ՀԵՏԱԶՈՏՈՒԹՅԱՆ ՆՊԱՏԱԿԸ ԵՎ ՔԱՅԼԵՐԸ

Հետազոտության նպատակն է մշակել Ծրագրային համակարգերի ճարտարագիտություն ոլորտի որակավորումների համապարփակ և ամբողջական նկարագրություն, որը հասկանալի կլինի ինչպես կրթական դաշտի, այնպես էլ գործատուների համար: Ընդ որում, շրջանակը պետք է ունենա այնպիսի կառուցվածք, որը հետագայում կդյուրացնի դրա վերանայման գործընթացը:

Նշված նպատակին հասնելու համար կատարվել են հետևյալ քայլերը՝

1. Գրականության վերլուծություն
2. Հետազոտական շրջանակի սահմանում
3. Հարցում գործատուների, դասախոսների, ուսանողների և շրջանավարտների շրջանում
4. Ֆոկուս խմբեր գործատուների հետ
5. Արդյունքների ընդհանրացում և վերլուծություն

Գրականության վերլուծության արդյունքում հստակեցվել են կրթության և աշխատաշուկայի միջև առկա հիմնական խնդիրները, սահմանվել է շրջանակի կառուցվածքը և հետազոտության մեթոդաբանությունը: Դրանք անցել են քննարկման փուլով, որին մասնակցել է ոլորտի մասնագետների լայն շրջանակ: Համաձայնեցվել և քննարկվել են հետազոտության բոլոր վերոնշյալ բաղադրիչները:

Մեթոդական մասը հստակեցնելուց հետո սահմանվել է աշխատանքների ժամանակացույց և ձևավորել է դաշտային հետազոտություններ իրականացնող խումբ: Դրանում ընդգրկվել են՝ 1 ոլորտի մասնագետ, 1 մեթոդական մասնագետ (ֆոկուս խմբերի և հարցումների մեթոդաբանության իմացությամբ), 2 կրթության ոլորտի մասնագետներ:

Հետազոտության հաջորդ փուլը հարցաշարի իրականացումն է

ուսանողների, շրջանավարտների, դասախոսների և գործատուների շրջանում: Հարցաշարերը ուղարկվել է համապատասխան ռեսպոնդենտներին և ենթարկվել են քանակական վերլուծության:

Ըստ սահմանված ժամանակացույցի հանդիպումներ են տեղի ունեցել Ծրագրային համակարգերի ճարտարագիտության ոլորտի գործատուների հետ, որոնց ընթացքում հավաքագրվել է տեղեկատվություն՝ համաձայն նախապես սահմանված հետազոտական շրջանակի: Հավաքագրված տեղեկատվությունը վերլուծվել է և արվել են համապատասխան հետևություններ:

Հաջորդ բաժիններում կանդրադառնանք վերևում բերված քայլերի արդյունքներին:

3. ՀԵՏԱԶՈՏԱԿԱՆ ՇՐՋԱՆԱԿ

ՈՈԱԾ-ի մշակման համար առաջարկվող մոդելը բաղկացած է հետևյալ հիմնական բաղադրիչներից՝

- Ապրանքի նկարագրություն
- Ապրանքի կամ ծառայության կենսափուլ
  - o Կոմպետենցիաներ
  - o Հմտություններ
- Մասնագիտական մտածողություն
- Գիտելիքի մարմին (BOK)

Ոլորտին հատուկ ապրանքի նկարագրությունը անհրաժեշտ է, որպեսզի շրջանակը ամբողջական պատկերացում տա մասնագետի գործունեության նպատակի մասին: Այն թույլ է տալիս վերացական նկարագրությունները դիտարկել շատ հստակ, բոլորի կողմից հասկանալի, հաճախ նաև շոշափելի առարկաների համատեքստում: Ստացվում է, որ ապրանքի նկարագրությունը շրջանակը օգագործողին թույլ է տալիս ավելի հստակ պատկերացում ունենալ ոլորտում

գործունեություն ծավալող մասնագետի առարկայական նպատակների մասին, իսկ կրթության ընթացքում հանձնարարությունները ձևակերպել այդ կոնտեքստում:

Ոլորտի ապրանքի համատեքստային մոտեցումը արդյունավետ է նաև գործատուների հետ հանդիպումների ընթացքում: Այդ արդյունավետությունը կայանում է հետևյալում: ԵՈՇ-ից ավելի ցածր կանգնած շրջանակներ ժառանգվող հիմնական հատկություններն են՝ ինքնուրույնությունը և համատեքստի բարդությունը: Հետևաբար գործատուների հետ երկխոսության ժամանակ հարկ է պարզել այդ հատկությունների որոշ առանձնահատկություններ: Սակայն գործատուներին դժվար է մտածել այդ հասկացությունների սահմաններում: Փոխարենը, նրանք շատ լավ պատկերացնում են իրենց կողմից արտադրվող ապրանքը: Հենց ապրանքի միջոցով է հնարավոր լինում դուրս բերել ինքնուրույնության և բարդության ցուցանիշները:

ՈՈԱՇ-ը ըստ էության կապող օղակ է աշխատաշուկայի և կրթական համակարգի միջև: Այդ կամուրջի ձևավորման նպատակով անհրաժեշտ է աշխատանքային միջավայրի հատկանիշները տեղափոխել դեպի կրթական միջավայր: Այդ նպատակով կիրառվում է ապրանքի կենսափուլը: Այն թույլ է տալիս ապահովել ոչ միայն աշխատանքային միջավայրի տեղափոխություն այլ նաև հնարավորություն է տալիս գործատուի խոսքը և մոտեցումը բերել կրթություն: Ընդ որում դա անել՝ ընդգրկելով տեխնոլոգիական ողջ ցիկլը: Ներկայիս պայմաններում, երբ պետությունը գնալով ավելի պասիվ դեր է վերցնում, կրթական պատվերի իմաստով, նման մոդելը խիստ արդյունավետ է դառնում: Միաժամանակ կրթական հաստատությունը իր արածի մասին կարծիք հարցնելուց կկարողանա պրակտիկ կողմը ցույց տալ արդեն գործատուի համատեքստում և արձագանքը առավել նպատակային կարող է լինել:

Մյուս կողմից բացարձակ աշխատաշուկայի տերմիններով և

լեզվով խոսելու դեպքում կա վտանգ, որ կրթությունը ամբողջությամբ չի ընկալի շուկայի պատվերը: Ավելին, ՈՈԱԾ-ը ի սկզբանե մտածված է եղել հենց այդ հաղորդակցական արգելքը հաղթահարելու համար: Հետևաբար անհրաժեշտ է, որպեսզի շրջանակը ներառի կրթական դաշտի համար հասկանալի բաղադրիչ:

Հենց այս խնդիրը լուծելու նպատակով շրջանակում ներառվել է ոլորտին հատուկ «գիտելիքի մարմին»: Կրթության խնդիրը կդառնա ուղղորդել՝ արդեն դասական դարձած գիտելիքը, բովանդակությունը դեպի աշխատաշուկայի պահանջներ: Այդ իմաստով հարկ է ուշադրություն դարձնել գնահատման և դասավանդման մեթոդների վրա, որոնք առավելագույնս պետք է մոտեցնել աշխատանքային պայմաններին: Սա նշանակում է, որ կախված կրթական ծրագրի պահանջներից և առանձնահատկություններից անհրաժեշտ է գնահատումը և հանձնարարությունները պլանավորել ապրանքի և դրա շրջափուլի համատեքստում: Մոտեցնել կրթական ծրագիրը դրանց առանձնահատկություններին:

Մասնագիտական մտածողությունը այս մոդելում հանդես է գալիս, որպես մասնագիտական խնդիրներ լուծելու գրավական: Այն ոլորտի մասնագետի ներքին՝ մտավոր վարքն է: Դրանով են տարբեր ոլորտների մասնագետները տարբերվում խնդիրներ լուծելու մոտեցումներում: Նկարագրված մոդելը դյուրացնում է նաև ոլորտային հետազոտությունների գործընթացը: Այն թույլ է տալիս ոլորտային հետազոտություններ իրականացնելիս հստակ բաժանում կատարել գաղափարական կոնստրուկտների մեջ: Օրինակ՝ հարցաշարեր մշակելիս հարցեր տալ միայն ապրանքի կենսափուլերից մեկի մասին: Այն ավելի է հեշտացնում մեթոդաբանության և գործիքների մշակումը և ապահովում է հավաստի արդյունքներով: Այսպիսով, ներկայացված մոդելը հանդես է գալիս, որպես ոլորտի բոլոր էական առանձնահատկությունները ընդգրկող, ամբողջական մոդել:

Առաջարկվող շրջանակը իրենից ներկայացնում է 2 մակարդակի

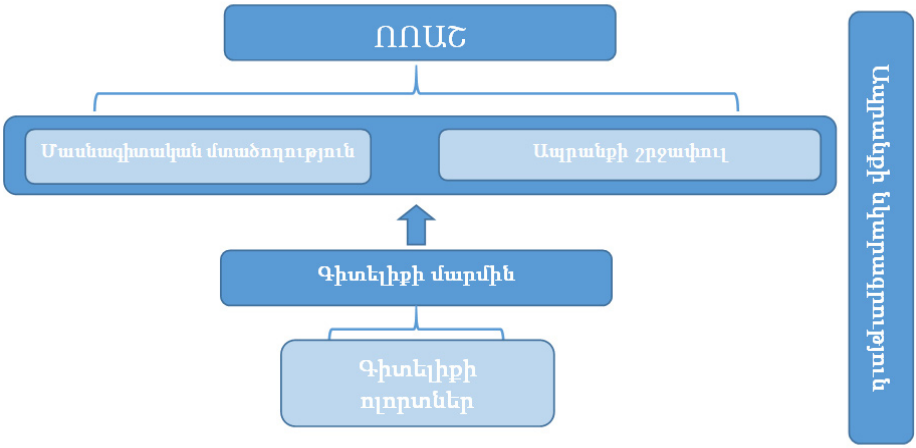
շրջանակների համադրություն ՈԱԾ-ի և ՈՈԱԾ-ի: Ոլորտային շրջանակը (ՈՈԱԾ) բաղկացած է երկու մասից: Առաջինը ունակությունների նկարագրությունն է, իսկ երկրորդը՝ ապրանքի կենսափուլի համար անհրաժեշտ կոմպետենցիաների նկարագրությունը: Անդրադառնանք դրանց առանձին- առանձին:

Ապրանքի կենսափուլի բաղադրիչները ունեն բնային դիզայն և ձևավորվում են ամբողջականության և օպերացիոնալիզացիայի սկզբունքով: Սա նշանակում է, որ ոլորտին վերաբերող կոմպետենցիաների ամբողջությունը դասակարգված է ըստ ապրանքի ստեղծման կենսափուլի: Այդ փուլերը թույլ են տալիս ապահովել, որ ոլորտին բնորոշ բոլոր անհրաժեշտ կոմպետենցիաները ներկայացված կլինեն շրջանակի նկարագրության մեջ: Սա ամբողջականության սկզբունքն է, որի պահպանումը հիմնավորվում է նաև այն հանգամանքով, որ ապրանքի ստեղծման կենսափուլը արտացոլում է աշխատաշուկայի իրական գործընթացները, հետևաբար ավելի է մոտեցնում շրջանակը գործնական կյանքին:

Բացի դա յուրաքանչյուր առանձին փուլի համար նկարագրված են այն բոլոր կոմպետենցիաները, որոնք անհրաժեշտ են այդ փուլի իրագործման համար: Յուրաքանչյուր կոմպետենցիայի համար ներկայացված է դրան համապատասխան հմտությունների շարքը, որը թույլ է տալիս ձևավորել կոմպետենցիան: Այս կառուցվածքի միջոցով պահպանվում է օպերացիոնալիզացիայի սկզբունքը և ավելի վերացական կառույցները ներկայացվում են հասկանալի և համեմատաբար ավելի չափելի միավորների տեսքով:

Ոլորտի մասին ամբողջական պատկերացում ունենալու համար հարկ է սահմանել այդ ոլորտին հատուկ ապրանքները: Ապրանքը, դա այն հիմնական արդյունքն է, որը ստացվում է գործընթացների արդյունքում:

Գծատկեր 1.



4. ԳՐԱԿԱՆՈՒԹՅԱՆ ՎԵՐԼՈՒԾՈՒԹՅՈՒՆ

Առաջին քայլով իրականացվել է դաշտի փաստաթղթային հենքի ուսումնասիրություն, որը նպատակ ուներ պարզելու Ծրագրային համակարգերի ճարտարագիտություն ոլորտի ներկայիս վիճակը և պահանջները:

Ծրագրային համակարգերի ճարտարագիտության ոլորտի ապրանքների մասնագիտական տերմինաբանական բառարանը և սահմանումները ներկայացված են IEEE[1] փաստաթղթում: Կան բազմաթիվ ոչ ֆորմալ մոտեցումներ, որոնք անուղղակիորեն նկարագրում են Ծրագրային համակարգերի ճարտարագիտության ապրանքների առանձնահատուկ հատկանիշները, ինչպես նաև դրանց մշակման և օգտագործման ձևերը:

Ծրագրային համակարգերի ճարտարագիտության համակարգչային գիտության ձեռքբերումների օգտագործումն է ճարտարագիտական համատեքստում՝ ելնելով ծրագրերի մշակման և պահպանման

[1] Աղբյուրը՝ https://www.ieee.org/

կարիքից: Համակարգչային գիտությունը հանդիսանում է ոլորտի մասնագիտական մտածողությունն ու արժեքները: Այն հանդիսանում է ոլորտի մասնագիտական մտածողության գիտական հիմքը: Մասնագիտական մտածողությունը ներկայացված է համակարգչային գիտության կրթական ծրագրերի ուսումնառության ակնկալվող արդյունքներում: Կրթական ծրագրերը ներառում են երկու տեսակի մտածողություն՝ ալգորիթմական մտածողություն և ճարտարագիտական մոտեցում: Համակարգչային գիտությունը գիտելիք և հմտություններ է տալիս ծրագրավորման աշխատանքի համար:

Ճարտարագիտական մոտեցումն իրականացվում է ապրանքի շրջափուլի էլեմենտների նկարագրության ձևով: Լավ հայտնի պրակտիկ մոտեցումներն են՝ CDIO[1] և e-competence[2]:

Գիտելիքի մարմինը այլ ձևով է ներկայացված (SE2014[3], Computing07[4], CS2013[5]):

Ոլորտի իրավիճակի ուսունասիրությունը բավարար հիմք է տալիս լայն համատեքստում ՈՈԱԾ-ի նկարագրման համար: Համատեքստի ուսումնասիրության արդյունքը մոդելի էլեմենտների և սեկտորի իրավիճակի դուրս բերումն է: ՈՈԱԾ-ի կարևոր փոխլրացնող էլեմենտը ազգային կարիքների ուսումնասիրությունն է, որը

[1] Crawley, E.F., Malmqvist, J., Östlund, S., Brodeur, D.R., Edström, K. Rethinking Engineering Education The CDIO Approach, 2014.

[2] Աղբյուրը՝ http://www.ecompetences.eu/

[3] Software Engineering 2014, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series, 23 February 2015, Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery

[4] "Computing 2007" © The Quality Assurance Agency for Higher Education 2007, <www.qaa.ac.uk>

[5] Computer Science Curricula 2013, Curriculum Guidelines for Undergraduate Degree Programs in Computer Science

December 20, 2013, The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society

ամբողջացնում է ՈՈԱԾ-ի նկարագրությունը: Ազգային կարիքների ուսումնասիրությունը և գործատուների կարիքների վերհանումը կատարվել է ոլորտին հատուկ բառապաշարի օգտագործմամբ:

Հավելված 3-ում ներկայացված են գրականության վերլուծության արդյունքում դուրսբերված ոլորտին վերաբերող սահմանումները և նկարագրությունները:

5. ՖՈԿՈՒՍ ԽՄԲԱՅԻՆ ՔՆՆԱՐԿՈՒՄՆԵՐ

ՈԱԾ մշակման համար անհրաժեշտ տեղեկատվությունը ստանալու նպատակով իրականացվել են ֆոկուս խմբային քննարկումներ ՏՏ ոլորտի գործատուների հետ: Գործատուի հետ հաղորդակցումն առավել նպատակային և արդյունավետ դարձնելու համար օգտագործվել է մշակված ՈԱԾ կառուցվածքը ուղղորդելու քննարկումները:

Ֆոկուս խմբային քննարկումները ուղղված էին բացահայտելու՝

1. Ո՞րոնք են մասնագիտությանը բնորոշ ապրանքները և որոնք են դրանց յուրահատուկ պահանջները
2. Որո՞նք են այն կոմպետենցիաները և հմտությունները, որոնք նպաստում են ապրանքի յուրաքանչյուր կենսափուլի ընթացքում բնորոշ խնդիրների իրականացմանը:
3. Ինչպիսի՞ մտածելակերպ է բնորոշ ոլորտի մասնագետին մասնագիտական խնդիրներ լուծելու ընթացքում:

ՈԱԾ մշակման համար անհրաժեշտ տեղեկատվությունը ստանալու նպատակով իրականացվել են ֆոկուս խմբային քննարկումներ ՏՏ ոլորտի գործատուների հետ: Գործատուի հետ հաղորդակցումն առավել նպատակային և արդյունավետ դարձնելու համար օգտագործվել է մշակված ՈԱԾ կառուցվածքը ուղղորդելու քննարկումները:

Ֆոկուս խմբային քննարկումները ուղղված էին բացահայտելու՝

1. Ո՞րոնք են մասնագիտությանը բնորոշ ապրանքները և որոնք

են դրանց յուրահատուկ պահանջները

2. Որո՞նք են այն կոմպետենցիաները և հմտությունները, որոնք նպաստում են ապրանքի յուրաքանչյուր կենսափուլի ընթացքում բնորոշ խնդիրների իրականացմանը:
3. Ինչպիսի՞ մտածելակերպ է բնորոշ ոլորտի մասնագետին մասնագիտական խնդիրներ լուծելու ընթացքում:
4. Ինչպիսի՞ գիտելիք է անհրաժեշտ ոլորտի մասնագետին մասնագիտական խնդիրներ լուծելու ընթացքում:

Հավելված 2-ում ներկայացված է քննարկումների վարման ուղեցույցը: Այսպիսով հավաքագրված տվյալները հետազոտական խումբը ներառել է ՈՈԱԾ-ի շրջանակում:

Հետազոտության նպատակներով իրականացվել են 6 ֆոկուս խումբ քննարկումներ ոլորտի գործատուների հետ: 30 ներկայացուցիչ 12 SS ընկերություններից մասնակցել են քննարկումներին: Ընկերությունները տարբերվում են չափերով (աշխատակիցների թիվ) և ոլորտներով (ապրանքների տեսակ): Մասնակից կազմակերպությունները Ծրագրային համակարգերի ճարտարագիտության ոլորտում բավականին ակտիվ են, ունեն ազգային և միջազգային աշխատաշուկայում ներգրավվածության փորձ, ինչպես նաև ապրանքների բազմազանություն: Ընկերությունների ցանկը, որոնք մասնակցել են քննարկմանը ներկայացված է Հավելված 3-ում:

Առաջին հանդիպումներն ու հարցազրույցները եղել են ՀՀ-ում Ծրագրային համակարգերի ճարտարագիտության ոլորտի գլխավոր գործադիր տնօրենների հետ: Ինֆորմացիոն Տեխնոլոգիաների Ձեռնարկությունների Միությունը հրավիրել էր մի շարք կազմակերպությունների՝ մասնակցելու ֆոկուս խմբային քննարկմանը:

Նախնական արդյունքները ցույց տվեցին, որ ՀԱԱՑ-ի հետազոտող խումբը զգալի տվյալներ է դուրս բերել, որոնք վերաբերում են ազգային աշխատաշուկային հատուկ ապրանքների և ոլորտի ընդհանուր կարքիներին: Հանդիպումների ժամանակ քննարկվեց

նաև, որ անհրաժեշտ է պետական կառույցների ուշադրությունը սևեռել շրջանավարտների որակի, բակալավրի և մագիստրոսի մակարդակների համար ընդհանրական պահանջների վրա:

Հետևելով սահմանված պլանին, հետազոտող խումբը մի շարք հանդիպումներ ունեցավ կազմակերպությունների գլխավոր տեխնիկական պատասխանատուների հետ (CTO) ՈՈԱԾ-ի խնդիրների վերաբերյալ: Քննարկումները հիմնված էին այն տվյալների վրա, որոնք դուրս էին բերվել կազմակերպությունների տնօրենների հետ նախնական զրույցների արդյունքնում:

Իրականացված ֆոկուս խմբերը մոդերացվում էին ՈԱԾ մշակման խմբի ղեկավարի կողմից, ով ունի ոլորտում դասավանդման, աշխատանքային փորձ և գիտելիք որակի ապահովման գործընթացների վերաբերյալ և կարող է պահպանել գործատուի հետ նույն լեզուն և ներկայացնել տեղեկատվությունը ճիշտ համատեքստում:

Ֆոկուս խմբերի միջին տևողությունը 1,5 ժամ էր: ՈԱԾ մոդելի գրաֆիկական պատկերը և նկարագրությունը ներկայացվեցին մասնակիցներին օգնելու գործատուներին առավել արագ ընկալել կառուցվածքի առանձնահատկությունները: Նշումները արվում էին մոդերատորի օգնականի կողմից: Ֆոկուս խումբը արձանագրվում էր հատուկ սարքով, որը հնարավորություն է տալիս իրականացնել վիդեո և աուդիո ձայնագրումներ: Այն լավ փորձի օրինակ է ծառայում գործատուների հետ հաղորդակցումը մշտապես ակտիվ պահելու և գործատուի մտահոգությունները հասկանալու համար: Ուսանողների համար վիդեո նյութերը հնարավարություն կտան հասկանալ ապագա աշխատանքային միջավայրը: Այս խոսակցությունները ուսանողների մոտ կապահովեն մասնագիտական շարժառիթ և հնարավորություն կտան գիտակցել ուսանողակենտրոն ուսուցման մոտեցումները: Նյութերը նաև կօգնեն առարկաների ընտրության վերաբերյալ որոշում կայացնելու ժամանակ:

ՈԱԾ-ի միջոցով կառուցված երկխոսությունը առավել արդյունավետ էր ՈՈԱԾ-ի մշակման համար, քանի որ այն հասկանալի էր

գործատուին, ի տարբերություն գործատուի հետ կրթական ծրագրի լեզվով հաղորդակցվելուն: Գործատուն ծանոթ է իր ապրանքի, ապրանքի շրջափուլի, իր աշխատակիցների ունակությունների, իրենց արժեքների/մտածողության հետ և շահագրգռված է առավել արդյունավետորեն ներկայացնելու իր աշխատանքին առնչվող հմտությունները: Այն նաև արդյունավետ է ստացված գիտելիքի մարմնի գնահատման և անհրաժեշտ նյութի ընտրության առումով: Այստեղ մասնակիցները հանդես են գալիս ինչպես որպես շրջանավարտ, այնպես էլ գործատու: ՈԱՇ-ը թույլ է տալիս ստեղծել միասնական լեզու գործատուների և բուհի միջև՝ սահմանելով համապատասխան պահանջներ:

Ֆոկուս խմբային քննարկումները ընդգծում են աշխատաշուկայում ակտիվ դերակատարների կողմից մատնանշված ուժեղ և թույլ կողմերը, անհրաժեշտ որակները և պրակտիկ աշխատանքի առաջին քայլերն անելու ժամանակ դուրս բերված բացերը և այն բացերը, որոնք ավելի կայուն են և դրանց բարելավումը պահանջում է հատուկ ջանքեր, քանի որ դրանք գալիս են կրթություն ստանալու առաջին փուլերից:

Այսպիսով, ծրագրի նախաձեռնությունը տալիս է ուսումնական պլանների շարունակական զարգացման և բարելավման մեխանիզմ, ինչպես նաև հնարավորություն երկրի ներսում դուրս բերել լավագույն փորձը՝ հաղորդակցվելով գործատուների, շրջանավարտների, ուսանողների և դասախոսների հետ՝ օգտագործելով ՈՈԱՇ-ի մեթոդաբանության մեջ առաջարկվող շրջանակը:

Գործատուների համար այս հետազոտությունը օգտակար է, քանի այն կողմնորոշիչ գործիք է հանդիսանում շուկայի կարիքների համար և ուղղորդում է, թե ինչպես հաղթահարել հնարավոր խոչընդոտները, երբ աշխատանքի են վերցնում ազգային բուհերի շրջանավարտների կամ նրանց ներառում են միջազգային խմբերում:

Մոդելի արդյունավետությունը հաստատեց նաև հարցազրույցների արդյունքում հավաքագրված տեղեկատվությունն ու տվյալները:

Ապրանք

Մասնակից գործատուները տվեցին առաջարկվող ապրանքների տարբեր դասակարգումներ (ապրանք/լուծում/ծառայություն, ծրագրեր/ծառայություններ, ցանց/ամպ/բջջային համակարգեր և այլն): Ապրանքի վերաբերյալ նրանք առանձնացրեցին առարկայական ոլորտի իմացության անհրաժեշտությունը: Կախված ապրանքի առանձնահատուկ պահանջներից (ճկուն, ձևափոխման ենթակա) աշխատակիցների հմտություններին առաջադրվող պահանջները տարբեր են:

Ապրանքի կենսափուլ

Գործատուները ընդգծել են ուսանողների շրջանում նախագծի կենսափուլի վերաբերյալ գիտելիքների կարևորությունը: Այս գիտելիքները նրանք պետք է փոխանցվեն կրթական ծրագրի սկզբում՝ ձևավորելու հատուկ մտածելակերպ/արժեհամակարգ: Խմբային առաջադրանքները կօգնեն յուրացնել ծրագրի կենսափուլը:

Մտածողություն

Քննարկումների առանցքը հանդիսանում էր մասնագիտական մտածողությունը: Գործատուները նշել են ալգորիթմական մտածողությունը, համակարգային մտածողությունը (համակարգը մասնատելու ունակություն), կոմբինատոր մտածողությունը որպես շրջանավարտների համար անհրաժեշտ: Համաձայն գործատուների արդյունքի թեստավորումը պարտադիր է մշակման բոլոր փուլերում, ըստ այդմ, ուսանողները պետք է հասկանան և արժևորեն այն: Թեստավորումը կարող է նաև հանդիսանալ ուսանողների գնահատման էական բաղկացուցիչ:

Ինքնուրույնություն

Գործատուները բակալավրի և մագիստրատուրայի կրթական ծրագրերի տարբերությունը տեսնում են նրանում, որ երկրորդը

պետք է ավելի կենտրոնանա նախագծերի վրա ձևավորելու ինքնուրույնություն ուսանողների շրջանում:

Ընդհանրական հմտություններ

Մասնակիցները նաև նշեցին շրջանավարտների ընդհանրական հմտությունների պակասը՝ հատկապես ընդգծելով հաղորդակցվելու, հետազոտելու, գրելու, ներկայացնելու և զեկուցելու հմտությունները: Բոլոր գործատուները շեշտեցին սովորել թե ինչպես սովորել կարողությունը որպես պահանջվող:

Թիմային աշխատանք

Գործատուները համաձայնության եկան աշխատավայրում թիմային աշխատանքի կարևորության վերաբերյալ, որը պետք է սերմանվի ուսանողների մոտ խմբային աշխատանքների միջոցով, ներառյալ ծրագրային համակարգերի ստուգում (software debugging), գործընկերային նախագծում/ծրագրավորում (peer development/programming), ծրագրի վերանայում (code review) և այլն: Նրանք նաև առաջարկեցին ներառել նման առաջադրանքներ կրթական ծրագրի մեջ և գնահատումն իրականացնել ուսանողի առաջընթացի հիման վրա:

Նախագծեր

Բոլոր գործատուները նշեցին, որ նախագծերը պետք է ներառվեն կրթական ծրագրում սկսած առաջին տարվանից: Այստեղ հիմնական նպատակը աշխատանքային համատեսքտի ներդրումն է ուսումնական գործընթացում՝ տեսական գիտելիքները կիրառական հմտությունների վերածելու և տարբեր առարկաները միմյանց կապելու նպատակով: Գործատուի հետ ուղղակի հաղորդակցությունը նշվել էր որպես մոտիվացիոն գործիք ուսանողների համար (ամառային դպրոցներ, պրակտիկա):

Գնահատում

Բացի աշխատաշուկայի պահանջներին համապատասխան գի-

տելիք տրմադրելու անհրաժեշտությունից, գործատուները նաև նշեցին, որ գնահատումը պետք է տեղի ունենա աշխատանքային պայմաններին մոտեցված, նույնիսկ իրականացվի գործատուի կողմից: Հետևաբար, պահանջվող գիտելիքի և հմտությունների մասին տեղեկատվությունը կարող է օգտագործվել գնահատման գործիքների մշակման գործընթացում:

Գիտելիքի մարմին

Պահանջվող գիտելիքի մարմնի վերաբերյալ գործատուները համաձայնեցին ֆունդամենտալ առարկաների (տվյալների բազաներ, ալգորիթմերի տեսություն, դիսկրետ մաթեմատիկա, տվյալների կառուցվածք, հավանականության տեսություն) և առնվազն մեկ ծրագրավորման լեզվի լիարժեք իմացության անհրաժեշտության շուրջ:

Միջդիսցիպլանարություն

Գործատուները նաև անդրադարձան տարբեր մասնագիտական կրթական ծրագրերի միջև կապի պակասին, որը, համաձայն քննարկումների մասնակիցների, նույնպես կարող է հաղթահարվել նախագծերի միջոցով: Տարբեր նպատակներին ծառայող նախագծերը սկսած ուսումնառության սկզբից կնպաստեն ուսանողների ինքնուրույնության, պատասխանատվության և կոնտեքստի բարդությանը ադապտացվելու ունակությունների զարգացմանը:

Աշխատանքային ստանդարտներ

Մասնակիցները քննարկեցին ուսանողների՝ աշխատանքային գործընթացի այլ ասպեկտների իմացության անհրաժեշտությունը, ինչպիսիք են կոդավորման ոճերը, նոր տեխնոլոգիաների կիրառումը, կորպորատիվ էթիկան և այլն:

Հարցազրույցների այս ձևաչափը արդյունավետ էր նաև դասախոսների համար: Այն համատեքստի վերաբերյալ տվյալներ ապահովեց, ինչը կարևոր հանգամանք է ուսանողների գնահատման

ժամանակ և օգնում է դասախոսներին պահել ապագա աշխատատեղի համատեքստի համապատասխանությունը:

Ինչպես նախկինում նշվեց, հետազոտող խումբը որոշեց հիմնական մեթոդաբանության մեջ պահել Ծրագրային համակարգերի ճարտարագիտության ոլորտի որակավորումների շրջանակի հիմնական նկարագրիչները միջազգային տեսանկյունից, միևնույն ժամանակ ազգային գործատուներին ներկայացրեց այդ նկարագրիչների մեկնաբանությունները: Դասախոսները, ովքեր պետք է ուսումնական պլաններում իրականացնեն նախատեսված ՈՈԱՇ-ը, պետք է օգտագործեն այդ մեկնաբանությունները՝ աշխատատեղի կոնտքեստը պրակտիկ գնահատման մեջ փոխակերպելու համար:

Դասախոսը կարող է փոխակերպել աշխատատեղի հանձնարարությունների մանրամասները գնահատման պահանջներում՝ առավել գործնական դարձնելու և ուսանողին գործատուի համատեքստին ծանոթացնելու համար:

Գիտելիքի մարմին, ազգային դաշտ: Ծրագրային ապահովման ճարտարագիտության ոլորտի որակավորումների շրջանակի մոդելի մշակման հաջորդ քայլը մոդելում ուսումնական պլանի էլեմենտների ներառումն է:

Հարցազրույցների ժամանակ շրջանավարտները շեշտադրել են այն հիմնական առարկաները, որոնք գործատուին կապահովեն անհրաժեշտ գիտելիքներ և հմտություններ: Բուհերն օգտագործում են ուսումնական պլաններ, որոնք գալիս են կիրառական մաթեմատիկայի կրթական չափորոշչից մշակված դեռևս սովետական ժամանակներում և վերանայված ՀՀ-ում 90-ներին և հիմնականում պահպանում են նմանությունները:

Միայն որոշ փոփոխություններ են եղել մի քանի կրթական ծրագրերում: ՀԱԱՅ-ի խումբը կարող է հաստատել, որ Ծրագրային համակարգերի ճարտարագիտության ուսումնական պլանը մինչ այսօր որոշ բաղադրիչներով համարվում է ամենազարգացածն ու

արդյունավետը: Հարգելով վերոնշյալը, հետազոտող խումբը նշում է նաև, որ Ծրագրային ապահովման ճարտարագիտություն ոլորտի պրակտիկ համատեքստը փոխվել է, հետևաբար արմատապես փոխվել է նաև գործատուի պրոֆիլը: Այժմ մոտեցումները այլ են: Գործատուները ընդունում են դասական առարկաները, օրինակ՝ գրաֆերի տեսություն (graph theory), տվյալների կառուցվածք (data structures), ալգորիթմների տեսություն (theory of algorithms), օպերացիոն համակարգեր (operating systems) և այլն: Սակայն միևնույն ժամանակ գործատուները մտահոգություն ունեն դասավանդման այն մոտեցումների վերաբերյալ, որոնցով ուսուցանվում են առարկաները: Համակարգչային համակարգերը շատ են փոխվել, իսկ նոր դաշտը չի համապատասխանում գոյություն ունեցող ուսումնական պլանին: Գործատուների ներգրավվածության անհրաժեշտությունը ըստ այդմ խիստ կարևոր է:

5. ՇԱՀԱԿԻՑՆԵՐԻ ՀԱՐՑՈՒՄ

ՈԱԾ-ի մշակման շրջանակներում իրականացվել են հարցումներ ոլորտի շահակիցների հետ ներառյալ գործատուներ, շրջանավարտներ, ուսանողներ և դասախոսներ պարզելու նրանց գնահատանակը մի շարք ընդհանրական և առարկայական հմտությունների և կարողությունների պահանջարկի վերաբերյալ: Համալսարան-աշխատաշուկա ազգային ցանցը իրականացրեց տվյալների հավաքագրման գործընթացը ծրագրի մասնակից բուհերի շրջանում:

Հարցաշարը կազմված է երկու բաժնից: Առաջինը ներառում է 18 ընդհանրական հմտություն և կարողություն: Երկրորդ բաժինը ներառում է առարկայական հմտություններ և կարողություններ խմբավորված ըստ ապրանքի յուրաքանչյուր կենսափուլի՝ մտահղացում, նախագծում, իրականացում, գործառնություն: Հարցվողները, երբ կիրառելի է, գնահատել են յուրաքանչյուր հմտություն 5 բալանոց սանդղակով (1-շատ քիչ, 2-քիչ, 3-որոշ չափով, 4-շատ,

5-առավելագույնս) ըստ հետևյալ չափանիշների.

- Անհրաժեշտություն
- Ձեռք է բերվել բուհում
- Ձեռք է բերվել աշխատելու ընթացքում
- Ես կարող եմ կատարել դա հետևյալ չափով:

Հարցումն իրականացվել է 4 մասնակից բուհերի SS մասնագիտությունների ուսանողների, շրջանավարտների պրոֆեսորադասախոսական կազմի շրջանում: Ներառված են եղել հետևյալ մասնագիտությունները յուրաքանչյուր բուհից:

1. Հայաստանի ազգային պոլիտեխնիկական համալսարան, Վանաձորի մասնաճյուղ
   - Ինֆորմատիկա և հաշվողական տեխնիկա (բակալավր)
2. Հայաստանի ազգային պոլիտեխնիկական համալսարան
   - Տեղեկատվական տեխնոլոգիաներ
   - Տեղեկատվական անվտանգություն
   - Ինֆորմատիկա և հաշվողական տեխնիկա
   - Համակարգչային գեղարվեստական նախագծում
   - Ինֆորմատիկա և կիրառական մաթեմատիկա
3. Գավառի պետական համալսարան
   - Ինֆորմատիկա և հաշվողական տեխնիկա (բակալավր)
4. Հայ-Ռուսական (Սլավոնական) համալսարան
   - Կիրառական մաթեմատիկա և ինֆորմատիկա (բակալավր)
   - Համակարգային ծրագրավորում (մագիստրատուրա)
   - Մաթեմատիկական մոդելավորում (մագիստրատուրա)

ՀԱԱՑ առցանց պլատֆորմի միջոցով իրականացված հարցումներին մասնակցել են 81 ուսանող, 45 շրջանավարտ, 28 դասախոս և 20 գործատու: Ստորև աղյուսակներում ներկայացնում են հարցումների արդյունքները:

5.1 ԸՆԴՀԱՆՐԱԿԱՆ ՀՄՏՈՒԹՅՈՒՆՆԵՐ ԵՎ ԿԱՐՈՂՈՒԹՅՈՒՆՆԵՐ

5.1.1 Գործատուներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Չեռք է բերվել բուհում	Չեռք է բերվել աշխատելու ընթացքում	Դրագրի բացը[1]
	Միջին գնահատականը			Միջին տարբերությունը
Փորձարկելու և նորարարական հմտություններ	4.9	3.0	3.8	1.9
Ստեղծագործ մտածելու ունակություն	4.7	3.0	3.9	1.7
Կոնֆլիկտի լուծում	4.6	2.7	3.4	1.9
Գաղափարները արդյունավետ փոխանցելու ունակություն	4.6	2.2	3.1	2.4
Բանավոր հաղորդակցություն	4.5	2.8	3.6	1.7
Բանակցային հմտություններ	4.5	2.4	3.3	2.1
Վերլուծելու և խնդիրներ լուծելու կարողություն	4.4	3.3	4.3	1.2
Տեսությունը գործնականում կիրառելու ունակություն	4.4	2.4	4.1	2.1
Սովորել, թե ինչպես սովորել	4.4	3.1	3.5	1.3

[1] “Անհրաժեշտություն” և “Չեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Ժամանակի և ռեսուրսների կառավարում	4.4	2.3	3.7	2.1
Մասնագիտական էթիկա	4.3	2.1	3.6	2.2
Թիմային աշխատանք և միջառարկայական թիմում աշխատելու ունակություն	4.3	2.9	4.0	1.4
Օտար լեզուների իմացություն	4.3	3.5	3.3	0.8
Գրավոր հաղորդակցություն	4.3	3.0	3.8	1.3
Ուրիշներին սովորեցնելու ունակություն	4.2	2.6	3.5	1.7
Ցանցեր ստեղծելու ունակություն	4.1	2.3	3.0	1.8
Առաջնորդում և ձեռնարկատիրական կարողություն	4.0	2.1	3.2	1.9

5.1.2 Դասախոսներ

Հմտություններ և կարողություններ	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Ծրագրի բացը[1]**
	Միջին գնահատականը		**Միջին տարբերությունը**
Տեսությունը գործնականում կիրառելու ունակություն	4.7	3.9	0.9
Մասնագիտական էթիկա	4.7	3.4	1.3
Վերլուծելու և խնդիրներ լուծելու կարողություն	4.7	3.6	1.0

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Բանավոր հաղորդակցություն	4.7	3.6	1.1
Գաղափարները արդյունավետ փոխանցելու ունակություն	4.6	3.3	1.3
Ստեղծագործ մտածելու ունակություն	4.6	3.6	1.0
Ժամանակի և ռեսուրսների կառավարում	4.5	3.1	1.4
Քննադատական մտածողություն	4.5	3.5	1.0
Կոնֆլիկտի լուծում	4.5	3.0	1.5
Թիմային աշխատանք և միջառարկայական թիմում աշխատելու ունակություն	4.5	3.2	1.3
Օտար լեզուների իմացություն	4.5	3.0	1.5
Փորձարկելու և նորարարական հմտություններ	4.5	3.5	1.0
Գրավոր հաղորդակցություն	4.4	3.7	0.7
Սովորել, թե ինչպես սովորել	4.4	3.6	0.8
Ցանցեր ստեղծելու ունակություն	4.4	2.7	1.7
Բանակցային հմտություններ	4.3	2.9	1.3
Ուրիշներին սովորեցնելու ունակություն	4.1	3.4	0.7
Առաջնորդում և ձեռնարկատիրական կարողություն	3.9	2.7	1.2

5.1.3 Շրջանավարտներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ձեռք է բերվել աշխատելու ընթացքում	Ես կարող եմ կատարել դա հետևյալ չափով	Ծրագրի բացը[1]	Կարողությունների բացը[2]
	Միջին գնահատականը				Միջին տարբերությունը	
Փորձարկելու և նորարարական հմտություններ	4.8	3.3	4.2	3.5	1.4	1.3
Բանավոր հաղորդակցություն	4.8	3.5	3.9	3.6	1.3	1.2
Կոնֆլիկտի լուծում	4.8	3.0	4.1	3.6	1.8	1.2
Գաղափարները արդյունավետ փոխանցելու ունակություն	4.6	3.6	4.4	3.7	1.0	0.9
Մասնագիտական էթիկա	4.5	3.6	4.3	3.8	0.9	0.7
Թիմային աշխատանք և միջառարկայական թիմում աշխատելու ունակություն	4.5	3.6	4.5	4.0	0.9	0.5

[1] "Անհրաժեշտություն" և "Ձեռք է բերվել բուհում" փոփոխականների միջին գնահատականների տարբերությունը:

[2] "Անհրաժեշտություն" և "Ես կարող եմ կատարել դա հետևյալ չափով" փոփոխականների միջին գնահատականների տարբերությունը:

Գրավոր հաղորդակցություն	4.5	2.9	4.5	3.7	1.6	0.8
Վերլուծելու և խնդիրներ լուծելու կարողություն	4.5	3.4	4.3	3.9	1.1	0.6
Տեսությունը գործնականում կիրառելու ունակություն	4.4	2.9	4.4	4.0	1.5	0.4
Ժամանակի և ռեսուրսների կառավարում	4.4	3.0	3.8	3.6	1.4	0.8
Օտար լեզուների իմացություն	4.4	3.4	3.9	3.6	0.9	0.8
Քննադատական մտածողություն	4.3	3.3	3.7	3.6	1.0	0.7
Առաջնորդում և ձեռնարկատիրական կարողություն	4.3	3.3	3.9	3.5	1.0	0.8
Սովորել, թե ինչպես սովորել	4.3	3.4	3.9	3.6	0.9	0.7
Ստեղծագործ մտածելու ունակություն	4.0	2.8	3.9	3.4	1.2	0.6
Բանակցային հմտություններ	4.0	3.3	3.8	3.4	0.8	0.6
Ցանցեր ստեղծելու ունակություն	3.8	3.3	3.8	3.0	0.5	0.8
Ուրիշներին սովորեցնելու ունակություն	3.8	3.6	3.9	3.6	0.1	0.2

5.1.4 Ուսանողներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ես կարող եմ կատարել դա հետևյալ չափով	Ծրագրի բացը[1]	Կարողությունների բացը[2]
	Միջին գնահատականը			Միջին տարբերությունը	
Վերլուծելու և խնդիրներ լուծելու կարողություն	4.5	3.4	3.5	1.1	1.0
Սովորել, թե ինչպես սովորել	4.4	3.2	3.6	1.2	0.7
Օտար լեզուների իմացություն	4.4	2.8	3.3	1.5	1.1
Թիմային աշխատանք և միջառարկայական թիմում աշխատելու ունակություն	4.4	3.3	3.8	1.0	0.6
Ցանցեր ստեղծելու ունակություն	4.3	2.7	3.0	1.6	1.4
Տեսությունը գործնականում կիրառելու ունակություն	4.3	3.3	3.8	1.0	0.5
Բանավոր հաղորդակցություն	4.3	3.3	3.8	1.0	0.5

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

[2] “Անհրաժեշտություն” և “Ես կարող եմ կատարել դա հետևյալ չափով” փոփոխականների միջին գնահատականների տարբերությունը:

Գաղափարները արդյունավետ փոխանցելու ունակություն	4.2	3.1	3.3	1.0	0.9
Բանակցային հմտություններ	4.2	3.2	3.6	1.0	0.6
Մասնագիտական էթիկա	4.2	3.3	3.4	0.9	0.7
Ժամանակի և ռեսուրսների կառավարում	4.2	2.9	3.3	1.2	0.8
Կոնֆլիկտի լուծում	4.1	3.1	3.6	1.0	0.5
Ստեղծագործ մտածելու ունակություն	4.1	2.9	3.8	1.2	0.3
Փորձարկելու և նորարարական հմտություններ	4.1	3.1	3.5	1.0	0.6
Գրավոր հաղորդակցություն	4.1	3.2	3.6	0.8	0.4
Քննադատական մտածողություն	3.9	3.2	3.4	0.7	0.5
Առաջնորդում և ձեռնարկատիրական կարողություն	3.8	2.8	3.1	1.0	0.7
Ուրիշներին սովորեցնելու ունակություն	3.8	3.0	3.6	0.7	0.2

5.1.5 Ընդհանրական հմտությունների և կարողությունների գնահատման արդյունքները ըստ բոլոր շահակիցների (միջին գնահատական)

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ծրագրի բացը[7]
	Միջին գնահատականը		Միջին տարբերությունը
Փորձարկելու և նորարարական հմտություններ	4.6	3.2	1.4
Բանավոր հաղորդակցություն	4.6	3.2	1.4
Վերլուծելու և խնդիրներ լուծելու կարողություն	4.5	3.4	1.1
Տեսությունը գործնականում կիրառելու ունակություն	4.5	2.9	1.6
Կոնֆլիկտի լուծում	4.5	2.8	1.7
Գաղափարները արդյունավետ փոխանցելու ունակություն	4.5	2.8	1.7
Սովորել, թե ինչպես սովորել	4.4	3.3	1.1
Ժամանակի և ռեսուրսների կառավարում	4.4	2.7	1.7
Մասնագիտական էթիկա	4.4	2.8	1.6
Թիմային աշխատանք և միջառարկայական թիմում աշխատելու ունակություն	4.4	3.2	1.3

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Օտար լեզուների իմացություն	4.4	3.4	1
Ստեղծագործ մտածելու ունակություն	4.3	3.1	1.2
Գրավոր հաղորդակցություն	4.3	3.1	1.2
Բանակցային հմտություններ	4.2	2.8	1.5
Ցանցեր ստեղծելու ունակություն	4.2	2.7	1.5
Քննադատական մտածողություն	4.1	3.3	0.8
Ուրիշներին սովորեցնելու ունակություն	4	3.1	0.9
Առաջնորդում և ձեռնարկատիրական կարողություն	4	2.6	1.4

5.2 ԱՌԱՐԿԱՅԱԿԱՆ ՀՄՏՈՒԹՅՈՒՆՆԵՐ ԵՎ ԿԱՐՈՂՈՒԹՅՈՒՆՆԵՐ

5.2.1 Գործատուներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ձեռք է բերվել աշխատելու ընթացքում	Ծրագրի բացը[1]
	Միջին գնահատականը			Միջին տարբերությունը
Մտահղացում	**4.5**	**2.3**	**3.9**	**2.2**
Շուկայի կարիքները հասկանալու ունակություն	4.6	2.8	4.3	1.8
Ծրագրի արժեքը հասկանալու և ներկայացնելու ունակություն	4.7	2.0	3.9	2.7
Ծրագրի իրականացման կոնտեքստը մեկնաբանելու ունակություն	4.1	2.2	3.6	2.0
Ինքնուրույն ծրագրեր նախաձեռնելու, պաշտպանելու ունակություն	4.4	2.0	3.9	2.4
Նախագծում	**4.1**	**2.0**	**4.1**	**2.2**
Ծրագրի իրականացման համար մեթոդաբանություն մշակելու ունակություն	4.1	2.0	4.3	2.1

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Մեթոդաբանության իրականացման նպատակով գործիքների մշակման ունակություն	4.4	1.9	4.3	2.6
Մեթոդաբանության արդյունավետությունը վերլուծելու ունակություն	4.1	2.2	3.9	2.0
Մեթոդաբանության մշակման ժամանակ առարկայական և միջառարկայական ասպեկտները մեկնաբանելու ունակություն	3.8	1.8	3.9	1.9
Մեթոդաբանության մշակման ժամանակ կանաչ տնտեսության տարրեր ինտեգրելու ունակություն (ռեսուրսախնայողություն, պատասխանատու վերաբերմունք գործատուի գույքին, շրջակա միջավայրին)	3.8	2.0	3.7	1.8
Ժամանակակից տեխնոլոգիաները մեթոդաբանության մշակման գործընթացում կիրառելու ունակություն	4.1	1.8	3.9	2.3
Իրականացում	**4.7**	**2.6**	**4.4**	**2.1**
Ինքնուրույն որոշումներ ընդունելու ունակություն	4.7	2.3	4.6	2.4
Իրականացման գործընթացը արդյունավետ պլանավորելու ունակություն	4.9	2.4	4.3	2.5
Ըստ պլանների աշխատելու ունակություն	4.6	2.8	4.5	1.8
Ռեսուրսների արդյունավետ կառավարման ունակություն	4.7	2.4	4.3	2.3

Մշակված մեթոդաբանությունը գործնականում կիրառելու ունակություն	4.6	3.0	4.4	1.6
Պլանների արդյունավետ իրականացումն ապահովելու ունակություն	4.6	2.7	4.5	1.9
Գործառնություն	**4.5**	**2.3**	**3.9**	**2.2**
Ընթացիկ գործընթացների արդյունավետությունը գնահատելու ունակություն	4.7	2.2	3.7	2.5
Ոչ համապատասխան գործընթացները դադարեցնելու ունակություն	4.6	2.2	3.6	2.4
Համապատասխանող գործընթացները կատարելագործելու ունակություն	4.4	2.3	3.6	2.1
Գնահատման/մոնիտորինգի գործընթացների կառավարման ունակություն	4.6	2.3	3.4	2.2
Գործընկերներին աջակցելու ունակություն	4.7	2.7	4.5	2.0
Արդյունքները հանրությանը ներկայացնելու ունակություն (առաջխաղացում)	4.3	2.3	4.3	2.0

5.2.2 Դասախոսներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ծրագրի բացը[1]
	Միջին գնահատականը		Միջին տարբերությունը
Մտահղացում	**4.4**	**3.1**	**1.3**
Շուկայի կարիքները հասկանալու ունակություն	4.4	3.0	1.4
Ծրագրի արժեքը հասկանալու և ներկայացնելու ունակություն	4.5	3.2	1.3
Ծրագրի իրականացման կոնտեքստը մեկնաբանելու ունակություն	4.3	3.0	1.3
Ինքնուրույն ծրագրեր նախաձեռնելու, պաշտպանելու ունակություն	4.4	3.1	1.3
Նախագծում	**4.1**	**3.1**	**1.0**
Ծրագրի իրականացման համար մեթոդաբանություն մշակելու ունակություն	4.0	3.2	0.8

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Մեթոդաբանության իրականացման նպատակով գործիքների մշակման ունակություն	4.1	3.1	0.9
Մեթոդաբանության արդյունավետությունը վերլուծելու ունակություն	4.1	2.9	1.1
Մեթոդաբանության մշակման ժամանակ առարկայական և միջառարկայական ասպեկտները մեկնաբանելու ունակություն	4.0	2.9	1.1
Մեթոդաբանության մշակման ժամանակ կանաչ տնտեսության տարրեր ինտեգրելու ունակություն (ռեսուրսախնայողություն, պատասխանատու վերաբերմունք գործատուի գույքին շրջակա միջավայրին)	3.9	3.0	0.9
Ժամանակակից տեխնոլոգիաները մեթոդաբանության մշակման գործընթացում կիրառելու ունակություն	4.3	3.1	1.1
Իրականացում	**4.4**	**3.4**	**1.0**
Ինքնուրույն որոշումներ ընդունելու ունակություն	4.6	3.4	1.2
Իրականացման գործընթացը արդյունավետ պլանավորելու ունակություն	4.4	3.5	0.9
Ըստ պլանների աշխատելու ունակություն	4.5	3.6	0.8
Ռեսուրսների արդյունավետ կառավարման ունակություն	4.4	3.3	1.1

Մշակված մեթոդաբանությունը գործնականում կիրառելու ունակություն	4.1	3.1	1.0
Պլանների արդյունավետ իրականացումն ապահովելու ունակություն	4.2	3.4	0.8
Գործառնություն	**4.3**	**3.3**	**1.1**
Ընթացիկ գործընթացների արդյունավետությունը գնահատելու ունակություն	4.5	3.1	1.4
Ոչ համապատասխան գործընթացները դադարեցնելու ունակություն	4.1	3.0	1.1
Համապատասխանող գործընթացները կատարելագործելու ունակություն	4.3	3.4	0.8
Գնահատման/մոնիտորինգի գործընթացների կառավարման ունակություն	4.2	3.2	1.0
Գործընկերներին աջակցելու ունակություն	4.3	3.4	1.0
Արդյունքները հանրությանը ներկայացնելու ունակություն (առաջխաղացում)	4.4	3.4	1.0

5.2.3 Շրջանավարտներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ձեռք է բերվել աշխատանքի ընթացքում	Ես կարող եմ կատարել դա հետևյալ հիմա՞	Ծրագրի բացը[1]	Կարողությունների բացը[2]
	Միջին գնահատականը				**Միջին տարբերությունը**	
Մտահղացում	**4.4**	**3.4**	**4.3**	**4.2**	**0.9**	**0.2**
Շուկայի կարիքները հասկանալու ունակություն	4.4	3.0	4.2	4.0	1.4	0.4
Ծրագրի արժեքը հասկանալու և ներկայացնելու ունակություն	4.4	3.6	4.6	4.3	0.9	0.1
Ծրագրի իրականացման կոնտեքստը մեկնաբանելու ունակություն	4.3	3.5	4.0	4.2	0.8	0.1

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

[2] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Ինքնուրույն ծրագրեր նախաձեռնելու, պաշտպանելու ունակություն	4.3	3.7	4.6	4.2	0.6	0.1
Նախագծում	**4.1**	**3.3**	**3.6**	**3.4**	**0.8**	**0.6**
Ծրագրի իրականացման համար մեթոդաբանություն մշակելու ունակություն	4.0	3.0	3.5	3.7	1.0	0.3
Մեթոդաբանության իրականացման նպատակով գործիքների մշակման ունակություն	4.1	3.3	3.8	3.4	0.9	0.7
Մեթոդաբանության արդյունավետությունը վերլուծելու ունակություն	4.0	3.7	3.7	3.4	0.3	0.6
Մեթոդաբանության մշակման ժամանակ առարկայական և միջառարկայական ասպեկտները մեկնաբանելու ունակություն	4.1	3.3	3.2	3.3	0.9	0.9
Մեթոդաբանության մշակման ժամանակ կանաչ տնտեսության տարրեր ինտեգրելու ունակություն (ռեսուրսախնայողություն, պատասխանատու վերաբերմունք գործատուի գույքին, ինչպես նաև շրջակա միջավայրին)	4.0	3.8	4.0	3.8	0.2	0.3
Ժամանակակից տեխնոլոգիաները մեթոդաբանության մշակման գործընթացում կիրառելու ունակություն	4.0	3.2	3.6	3.6	0.8	0.4
Իրականացում	**4.6**	**3.8**	**4.4**	**4.4**	**0.8**	**0.2**
Ինքնուրույն որոշումներ ընդունելու ունակություն	4.7	3.7	4.4	4.5	1.0	0.2

Իրականացման գործընթացը արդյունավետ պլանավորելու ունակություն	4.4	3.5	4.6	4.5	0.9	-0.1
Ըստ պլանների աշխատելու ունակություն	4.6	3.9	4.5	4.2	0.7	0.4
Ռեսուրսների արդյունավետ կառավարման ունակություն	4.7	4.0	4.4	4.3	0.7	0.4
Մշակված մեթոդաբանությունը գործնականում կիրառելու ունակություն	4.7	3.7	4.1	4.3	1.0	0.4
Պլանների արդյունավետ իրականացումն ապահովելու ունակություն	4.5	3.8	4.5	4.4	0.7	0.1
Գործառնություն	**4.5**	**3.5**	**4.0**	**4.0**	**1.0**	**0.5**
Ընթացիկ գործընթացների արդյունավետությունը գնահատելու ունակություն	4.5	3.6	4.0	4.0	0.9	0.5
Ոչ համապատասխան գործընթացները դադարեցնելու ունակություն	4.6	3.3	4.1	4.2	1.3	0.4
Համապատասխանող գործընթացները կատարելագործելու ունակություն	4.6	3.6	4.3	4.2	1.0	0.4
Գնահատման/մոնիտորինգի գործընթացների կառավարման ունակություն	4.4	3.4	3.6	3.7	1.0	0.8
Գործընկերներին աջակցելու ունակություն	4.6	3.7	4.3	4.2	0.9	0.4
Արդյունքները հանրությանը ներկայացնելու ունակություն (առաջխաղացում)	4.5	3.6	4.0	3.8	0.9	0.7

5.2.4 Ուսանողներ

Հմտություններ և կարողություններ	Անհրաժեշտություն	Ձեռք է բերվել բուհում	Ես կարող եմ կատարել դա հետևյալ չափով	Ծրագրի բացը[1]	Կարողությունների բացը[2]
	Միջին գնահատականը			Միջին տարբերությունը	
Մտահղացում	**4.1**	**3.2**	**3.4**	**0.9**	**0.7**
Շուկայի կարիքները հասկանալու ունակություն	4	3	3.4	1	0.6
Ծրագրի արժեքը հասկանալու և ներկայացնելու ունակություն	4.2	3.2	3.4	1	0.8
Ծրագրի իրականացման կոնտեքստը մեկնաբանելու ունակություն	4	3.3	3.4	0.8	0.6
Ինքնուրույն ծրագրեր նախաձեռնելու, պաշտպանելու ունակություն	4.2	3.2	3.6	1	0.6

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

[2] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Նախագծում	**4.1**	**3.3**	**3.5**	**0.8**	**0.6**
Ծրագրի իրականացման համար մեթոդաբանություն մշակելու ունակություն	4.2	3.4	3.5	0.8	0.7
Մեթոդաբանության իրականացման նպատակով գործիքների մշակման ունակություն	3.9	3	3.3	0.9	0.6
Մեթոդաբանության արդյունավետությունը վերլուծելու ունակություն	4.2	3.3	3.5	0.9	0.7
Մեթոդաբանության մշակման ժամանակ առարկայական և միջառարկայական ասպեկտները մեկնաբանելու ունակություն	4.1	3.5	3.6	0.6	0.5
Մեթոդաբանության մշակման ժամանակ կանաչ տնտեսության տարրեր ինտեգրելու ունակություն (ռեսուրսախնայողություն, պատասխանատու վերաբերմունք գործատուի գույքին, ինչպես նաև շրջակա միջավայրին)	4.1	3.3	3.5	0.7	0.6
Ժամանակակից տեխնոլոգիաները մեթոդաբանության մշակման գործընթացում կիրառելու ունակություն	4.3	3.3	3.4	1	0.9
Իրականացում	**4.3**	**3.4**	**3.8**	**0.8**	**0.5**
Ինքնուրույն որոշումներ ընդունելու ունակություն	4.3	3.4	4	0.9	0.3

Իրականացման գործընթացը արդյունավետ պլանավորելու ունակություն	4.4	3.6	3.8	0.8	0.6
Ըստ պլանների աշխատելու ունակություն	4.2	3.6	3.9	0.7	0.3
Ռեսուրսների արդյունավետ կառավարման ունակություն	4.3	3.7	3.8	0.6	0.5
Մշակված մեթոդաբանությունը գործնականում կիրառելու ունակություն	4.1	3.1	3.5	1	0.6
Պլանների արդյունավետ իրականացումն ապահովելու ունակություն	4.3	3.3	3.6	0.9	0.7
Գործառնություն	**4.3**	**3.5**	**3.6**	**0.8**	**0.7**
Ընթացիկ գործընթացների արդյունավետությունը գնահատելու ունակություն	4.2	3.5	3.6	0.7	0.6
Ոչ համապատասխան գործընթացները դադարեցնելու ունակություն	4.2	3.3	3.4	0.9	0.8
Համապատասխանող գործընթացները կատարելագործելու ունակություն	4.3	3.7	3.6	0.7	0.7
Գնահատման/մոնիտորինգի գործընթացների կառավարման ունակություն	4.2	3.4	3.7	0.8	0.5

Գործընկերներին աջակցելու ունակություն	4.3	3.6	3.7	0.7	0.6
Արդյունքները հանրությանը ներկայացնելու ունակություն (առաջխաղացում)	4.4	3.7	3.9	0.7	0.5

5.2.5 Առարկայական հմտությունների և կարողությունների գնահատման արդյունքները ըստ բոլոր շահակիցների (միջին գնահատական)

	Միջին գնահատականը		
Հմտություններ և կարողություններ	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Ծրագրի բացը[1]**
Մտահղացում	**4.3**	**3.0**	**1.4**
Շուկայի կարիքները հասկանալու ունակություն	4.4	3.0	1.4
Ծրագրի արժեքը հասկանալու և ներկայացնելու ունակություն	4.5	3.0	1.5
Ծրագրի իրականացման կոնտեքստը մեկնաբանելու ունակություն	4.2	3.0	1.2
Ինքնուրույն ծրագրեր նախաձեռնելու, պաշտպանելու ունակություն	4.3	3.0	1.3

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

Նախագծում	**4.1**	**2.9**	**1.2**
Ծրագրի իրականացման համար մեթոդաբանություն մշակելու ունակություն	4.1	2.9	1.2
Մեթոդաբանության իրականացման նպատակով գործիքների մշակման ունակություն	4.1	2.8	1.3
Մեթոդաբանության արդյունավետությունը վերլուծելու ունակություն	4.1	3.0	1.1
Մեթոդաբանության մշակման ժամանակ առարկայական և միջառարկայական ասպեկտները մեկնաբանելու ունակություն	4.0	2.9	1.1
Մեթոդաբանության մշակման ժամանակ կանաչ տնտեսության տարրեր ինտեգրելու ունակություն (ռեսուրսախնայողություն, պատասխանատու վերաբերմունք գործատուի գույքին, շրջակա միջավայրին)	3.9	3.0	0.9
Ժամանակակից տեխնոլոգիաները մեթոդաբանության մշակման գործընթացում կիրառելու ունակություն	4.2	2.9	1.3
Իրականացում	**4.5**	**3.3**	**1.2**
Ինքնուրույն որոշումներ ընդունելու ունակություն	4.6	3.2	1.4

Իրականացման գործընթացը արդյունավետ պլանավորելու ունակություն	4.5	3.2	1.3
Ըստ պլանների աշխատելու ունակություն	4.5	3.5	1.0
Ռեսուրսների արդյունավետ կառավարման ունակություն	4.5	3.4	1.2
Մշակված մեթոդաբանությունը գործնականում կիրառելու ունակություն	4.4	3.2	1.1
Պլանների արդյունավետ իրականացումն ապահովելու ունակություն	4.4	3.3	1.1
Գործառնություն	**4.4**	**3.2**	**1.3**
Ընթացիկ գործընթացների արդյունավետությունը գնահատելու ունակություն	4.5	3.1	1.4
Ոչ համապատասխան գործընթացները դադարեցնելու ունակություն	4.4	2.9	1.4
Համապատասխանող գործընթացները կատարելագործելու ունակություն	4.4	3.3	1.2
Գնահատման/մոնիտորինգի գործընթացների կառավարման ունակություն	4.4	3.1	1.3
Գործընկերներին աջակցելու ունակություն	4.5	3.3	1.1
Արդյունքները հանրությանը ներկայացնելու ունակություն (առաջխաղացում)	4.4	3.3	1.2

5.2.6 Ապրանքի յուրաքանչյուր շրջափուլի առարկայական հմտությունների և կարողությունների գնահատման արդյունքները ըստ բոլոր շահակիցների (միջին գնահատական)

	Ուսանողներ		Շրջանավարտներ		Գործատուներ		Դասախոսներ		Միջինը		
Հմտություններ և կարողություններ	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Անհրաժեշտություն**	**Ձեռք է բերվել բուհում**	**Ըրագրի բացը[1]**
Մտահղացում	4.5	2.3	4.1	3.2	4.4	3.4	4.4	3.1	4.3	3.0	1.4
Նախագծում	4.1	2.0	4.1	3.3	4.1	3.3	4.1	3.1	4.1	2.9	1.2
Իրականացում	4.7	2.6	4.3	3.4	4.6	3.8	4.4	3.4	4.5	3.3	1.2
Գործառնություն	4.5	2.3	4.3	3.5	4.5	3.5	4.3	3.3	4.4	3.2	1.3

1.

[1] “Անհրաժեշտություն” և “Ձեռք է բերվել բուհում” փոփոխականների միջին գնահատականների տարբերությունը:

6. ԾՐԱԳՐԱՅԻՆ ՀԱՄԱԿԱՐԳԵՐԻ ՃԱՐՏԱՐԱԳԻՏՈՒԹՅՈՒՆ ՈՒԼՈՐՏԻ ՈՐԱԿԱՎՈՐՈՒՄՆԵՐԻ ԱՋԳԱՅԻՆ ՇՐՋԱՆԱԿ

<table>
<tr><th colspan="3">Ծրագրային համակարգերի ճարտարագիտության ապրանքը</th></tr>
<tr>
<td>Ընդհանրական պահանջներ
• Մինիմալ սխալներ
• Օգտվողի մակսիմալ բավարարվածություն
• Խնդիրներին արձագանքման նվազագույն ժամանակ
• Լավ սպասարկելիություն
• Լավ ընդարձակելիություն
• Բարձր հուսալիություն
• Բարձր ճշտություն</td>
<td>Ծրագրի նպատակը
• Համակարգային ծրագրեր
• Իրական ժամանակի ծրագրեր
• Գործարարության ծրագրեր
• Ինժեներական և գիտական ծրագեր
• Արհեստական ինտելեկտի ծրագեր
• Վեբ ծրագրեր
• Անհատական համակարգչի (PC) ծրագեր</td>
<td>Օգտագործման միջավայրը
• Ներդրված համակարգեր
o Առանձնացված ներդրված համակարգեր
o Իրական ժամանակի ներդրված համակարգեր
o Ցանցային ներդրված համակարգեր
o Շարժական ներդրված համակարգեր
o Ցանցային և ամպային/ համացանցային համակարգեր
Ծրագրավորման մեթոդը
• Բաց աղբյուրով ծրագրեր</td>
</tr>
<tr>
<td colspan="3">Գործատուի թվարկած հիմնական ապրանքների ցանկ
• Օգտվողի կարիքներին հարմարեցված ծրագրավորում և ծրագրավորման արտահանում
• Չիպերի նախագծում, թեստավորում և այլ հարակից գործածություններ
• Ինտերնետ ծառայություններ
• Ցանցերի համակարգեր և կապ</td>
</tr>
</table>

- Ինտերնետ ծրագրեր և էլեկտրոնային/առցանց առևտուր
- SS ծառայություններ և խորհրդատվություն Հաշվապահական, բանկային և ֆինանսական ծրագրեր
- Վեբ դիզայն և մշակում
- Համակարգչային գրաֆիկա, մուլտիմեդիա և խաղեր
- Տվյալների բազաներ և Կառավարման տեղեկատվության համակարգեր (MIS)
- Այլ

Մասնագիտական մտածողություն

Մասնագիտական մտածողությունը խնդիրների լուծման գործընթաց է, որ ներառում է (սակայն սահմանափակված չէ) հետևյալ հատկանիշները.

- Խնդիրների ձևակերպում այնպես, որ լուծելու համար հնարավոր լինի օգտագործել համակարգիչ և այլ գործիքներ
- Տվյալների տրամաբանորեն կազմակերպում և վերլուծություն նրանց արժեքների և ծավալների փոփոխության մեծ հաճախության պայմաններում
- Տվյալների ներկայացում վերացարկված ձևով, ինչպիսիք են մոդելները և նմանարկումները, օգտագործելով բազմամակարդակ վերացարկում
- Մաթեմատիկայի կիրառում ալգորիթմների մշակման համար և ստուգելու, թե որքանով ճիշտ է լուծումը տարածվում խնդիրների տարբեր չափերի նկատմամբ
- Լուծումների ավտոմատացում ալգորիթմական մտածելակերպի միջոցով (քայլերի հաջորդական շղթա, հնարավոր է, զուգահեռ պատահարների միջավայրում)
- Հնարավոր լուծումների դուրսբերում, վերլուծություն և կիրառում քայլերի և ռեսուրսների առավել արդյունավետ համադրություն ստանալու նպատակով, ընդհանրացնելով և փոխանցելով խնդրի լուծման գործընթացը ավելի լայն խնդիրների դասերի համար:

Գործատուների արձագանքը		
• Թիմային մտածողություն • Կիրառության ոլորտին համապատասխան պատասխանատվության գիտակցում • Ալգորիթմական մտածողություն • Կիրառման հնարավոր ելքերի կանխատեսում • Կիրառությունից բխող անվտանգության արժևորում • Համակարգային մտածողություն • Նորը սովորելու արժևորում • Վերացական մտածողություն • Խնդիր լուծելու կարողություն		
Ապրանքի կենսափուլ		
ՄՏԱՀՂԱՑՈՒՄ		
Նպատակների և պահանջների սահմանում		
Կոմպետենցիա	ԲԱ: Ներգրավված է ծրագրային ապրանքների կամ ծառայությունների/արդյունքի անհրաժեշտության բացահայտման/դուրսբերման և նպատակների սահմանման գործընթացում և կարող է պատասխանատվություն վերցնել տեխնիկական առաջադրանքների կամ նման առաջադրանքներ լուծող խմբի աշխատանքների համար:	ՄԱ: Սահմանված լիազորությունների շրջանակում իրականացնում է ծրագրային ապրանքների և ծառայությունների անհրաժեշտության բացահայտում և նպատակների սահմանում ներառյալ տեխնիկական, ֆինանսական և որակի ասպեկտները: Պատասխանատվություն է կրում դրանում ներգրավված խմբի աշխատանքների համար:

Հմտություն	• Դուրս է բերում շուկայի կարիքները և հնարավորությունները • Բացահայտում և մեկնաբանում է պատվիրատուի կարիքները • Դուրս է բերում թաքնված կարիքներից կամ նոր տեխնոլոգիաներից բխող հնարավորությունները • Բացատրում է պահանջների կոնտեքստը պայմանավորող գործոնները • Դուրս է բերում ձեռնարկության նպատակները, ռազմավարությունները, կարողությունները և կապերը • Մեկնաբանում է պահանջների ամբողջականությունը և հետևողականությունը	• Հայտնաբերում է և դասակարգում է մրցակիցներին և համեմատական տեղեկատվությունը • Մեկնաբանում է էթիկական, սոցիալական, բնապահպանական, իրավական և կարգավորող գործոնները • Բացատրում է համակարգի վրա ազդող գործոնների փոփոխության հավանականությունը, նպատակները և առկա ռեսուրսները • Մեկնաբանում է համակարգի նպատակները և պահանջները • Բացահայտում է նպատակների և պահանջների լեզուն/ֆորմատը • Մեկնաբանում է նախնական նպատակները (հիմնված կարիքների, հնարավորությունների և այլ ազդեցությունների վրա) • Բացատրում է համակարգի կատարողականի ցուցանիշները
Ֆունկցիաների, սկզբունքների և ճարտարապետության սահմանում		
Կոմպետենցիա	ԲԱ: Ներգրավված է ծրագրային ապրանքների կամ ծառայության գործառույթների սահմանման, ընդհանուր գաղափարի մշակման և ճարտարապետական դիզայն իրականացնող խմբերում և պատասխանատու է տեխնիկական առաջադրանքների իրականացման համար:	ՄԱ: Պատասխանատու է ծրագրային ապրանքների կամ ծառայության գործառույթների սահմանման, ընդհանուր գաղափարի մշակման և բարձր մակարդակի/ճարտարապետական դիզայնի համար: Սահմանում է դրա իրականացման համար անհրաժեշտ քայլերը:

Հմտություն	• Պարզաբանում է համակարգի անհրաժեշտ ֆունկցիաները և համակարգի վարքի նկարագիրը • Ընտրում է համակարգի կառուցվածքային սկզբունքը/մոտեցումը • Պարզաբանում է տեխնոլոգիական հնարավորությունները • Վերլուծում է հնարավոր կառուցվածքային սկզբունքների ընտրությունը	• Պարզաբանում է բարձր մակարդակի ճարտարապետական ձևը և կառուցվածքը • Քննարկում է ձևի մասնատումը էլեմենտների, նշում էլեմենտների ֆունկցիաները և սահմանում է կապերը
Համակարգի մոդելավորում և նպատակների իրականացման ապահովում		
Կոմպետենցիա	ԲԱ: Մասնակցում է նպատակներից բխող ապրանքի մոդելավորման աշխատանքներին և պատասխանատու է նախագծման (մասնագիտական ոլորտի խնդիրներ լուծելով) աշխատանքների համար:	ՄԱ: Պատասխանատու է այնպիսի ապրանքի մոդելի ստեղծման համար, որը կապահովի նպատակների ձեռքբերումը/իրականացումը: Աշխատանքները իրականացնում է թիմի հետ միասին և պատասխանատու է այդ թիմի աշխատանքների համար:
Հմտություն	• Որոշում է համապատասխան տեխնիկական մոդելների կատարումը/ներկայացումը • Քննարկում է ամբողջ շրջափուլի գործի ծավալը և արժեքը (նախագծում, իրականացում, գործարկում, հնարավորություններ և այլն)	• Քննարկում է տարբեր նպատակների, ֆունկցիաների մոտեցումների ու կառուցվածքների հնարավոր ընտրությունների ինչպես նաև մինչև համաձայնության գալը կատարման հնարավոր իտերացիաների միջև

Նախագծի կառավարման մշակում		
Կոմպետենցիա	ԲԱ: Ներգրավված է ինժեներական ապրանքի ստեղծման նախագծի մշակման գործընթացում և լուծում է տեխնիկական առաջադրանքներ կապված արդյունավետության ցուցիչների, անհրաժեշտ փաստաթղթերի և ռեսուրսների հետ:	ՄԱ: Պատասխանատու է ինժեներական ապրանքի ստեղծման նախագծի/գործառույթների մշակման համար՝ ներառյալ դրա ժամկետների, արդյունավետության ցուցիչների, անհրաժեշտ փաստաթղթերի և ռեսուրսների սահմանումը:
Հմտություն	• Նկարագրում է նախագծի գնի, կատարման և ժամանակացույցի վերահսկումը • Բացատրում է համապատասխան անցումային և վերանայումների կետերը • Բացատրում է կոնֆիգուրացիայի կառավառումը և փաստաթղթավորումը	• Մեկնաբանում է կատարողականը մինիմալ բազայինի նկատմամբ • Քննարկում է ռեսուրսների գնահատումը և տեղաբաշխումը • Բացահայտում է ռիսկերը և այլընտրանքները • Նկարագրում է մշակման գործընթացի հնարավոր բարելավումները
Գործատուների արձագանքը		
	• Ծանոթ լինի կիրառության ոլորտին • Միայն կոդ չպետք է գրի, պետք է մտածի դրա արդյունքի կիրառության մասին • Պատվիրատուի հետ շփման ընթացքում մտածի նրա կարիքների հայտնաբերման մասին • Ունենա շուկայի ուսումնասիրման հմտություններ • Համեմատի ապրանքի գործառույթները պատվերի/նպատակի հետ • Ծանոթ լինի շուկայի արդի պահաջարկին • Մտածի գլոբալ շուկայի մակարդակով • Բացահայտի պատվիրատուի կարիքները և լուծումներ առաջարկի • Տեղափոխի գիտելիքը գործնական միջավայր	

ՆԱԽԱԳԾՈՒՄ		
The design process		
Կոմպետենցիա	ԲԱ: Մասնակցում է ծրագրի նախագծի մշակման աշխատանքներին՝ իրականացնելով մասնագիտական ոլորտին վերաբերող վերլուծություններ (անորոշ մասնագիտական միջավայր):	ՄԱ: Պատասխանատու է ծրագրի նախագծի մշակման համար ներառյալ դրա այլընտանքային տարբերակների վերլուծությունը (սա անորոշ միջավայրն է) և մշակված դիզայնի փորձարկման մեխանիզմների մշակումը (սա անորոշ արդյունքն է):
Հմտություն	• Ամեն էլէմենտի կամ կոմպոնենտի համար ընտրում է պահանջները, որոնք բխում են համակարգային մակարդակի նպատակներից և պահանջներից • Վերլուծում է նախագծման այլընտրանքները • Ընտրում է նախանական նախագիծը • Օգտագործում է նախատիպներ և թեստային օրինակներ նախագծի մշակման ընթացքում	• Կատարում է համապատասխան օպտիլմալացումներ սահմանափակումների առկայության դեպքում • Ցուցադրում է հնարավոր իտերացիաները մինչև համաձայնության գալը • Սինթեզում է վերջնական դիզայնը • Ցուցադրում է հնարավոր փոփոխությունները
Նախագծման գործընթացի փուլերի բաժանում և մոտեցումներ		

Կոմպետենցիա	ԲԱ: Մասնակցում է ծրագրի նախագծի փուլերի և մոտեցումների մշակման աշխատանքներին՝ իրականացնելով մասնագիտական ոլորտին բնորոշ գործառույթներ:	ՄԱ: Պատասխանատու է ծրագրի նախագծման փուլերի և մոտեցումների մշակման համար իրականացնելով ինչպես մասնագիտական, այնպես էլ ոչ մասնագիտական ոլորտին առնչվող վերլուծություններ:
Հմտություն	• Բացատրում է համակարգի նախագծման փուլի գործառույթները (օրինակ՝ սկզբունքային, նախնական և մանրամասն նախագիծ) • Քննարկում է կոնկրետ/տվյալ նախագծի մշակման գործընթացի համապատասխան մոդելը (ջրվեժ, պտուտակի, և այլն)	• Քննարկում է գործառույթը միակ (նպատակային), հարթակ (այլ բանի հիմք ծառայող) և աճանցյալ (անուղղակի արդյունք հանդիսացող) տեսակի արդյունքների համար
Դիսցիպլինար նախագծում		
Կոմպետենցիա	ԲԱ: Մասնակցում է ծրագրի նախագծի մշակման համար անհրաժեշտ գործիքների ընտրության, հիմնավորման և դրա որակական ցուցիչների վերլուծությանը:	ՄԱ: Պատասխանատվություն է կրում ծրագրի նախագծի մշակման համար անհրաժեշտ գործիքների ընտրության, հիմնավորման և դրա որակական ցուցիչների վերլուծության համար: Աշխատանքը կարող է կատարել թիմը ղեկավարելու միջոցով:

Հմտություն	• Ընտրում է համապատասխան տեխնիկական ձև/միջոց, գործիք և գործառույթ • Բացատրում է նախագծման գործիքի ընտրության հարմարեցում և հիմնավորում	• Գործարկում է այլընտրանքների քանակական վերլուծությունը • Գործարկում է մոդելավորում, նմանարկում և թեստավորում/փորձարկում • Քննարկում է նախագծի անալիտիկ ճշգրտումը
Միջոլորտային նախագծում		
Կոմպետենցիա	ԲԱ: Մասնակցում է ծրագրի նախագծի մշակման ընթացքում առաջացող միջառարկայական խնդիրների լուծման գործընթացին՝ պատասխանատվություն վերցնելով մասնագիտական ոլորտին վերաբերող առաջադրանքների լուծման համար:	ՄԱ: Պատասխանատու է ծրագրի նախագծի մշակման ընթացքում առաջացող միջառարկայական խնդիրների լուծման համար:
Հմտություն	• Բացահայտում է միջոլորտային կապերը • Բացահայտում է իրարամերժ/աննախադեպ կոնվենցիաները և ենթադրությունները/նախապայմանները • Բացատրում է կարգապահական (նորմերի) մոդելների ժամկետայնության տարբերությունները	• Բացատրում է բազմաոլորտ նախագծման միջավայրերը • Բացատրում է բազմաոլորտ նախագիծը

<table>
<tr><td colspan="3">Գործատուների արձագանքը</td></tr>
<tr><td></td><td colspan="2">• Համեմատել և ընտրել ժամանակի, գնի և որակի նպատակահարմար համամասնությունը
• Գնահատել նախագծի կառուցվածքի դրական և բացասական կողմերը
• Սահմանել նախագծի հիմնական հատկությունների օպտիմալ համամասնությունը
• Կիրառել պատրաստի և ստուգված բաղադրիչներ
• Նախագծել թեստավորման հնարավորությունները
• Հաշվի առնել պահանջների հնարավոր փոփոխությունները
• Փոփոխել և մեկնաբանել անծանոթ ծրագիրը
• Նախագծել համակարգի տեղափոխելու հնարավորությունը այլ միջավայր</td></tr>
<tr><td colspan="3">ԻՐԱԿԱՆԱՑՈՒՄ</td></tr>
<tr><td colspan="3">Իրականացման գործընթացի նախագծում</td></tr>
<tr><td>Կոմպետենցիա</td><td>ԲԱ: Մասնակցում է ծրագրի իրականացման քայլերի մշակման խմբի աշխատանքներին՝ իրականացնելով մասնագիտական ոլորտին առնչվող առաջադրանքներ:</td><td>ՄԱ: Պատախանատու է ծրագրի իրականացման քայլերի մշակման համար: Աշխատանքները կարող է կատարել խմբի հետ միասին և ղեկավարել այն:</td></tr>
<tr><td>Հմտություն</td><td colspan="2">• Որոշում է նպատակները և չափման եղանակը իրականացման կատարողականի, գնի և որակի համար
• Ընտրում է իրականացման համակարգի նախագիծը</td></tr>
</table>

Սարքավորման արտադրման գործընթաց		
Կոմպետենցիա	ԲԱ: Մասնակցում է սարքավորումների արտադրության ամբողջ գործընթացին: Խնդիրները ստանում է առաջադրանքների տեսքով և իրականացնում մասնագիտական ոլորտին բնորոշ աշխատանքներ:	ՄԱ: Պատասխանատու է սարքավորումների արտադրության ամբողջ գործընթացի համար սկսած առանձին փոքր մասերի արտադրությունից մինչև ամբողջական ապրանքը:
Հմտություն	• Նկարագրում է մասերի արտադրությունը • Նկարագրում է մասերի միակցումը ավելի մեծ կառույցների մեջ	• Սահմանում է թույլատրելի շեղումները, փոփոխականությունը, հիմնական բնութագրերը և գործառույթի վիճակագրական վերահսկումը
Ծրագրային ապահովման իրականացման գործընթաց		
Կոմպետենցիա	ԲԱ: Մասնակցում է ծրագրային ապահովման իրականացմանը: Լուծում է մոդուլային դիզային, ծրագրավորման լեզվի հետ կապված խնդիրներ:	ՄԱ: Պատասխանատու է ծրագրային ապահովման իրականացման համար ներառյալ ծրագրի մասնիկների միջև փոխհարաբերությունների բնույթի սահմանումը և իրականացման լեզուն:

Հմտություն	• Բացատրում է բարձր մակարդակի կոմպոնենտների տրոհումը մոդուլների նախագծերի (ներառյալ ալգորիթմները և տվյալների կառույցները) • Քննարկում է ալգորիթմները (տվյալների կառույցները, կառավարման ընթացքը և տվյալների հոսքը)	• Քննարկում է ծրագրավորման լեզուն • Կատարում է ցածր մակարդակի նախագծում (կոդի ծրագրավորում) • Նկարագրում է համակարգի մեկնարկումը
Սարքերի և ծրագրային ապահովման ինտեգրում		
Կոմպետենցիա	ԲԱ: Մասնակցում է սարքավորումների և ծրագրային ապահովման ինտեգրացիայի գործընթացին՝ լուծելով առաջադրանքներ, որոնք վերաբերում են նրա մասնագիտական գործունեության բնագավառին:	ՄԱ: Ղեկավարում է սարքավորումների և ծրագրային ապահովման ինտեգրացիայի գործընթացը՝ համաձայնեցնելով իր գործողությունները այլ ոլորտի մասնագետների հետ:
Հմտություն	• Նկարագրում է ծրագրային համակարգի ինտեգրումը էլեկտրոնային համակարգի մեջ (պրոցեսորի հզորությունը, հաղորդակցումները և այլն) • Նկարագրում է ծրագրային համակարգի ինտեգրումը սենսորների, գործարկողների և մեխանիկական սարքավորումների հետ	• Նկարագրում է ծրագրային և էլեկտրոնային համակարգի աշխատանքը և անվտանգությունը

Թեստ, ստուգում, վավերացում, հավաստագրում		
Կոմպետենցիա	ԲԱ: Մասնակցում է ապրանքի կամ ծառայության համապատասխանության հավաստման գործընթացին: Աշխատում է խմբում՝ ստանալով հանձնարարականներ առաջադրանքների տեսքով:	ՄԱ: Հավաստում է ապրանքի կամ ծառայության համապատասխանությունը՝ ղեկավարելով դրա փորձարկման գործընթացը:
Հմտություն	• Քննարկում և վերլուծում է գործընթացները (ծրագրային և էլեկտրոնային, ընդունելին և որակավորումը) • Քննարկում է համակարգի պահանջների կատարողականի ստուգումը	• Քննարկում է պատվիրատուի կարիքների կատարողականի հիմնավորումը • Բացատրում է չափորոշիչների/ ստանդարտների սերտիֆիկացումը
Իրականացման կառավարում		
Կոմպետենցիա	ԲԱ: Աշխատում է ապրանքի կամ ծառայության իրականացման գործընթացում ներգվված խմբում և իրականացնում է մասնագիտական ոլորտին վերաբերող գործունեություն՝ հստակ դրված առաջադրանքներ լուծելու տեսքով:	ՄԱ: Պատասխանատու է ապրանքի կամ ծառայության իրականացման գործընթացում ներգրավված խմբի աշխատանքի համար՝ ներառյալ աղբյուրները, համագործակցություններ և մատակարարման շղթաներ:

Հմտություն	• Նկարագրում է իրականացման կառուցվածքը և կազմակերպումը • Նկարագրում է աղբյուրները, գործընկերությունը և մատակարարման շղթան • Կարողանում է վերահսկել իրականացման արժեքը, կատարողականը և ժամանակացույցը	• Նկարագրում է որակի և անվտանգության վերահսկումը • Նկարագրում է իրականցման գործառույթի հնարավոր բարելավումները
Գործատուների արձագանքը		
	• Ընտրի ծրագրավորման նպատակահարմար կառույցներ • Թեստավորման միջավայրի ապահովում • Ապահովի ծրագրի սպասարկման հնարավորությունը • Մոդելավորի կիրառության միջավայրը • Ապահովի համակարգի կենսունակությունը կիրառության միջավայրում • Խորապես տիրապետի առնվազն մեկ ծրագրավորման լեզվի • Ծանոթ լինի համակարգերի ծրագրավորման տեխնոլոգիաներին • Պահպանի գոյություն ունեցող միջազգային ստանդարտները • Արժևորի կորպորատիվ էթիկան • Գնահատի ծրագրի ամբողջականությունը	
ԳՈՐԾԱՌՆՈՒԹՅՈՒՆ		
Գործառնման նախագծում և օպտիմալացում		

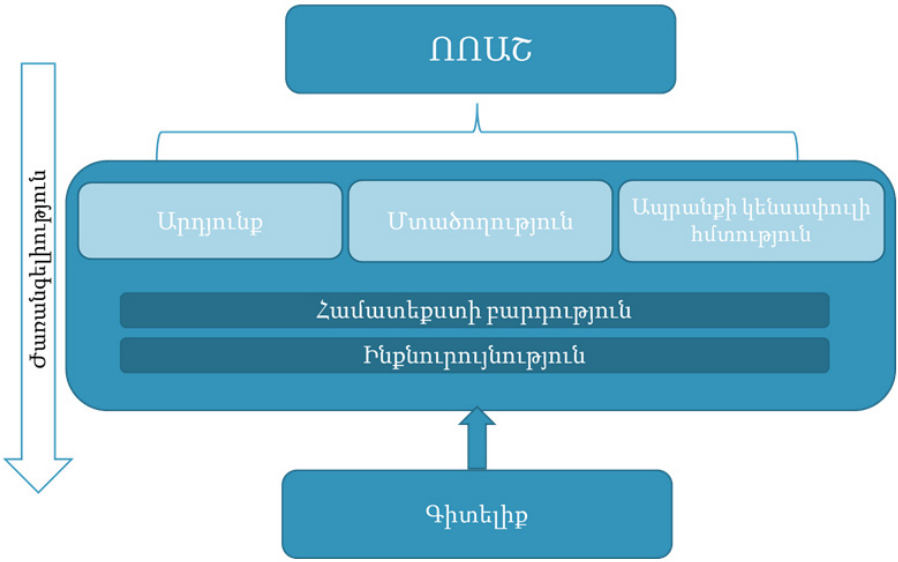
Կոմպետենցիա	ԲԱ: Ներգրավված է ինժեներական ապրանքի կիրառման նախագծի մշակման գործընթացում և լուծում է տեխնիկական առաջադրանքներ կապված արդյունավետության ցուցիչների, ժամկետների և նպատակների սահմանման հետ:	ՄԱ: Պատասխանատու է ինժեներական ապրանքի կիրառման նախագծի/գործառույթների մշակման համար ներառյալ դրա ժամկետների, արդյունավետության ցուցիչների, նպատակների սահմանումը:
Հմտություն	• Մեկնաբանում է գործառնական կատարողականի, գնի և ծավալի չափման եղանակը և նպատակները • Բացատրում է գործարկման գործառույթի ճարտարապետությունը և մշակումը	• Բացատրում է գործառնության վերլուծությունը և մոդելավորումը
Վերապատրաստում և գործառնում		
Կոմպետենցիա	ԲԱ: Ներգրավված է սպառողների կրթական կարիքներին համապատասխան ծրագրերի ստեղծմանը:	ՄԱ: Պատասխանատվություն է կրում ապրանքի առանձնահատկություններից բխող սպառողների կրթական ծրագրերը կազմելու համար:

<table>
<tr><td>Հմտություն</td><td>• Նկարագրում է արհեստավարժ գործառնման համար վերապատրաստումները՝
o Նմանարկում/մոդելավորում,
o Դասավանդում և ծրագրեր,
o Գործառույթներ/գործողություններ</td><td>• Ճանաչում է հաճախորդի գոծառնման համար անհրաժեշտ կրթությունը/որակավորումը
• Նկարագրում է գործառնման գործընթացը
• Ճանաչում է գործառնման գործընթացների փոխազդեցությունը</td></tr>
<tr><td colspan="3">Համակարգի կյանքի շրջափուլի աջակցում</td></tr>
<tr><td>Կոմպետենցիա</td><td>ԲԱ: Մասնակցում է ապրանքի կենսափուլի իրականացման աջակցման գործընթացին՝ իրականացնելով մասնագիտական ոլորտին առընչվող վերլուծություններ:</td><td>ՄԱ: Ղեկավարում է ապրանքի/համակարգի կենսափուլի իրականացման աջակցման գործընթացը: Պատասխանատվություն է կրում դրանում ներգրավված խմբի աշխատանքների համար:</td></tr>
<tr><td>Հմտություն</td><td>• Բացատրում է սպասարկումը և կազմակերպչականը
• Նկարագրում է կյանքի շրջափուլի արդյունավետությունը և հուսալիությունը
• Նկարագրում է կյանքի շրջափուլի ծավալը և արժեքը</td><td>• Մեկնաբանում է հետադարձ կապը համակարգի լավարկմանը/բարելավմանը աջակցելու/ուղղորդելու համար</td></tr>
<tr><td colspan="3">Համակարգի բարելավում և էվոլյուցիա</td></tr>
</table>

Կոմպետենցիա	ԲԱ: Մասնակցում է ապրանքի կամ ծառայության բարելավման ծրագրի մշակման խմբի աշխատանքներին և իրականացնում է մասնագիտական խորհրդատվություն:	ՄԱ: Պատասխանատվություն է կրում ապրանքի կամ ծառայության բարելավման ծրագրի մշակման խմբի աշխատանքների համար:
Հմտություն	• Սահմանում է նախապես պլանավորված (ըստ պլանի նախագծված) արդյունքի/արտադրանքի բարելավումը • Ճանաչում է բարելավման անհրաժեշտությունը գործառնության ժամանակ կարիքների դիտարկման հիման վրա	• Ճանաչում է համակարգի էվոլյուցիոն արդիականացումը/նորացումը • Ճանաչում է գործառնական անհրաժեշտությունից բխող չնախատեսված բարելավումները/լուծումները
Գործառնման կառավարում		
Կոմպետենցիա	ԲԱ: Ներգրավված է ապրանքի կիրառումը իրականացնող խմբի աշխատանքներում: Աշխատում է թիմում և համագործակցում է հստակ սահմանված առաջադրանքներ լուծելու միջոցով:	ՄԱ: Պատասխանատու է ապրանքի կամ ծառայության կիրառման ամբողջ գործընթացի համար: Աշխատում է թիմը ղեկավարելով՝ սահմանված լիազորությունների սահմաններում:

Հմտություն	• Նկարագրում է գործառնման համար անհրաժեշտ կազմակերպումը և կառուցվածքը • Բացահայտում է գործընկերություն և միություններ • Ժանաչում է գործառնման արժեքի, կատարման և ժամանակցույցի վերահսկումը	• Նկարագրում է որակի և անվտանգության ապահովումը • Նկարագրում է կյանքի շրջափուլի կառավարումը • Ճանաչում է գործառնման գործառույթի հնարավոր բարելավումները
Գործատուների արձագանքը		
	• Փոփոխվող պահանջների հասցեագրում • Անոմալիաների բացահայտում և հասցեագրում • Սխալների բացահայտում և խմբավորում • Փոփոխությունների դասակարգում և լուծումների իրականացում • Փաստաթղթերի նորացում	

Հավելված 1. Ֆոկուս խմբային քննարկման անցկացման ուղեցույց



1. **Ապրանք**

- Ո՞րոնք են մասնագիտությանը բնորոշ ապրանքները:

2. Ինչն է հանդիսանում ոլորտի մասնագիտական գործունեության արդյունքը կամ ապրանքը:

- Արդյունքի նկարագիր
- Արդյունքի դասակարգում

3. **Մտածելակերպ**

- Ինչպիսի՞ մտածելակերպ է բնորոշ ոլորտի մասնագետին մասնագիտական խնդիրներ լուծելու ընթացքում:

4. **Ապրանքի կենսափուլ**

- Որո՞նք են այն կոմպետենցիաները (սահմանումը՝

մեթոդաբանության մեջ), որոնք նպաստում են փուլին բնորոշ խնդիրների իրականացմանը:

- Որո՞նք են այն հմտությունները (սահմանումը՝ մեթոդաբանության մեջ), որոնք նպաստում են փուլին բնորոշ խնդիրների իրականացմանը:

5. Գիտելիքի մարմին (Body of Knowledge/BOK)

- Ինչպիսի՞ գիտելիք (առարկաներ) է անհրաժեշտ ոլորտի մասնագետին մասնագիտական խնդիրներ լուծելու ընթացքում:

Հավելված 2. Ֆոկուս խմբային քննարկման ՄԱՍՆԱԿԻՑ ԿԱԶՄԱԿԵՐՊՈՒԹՅՈՒՆՆԵՐ

1. ԷՍԷՖԷԼ (SFL Pro)
2. Հայկական ծրագրեր (Amenian Software)
3. Սիներջի ինթերնեյշնլ սիսթեմզ (Synergy International Systems)
4. Սինոփսիս Արմենիա (Synopsis Armenia)
5. Մակադամիան առ (Macadamian AR)
6. ՎՕԼՕ (Volo)
7. Ինստիգեյթ (Instigate)
8. Լևիաթան (Leviathan)
9. Սի Քյու Ջի Այ Մա (CQGI MA)
10. Օ-Էմ-Դի (One Market Data)
11. ԱՅՅՈՒՆեթվորքս (IUNetworks)
12. Ինֆորմացիոն Տեխնոլոգիաների Ձեռնարկությունների Միություն ՀԿ (Union of Information Technology Enterprises (UITE) NGO)

Հավելված 3. ԳՐԱԿԱՆՈՒԹՅԱՆ ՈՒՍՈՒՄՆԱՍԻՐՈՒԹՅՈՒՆ

Software engineering and types of product used in software

IEEE Computer Society has the Software and Systems Engineering Vocabulary by whose definition:

Software and software product. (1) computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system **(2)** program or set of programs used to run a computer, include both executable and non-executable software, such as fonts, graphics, audio and video recordings, templates, dictionaries, documents, and information structures such as database records.

Software is more than just a **program code**. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called ***software product***.

Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable **software product**.

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

The term ***Software Engineering*** is made of two words, **software** and **engineering**.

Software engineering requires an understanding, not only of the unique characteristics of “software”, but also of how “engineering” concepts must be adapted to address those characteristics.

IEEE defines software engineering as:

1. *The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.*

2. *The study of approaches as in the above statement.*

Fritz Bauer, a German computer scientist, defines software engineering as:

The establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

Main logic of software developed by **computer science** specialists. Finally, software control the computer and it is a program.

There are many indirect reflections about software engineering in literature which give us felling of concepts what is software and how to do its engineering.

Character of product in this sector is not so usual and that unusual is a reason of approaches which used in production, distribution, operation and maintenance specific not way.

From daily practice software engineering is the process of solving customers' needs by the systematic development and evolution of large, high-quality software systems within cost, time and other constraints (normally being in deficit of that resources).

Software engineering is a systematic design and development of software products and the management of the software process.

Software engineering has as one of its primary objectives the production of **programs** that meet specifications, and are demonstrably accurate, produced on time, and within budget.

Software product development is a symbiosis of two way of thinking: ***computing*** **(programming)** and **engineering of (programming) process**.

Computing has scholar approach with values like quality of program which depending on different effectiveness criteria: minimum resources like time, speed; or correctness of results: min of bugs, etc. Engineering approach is selecting one of approaches as practically useful/appropriate for specific case and organize the way when scholar criteria used practically and no need for looking the best for each component if it is difficult prove effectiveness. How to organize work to have demonstrably accurate, produced on time, and within budget is responsible engineering. The nature of any software product development is **computer programming**, but development of compound software products as systems is different, because development of relatively big program become engineering. Software products are now among the *most complex manmade systems*, with high rate of *changeability.*

CONTEXT OF SOFTWARE ENGINEERING THINKING

Despite of big successes in increasing computer effectiveness and its technical features nowadays developers met serious problems in terms of the development *costs, timeliness, and quality* of many software products.

Techniques and processes of making the software that work effectively when used by an individual or *small team* to develop modest-sized programs do not *scale well* to the development of ***large, complex systems***. **Development complexity** can arise with just a few hundred lines of code, and *large systems* can run to *millions of lines of code*, requiring *years of work by hundreds of software developers*. Complexity of organization rise dramatically.

The *change in computer and software technology* drives the

demand for *new and evolved software products*. This situation has created customer expectations and competitive forces that tension our ability to produce quality software within acceptable development schedules. Fast change in hardware bring fast competitive change in soft for earlier sale reasons, but to have reliability is difficult.

New software is not always maintenance, but it is new development. Testing is difficulty.

The availability of *qualified software engineers* has not kept pace with the demand from industry, so that *systems* are designed and built by people with insufficient educational background or experience. *Software system* developed in today may *well reuse* major components of other systems, execute on *multiple machines* and platforms, and interact with other, globally distributed systems.

Software engineering is *more than just programming*; it includes attention to details such as *quality, schedule, and economic goals.* Hence, a professional *software developer needs both knowledge of such principles and experience with applying* them.

Software is *abstract* and *invisible*. These characteristics present challenges for managing software development because of the problems they create for important engineering concepts such as *quantification and measurement*. They also complicate efforts to describe and organize software in ways that will facilitate knowledge exchange during the processes of its design, implementation, and maintenance.

Software has both static and dynamic properties. This duality provides a further challenge for description and measurement. It also makes it difficult to predict the effects arising from any changes made to a software product.

Software is intrinsically complex in terms of its organization. Even a small software unit may possess many different execution paths,

and there may be a large and complex set of relationships among its elements. This in turn presents *challenges for verification* and validation, documentation, and maintenance.

No universal *measures of quality exist* for assessing a software product. An engineering process should lead to products that are of "good" quality, but the relative importance of different *quality measures* will vary with the *role of the product* and differ for each stakeholder (producer, customer, or user).

The manufacturing cycle for software products is not a significant element in software development, and it mainly involves the *needs of distribution mechanisms*. Software development is essentially a process that involves a *progression through many layers of design abstraction* and, hence, is unlike any conventional engineering processes, such as those that occur within mechanical and civil engineering.

Software is intangible because hard to understand development effort.

Software is easy to reproduce Cost is in its development, in other engineering products, manufacturing is the costly stage.

The software industry is labor-intensive, hard to automate.

(Untrained) people can hack something together, quality problems are hard to notice.

Software is easy to modify, people make changes without fully understanding it.

Software does not 'wear out', it deteriorates by having its design changed:

- erroneously, or
- in ways that were not anticipated, thus making it complex

Software does *not wear out*. The maintenance activities associated with a software product are really part of an evolutionary design process.

Software engineering practices are largely concerned with *managing relevant processes* and with *design activities*, and these can appear in a range of appearances.

Most of the activities involved in software development and evolution tend to use team-based processes that embody some form of design element, spanning from an initial choice of a high-level architecture through the choices of test and evaluation strategies. Each of these adds yet another layer of complication: teams must be organized with regard to aspects such as communication, coordination, and management and design activities are nondeterministic (not clear) processes that lead to solutions that are rarely right or wrong.

Finally, there are also many different measures of quality that can be employed when assessing the different choices involved. Differentiation of employers internally can be if we ask people who busy with effective algorithms and reusable software design in their mind. The same time managers and architects think from another perspective.

Software is an intangible economic good, with no physical form, its utility or value not even perceptible in another form. So only the functionality of software is perceptible e.g. via a user interface, or as the result of a controlled transaction via software, e.g. as an account movement.

What exactly constitutes a software product is often rather subjective. As with many highly technical products, many people do not understand how software products work. Software is therefore in the truest sense of the word “intangible.” Software thus contrasts greatly with other investments or acquisitions of consumer goods. In particular, the customer does not really acquire the product when buying software, rather very specific, precisely defined rights of use in a license contract.

However, investments in software today represent a larger proportion of spending for IT infrastructure than investments in hardware, and software contracts with large companies often amount to multiple millions of dollars.

Software belongs not to the three classic economic factors of *capital, land, and labor*, but in the new fourth category of "*knowledge*."

Software is the manifestation of human know-how in bits and bytes and in this form also has the invaluable advantage (and simultaneous disadvantage), that it can be easily copied and quickly circulated over any distance.

The **right software**, ideally applied, can represent a more important strategic competitive advantage in today's economic life than all the other factors. Software can be crucial for competitiveness in production processes, functionality, the availability of service products, and thus for a company's success or failure on the market.

Engineering and not engineering approach to software development

Whereas scientists observe and study existing behaviors and then develop models to describe those, engineers use such models as a starting point for designing and developing technologies that enable new forms of behavior.

Engineers proceed by making a series of decisions, carefully evaluating options, and choosing an approach at each decision point that is appropriate for the current task in the current context. Appropriateness can be judged by trade-off analysis, which balances costs against benefits.

Engineers measure things, and when appropriate, work quantitatively. They calibrate and validate their measurements, and

they use approximations based on experience and empirical data.

Engineers emphasize the use of a disciplined process when creating and implementing designs and can operate effectively as part of a team in doing so.

Engineers can have multiple roles: research, development, design, production, testing, construction, operations, and management in addition to others such as sales, consulting, and teaching.

Engineers use tools to apply processes systematically. Therefore, the choice and use of appropriate tools is a key aspect of engineering.

Engineers, via their professional societies, advance by the development and validation of principles, standards, and best practices.

Engineers reuse designs and design artifacts.

Although strong similarities exist between software engineering and more traditional engineering, there are also some differences (not necessarily to the detriment of software engineering):

Software engineering's foundations are primarily in computer science, not natural sciences.

Software engineering models make more use of discrete than continuous mathematics.

The concentration is on abstract/logical entities instead of concrete/physical artifacts.

There is no manufacturing phase in the traditional sense.

Software maintenance primarily refers to continued development, or evolution, and not to conventional wear and tear.

Software engineering is not always viewed as a "professional" activity. One concern for these curriculum guidelines is to help with the evolution of software engineering toward a more "professional" status.

SOFTWARE PRODUCT LIFE CYCLE

Software Evolution

The process of developing a software product using software engineering principles and methods is referred to as software evolution. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.

Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

Software Evolution Laws

Lehman has given laws for software evolution. He divided the software into three different categories:

- First is a software, which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. This type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.

- Second is a software with a collection of procedures. This is defined by exactly what procedure can do. In this software, the specification can de described but the practical type solution is not obvious instantly. For example, gaming software.
- Third type software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, online trading software.
- Lehman has given eight laws for third type software evolution -
- Continuing change - software system must continue to adapt to the real world changes, else it becomes progressively less useful.
- Increasing complexity - software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.
- Conservation of familiarity - The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system.
- Continuing growth- In order for a software system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
- Reducing quality - software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
- Feedback systems- software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
- Self-regulation - system evolution processes are self-regulating with the distribution of product and process measures close to normal.

- Organizational stability - The average effective global activity rate in an evolving system is invariant over the lifetime of the product.

Software Paradigms

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:

Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

Software Development Paradigm

This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build.

Waterfall Model

Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.

Iterative Model

This model leads the software development process in iterations. It projects the process of development in cyclic manner repeating every step after every cycle of SDLC process.

The software is first developed on very small scale and all the

steps are followed which are taken into consideration. Then, on every next iteration, more features and modules are designed, coded, tested and added to the software. Every cycle produces a software, which is complete in itself and has more features and capabilities than that of the previous one.

After each iteration, the management team can do work on risk management and prepare for the next iteration. Because a cycle includes small portion of whole software process, it is easier to manage the development process but it consumes more resources.

Spiral Model

Spiral model is a combination of both, iterative model and one of the SDLC model. It can be seen as if you choose one SDLC model and combine it with cyclic process.

This model considers risk, which often goes un-noticed by most other models. The model starts with determining objectives and constraints of the software at the start of one iteration. Next phase is of prototyping the software. This includes risk analysis. Then one standard SDLC model is used to build the software. In the fourth phase of the plan of next iteration is prepared.

V – Model

The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages. V-Model provides means of testing of software at each stage in reverse manner.

At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product

is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.

This makes both verification and validation go in parallel. This model is also known as verification and validation model.

Big Bang Model

This model is the simplest model in its form. It requires little planning, lots of programming and lots of funds. This model is conceptualized around the big bang of universe. As scientists say that after big bang lots of galaxies, planets and stars evolved just as an event. Likewise, if we put together lots of programming and funds, you may achieve the best software product.

For this model, very small amount of planning is required. It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary.

This model is not suitable for large software projects but good one for learning and experimenting.

Need of Software Engineering

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

- Large software - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
- Scalability- If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.

- Cost- As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- Dynamic Nature- The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.

Need of software project management

Software is said to be an intangible product. Software development is a kind of all new stream in world business and there's very little experience in building software products. Most software products are tailor made to fit client's requirements. The most important is that the underlying technology changes and advances so frequently and rapidly that experience of one product may not be applied to the other one. All such business and environmental constraints bring risk in software development hence it is essential to manage software projects efficiently.

The image above shows triple constraints for software projects. It is an essential part of software organization to deliver quality product, keeping the cost within client's budget constrain and deliver the project as per scheduled. There are several factors, both internal and external, which may impact this triple constrain triangle. Any of three factor can severely impact the other two.

Therefore, software project management is essential to incorporate user requirements along with budget and time constraints.

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in **software engineering to develop the intended software product**.

SDLC Activities

SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps:

Communication

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization in writing.

Requirement Gathering

This step onwards the software development team works to carry on the project. The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements.

Feasibility Study

After requirement gathering, the team comes up with a rough plan of software process. At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful. It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.

System Analysis

At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project. System analysis includes Understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc. The project team analyzes the scope of the project and plans the schedule and resources accordingly.

Software Design

Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product. The inputs from users and information gathered in requirement gathering phase are the inputs of this step. The output of this step comes in the form of two designs; logical design and physical design. Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams and in some cases pseudo codes.

Coding

This step is also known as programming phase. The implementation

of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

An estimate says that 50% of whole software development process should be tested. Errors may ruin the software from critical level to its own removal. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user's end. Early discovery of errors and their remedy is the key to reliable software.

Integration

Software may need to be integrated with the libraries, databases and other programs. This stage of SDLC is involved in the integration of software with outer world entities.

Implementation

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational. The software is maintained timely

by updating the code according to the changes taking place in user end environment or technology. This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition

As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense upgradation. Hence a pressing need to eliminate a major portion of the system arises. This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time.

CDIO Life cycle in enterprise

1 **CONCEIVING AND ENGINEERING SYSTEMS**

1.1 **Setting System Goals and Requirements**

Identify market needs and opportunities

Elicit and interpret customer needs

Identify opportunities that derive from new technology or latent needs

Explain factors that set the context of the requirements

Identify enterprise goals, strategies, capabilities and alliances

Locate and classify competitors and benchmarking information

Interpret ethical, social, environmental, legal and regulatory influences

Explain the probability of change in the factors that influence the system, its goals and resources available

Interpret system goals and requirements

Identify the language/format of goals and requirements Interpret initial target goals (based on needs, opportunities and other influences)

Explain system performance metrics

Interpret requirement completeness and consistency

1.2 **Defining Function, Concept and Architecture**

Identify necessary system functions (and behavioral specifications)

Select system concepts

Identify the appropriate level of technology

Analyze trade-offs among and recombination of concepts

Identify high level architectural form and structure

Discuss the decomposition of form into elements, assignment of function to elements, and definition of interfaces

1.3 **Modeling of System and Ensuring Goals Can Be Met**

Locate appropriate models of technical performance

Discuss the concept of implementation and operations

Discuss life cycle value and costs (design, implementation, operations, opportunity, etc.)

Discuss trade-offs among various goals, function, concept and structure and iteration until convergence

1.4 **Development Project Management**

Describe project control for cost, performance, and schedule

Explain appropriate transition points and reviews

Explain configuration management and documentation

Interpret performance compared to baseline

Define earned value process

Discuss the estimation and allocation of resources

Identify risks and alternatives

Describe possible development process improvements

2 DESIGNING

2.1 The Design Process

Choose requirements for each element or component derived from system level goals and requirements

Analyze alternatives in design

Select the initial design

Use prototypes and test articles in design development

Execute appropriate optimization in the presence of constraints

Demonstrate iteration until convergence

Synthesize the final design

Demonstrate accommodation of changing requirements

2.2 The Design Process

Phasing and Approaches

Explain the activities in the phases of system design (e.g. conceptual, preliminary, and detailed design)

Discuss process models appropriate for particular development projects (waterfall, spiral, concurrent, etc.)

Discuss the process for single, platform and derivative products

2.3 Utilization of Knowledge in Design

Utilize technical and scientific knowledge

Practice creative and critical thinking, and problem solving

Discuss prior work in the field, standardization and reuse of designs (including reverse engineer and redesign)

Discuss design knowledge capture

2.4 **Disciplinary Design**

Choose appropriate techniques, tools, and processes

Explain design tool calibration and validation

Practice quantitative analysis of alternatives

Practice modeling, simulation and test

Discuss analytical refinement of the design

2.5 **Multidisciplinary Design**

Identify interactions between disciplines

Identify dissimilar conventions and assumptions

Explain differences in the maturity of disciplinary models

Explain multidisciplinary design environments

Explain multidisciplinary design

2.6 **Multi-Objective Design (DFX)**

Demonstrate design for: Performance, life cycle cost and value

Aesthetics and human factors

Implementation, verification, test and environmental sustainability

Operations Maintainability, reliability, and safety

Robustness, evolution, product improvement and retirement

3 **IMPLEMENTING**

3.1 **Designing the Implementation Process**

State the goals and metrics for implementation performance, cost and quality

Recognize the implementation system design:

3.2 **Hardware Manufacturing Process**

Describe the manufacturing of parts

Describe the assembly of parts into larger constructs

Define tolerances, variability, key characteristics and statistical process control

3.3 **Software Implementing Process**

Explain the breakdown of high-level components into module designs (including algorithms and data structures)

Discuss algorithms (data structures, control flow, data flow)

Describe the programming language

Execute the low-level design (coding)

Describe the system build

3.4 **Hardware Software Integration**

Describe the integration of software in electronic hardware (size of processor, communications, etc.)

Describe the integration of software integration with sensor, actuators and mechanical hardware

Describe hardware/software function and safety

3.5 **Test, Verification, Validation, and Certification**

Discuss test and analysis procedures (hardware vs. software, acceptance vs. qualification)

Discuss the verification of performance to system requirements

Discuss the validation of performance to customer needs

Explain the certification to standards

3.6 **Implementation Management**

Describe the organization and structure for implementation

Discuss sourcing, partnering, and supply chains

Recognize control of implementation cost, performance and schedule

Describe quality and safety assurance

Describe possible implementation process improvements

4 OPERATING

4.1 Designing and Optimizing Operations

Interpret the goals and metrics for operational performance, cost, and value

Explain operations process architecture and development

Explain operations (and mission) analysis and modeling

4.2 Training and Operations

Describe training for professional operations:

Simulation Instruction and programs

Procedures

Recognize education for consumer operation

Describe operations processes

Recognize operations process interactions

4.3 Supporting the System Lifecycle

Explain maintenance and logistics

Describe lifecycle performance and reliability

Describe lifecycle value and costs

Explain feedback to facilitate system improvement

4.4 System Improvement and Evolution

Define pre-planned product improvement

Recognize improvements based on needs observed in operation

Recognize evolutionary system upgrades

Recognize contingency improvements/solutions resulting from operational necessity

4.5 **Disposal and Life-End Issues**

Define the end of life issues

List disposal options

Define residual value at life-end

List environmental considerations for disposal

4.6 **Operations Management**

Describe the organization and structure for operations

Recognize partnerships and alliances

Recognize control of operations cost, performance and scheduling

Describe quality and safety assurance

Define life cycle management

Recognize possible operations process improvements

Professional thinking in Software Engineering

At a broad level, the expected characteristics of computer science graduates include the following [CS2013]:

Technical understanding of computer science

Graduates should have a mastery of computer science as described by the core of the Body of Knowledge.

Familiarity with common themes and principles

Graduates need understanding of a number of recurring themes, such as abstraction, complexity, and evolutionary change, and a set of general principles, such as sharing a common resource, security, and concurrency. Graduates should recognize that these themes and principles have broad application to the field of computer science and should not consider them as relevant only to the domains in which they were introduced.

Appreciation of the interplay between theory and practice

A fundamental aspect of computer science is understanding the interplay between theory and practice and the essential links between them. Graduates of a computer science program need to understand how theory and practice influence each other.

System-level perspective

Graduates of a computer science program need to think at multiple levels of detail and abstraction. This understanding should transcend the implementation details of the various components to encompass an appreciation for the structure of computer systems and the processes involved in their construction and analysis. They need to recognize

the context in which a computer system may function, including its interactions with people and the physical world.

Problem solving skills

Graduates need to understand how to apply the knowledge they have gained to solve real problems, not just write code and move bits. They should to be able to design and improve a system based on a quantitative and qualitative assessment of its functionality, usability and performance. They should realize that there are multiple solutions to a given problem and that selecting among them is not a purely technical activity, as these solutions will have a real impact on people's lives. Graduates also should be able to communicate their solution to others, including why and how a solution solves the problem and what assumptions were made.

Project experience

To ensure that graduates can successfully apply the knowledge they have gained, all graduates of computer science programs should have been involved in at least one substantial project. In most cases, this experience will be a software development project, but other experiences are also appropriate in particular circumstances. Such projects should challenge students by being integrative, requiring evaluation of potential solutions, and requiring work on a larger scale than typical course projects. Students should have opportunities to develop their interpersonal communication skills as part of their project experience.

Commitment to life-long learning

Graduates should realize that the computing field advances at a rapid pace, and graduates must possess a solid foundation that allows and encourages them to maintain relevant skills as the field evolves.

Specific languages and technology platforms change over time. Therefore, graduates need to realize that they must continue to learn and adapt their skills throughout their careers. To develop this ability, students should be exposed to multiple programming languages, tools, paradigms, and technologies as well as the fundamental underlying principles throughout their education. In addition, graduates are now expected to manage their own career development and advancement. Graduates seeking career advancement often engage in professional development activities, such as certifications, management training, or obtaining domain-specific knowledge.

Commitment to professional responsibility

Graduates should recognize the social, legal, ethical, and cultural issues inherent in the discipline of computing. They must further recognize that social, legal, and ethical standards vary internationally. They should be knowledgeable about the interplay of ethical issues, technical problems, and aesthetic values that play an important part in the development of computing systems. Practitioners must understand their individual and collective responsibility and the possible consequences of failure. They must understand their own limitations as well as the limitations of their tools.

Communication and organizational skills

Graduates should have the ability to make effective presentations to a range of audiences about technical problems and their solutions. This may involve face-to-face, written, or electronic communication. They should be prepared to work effectively as members of teams. Graduates should be able to manage their own learning and development, including managing time, priorities, and progress.

Awareness of the broad applicability of computing

Platforms range from embedded micro-sensors to high-performance clusters and distributed clouds. Computer applications impact nearly every aspect of modern life. Graduates should understand the full range of opportunities available in computing.

Appreciation of domain-specific knowledge

Graduates should understand that computing interacts with many different domains. Solutions to many problems require both computing skills and domain knowledge. Therefore, graduates need to be able to communicate with, and learn from, experts from different domains throughout their careers.

Guidance on Learning Outcomes [CS2013]

Learning outcomes are not of equal size and do not have a uniform mapping to curriculum hours; topics with the same number of hours may have quite different numbers of associated learning outcomes. Each learning outcome has an associated level of mastery. In defining different levels, we drew from other curriculum approaches, especially Bloom's Taxonomy, which has been well explored within computer science. We did not directly apply Bloom's levels in part because several of them are driven by pedagogic context, which would introduce too much plurality in a document of this kind; in part because we intend the mastery levels to be indicative and not to impose theoretical constraint on users of this document.

We use three levels of mastery, defined as:

● ***Familiarity***: The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept

as opposed to expecting real facility with its application. It provides an answer to the question “What do you know about this?”

- ***Usage***: The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question “What do you know how to do?”
- ***Assessment***: The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question “Why would you do that?”

As a concrete, although admittedly simplistic, example of these levels of mastery, we consider the notion of iteration in software development, for example for-loops, while-loops, and iterators. At the level of “Familiarity,” a student would be expected to have a definition of the concept of iteration in software development and know why it is a useful technique. In order to show mastery at the “Usage” level, a student should be able to write a program properly using a form of iteration. Understanding iteration at the “Assessment” level would require a student to understand multiple methods for iteration and be able to appropriately select among them for different applications. The descriptions we have included for learning outcomes may not exactly match those used by institutions, in either specifics or emphasis. Institutions may have different learning outcomes that capture the same level of mastery and intent for a given topic. Nevertheless, we believe that by giving descriptive learning outcomes, we both make

our intention clear and facilitate interpretation of what outcomes mean in the context of a particular curriculum.

Software engineering competency

Software engineering program educational objectives [ANNA]:

1. Apply software engineering theory, principles, tools and processes, as well as the theory and principles of computer science and mathematics, to the development and maintenance of complex, scalable software systems.
2. Design and experiment with software prototypes
3. Select and use software metrics
4. Communicate effectively through oral and written reports, and software documentation
5. Elicit, analyze and specify software requirements through a productive working relationship with project stakeholders
6. Demonstrate professionalism including continued learning and professional activities.
7. Contribute to society by behaving ethically and responsibly.
8. Successfully assume a variety of roles in teams of diverse membership.
9. Apply a systematic, disciplined, quantifiable approach to the cost-effective development, operation and maintenance of software systems to the satisfaction of their beneficiaries.
10. Build solutions using different technologies, architectures and life-cycle approaches in the context of different organizational structures.
11. Insist the development, adoption and sustained use of standards of excellence for software engineering practices.

Software engineering program outcomes:

- Upon completion of the course, students would have obtained:
- An ability to apply knowledge of mathematics, science, and engineering.
- An ability to design and conduct experiments, as well as to analyze and interpret data.
- An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, safety, and sustainability.
- Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
- An ability to identify, formulate, and solve engineering problems.
- An understanding of professional and ethical responsibility.
- An ability to communicate effectively.
- Demonstrate a knowledge and understanding of management and business practices, such as risk and change management, and understand their limitations.
- A recognition of the need for, and an ability to engage in life-long learning.
- An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.
- An understanding of real-time, safety-critical, embedded computer systems.

Computer science and engineering program educational objectives [ANNA]:

Graduates of this M. E. Computer Science and Engineering will be able to

1. Apply the necessary mathematical tools and fundamental & advanced knowledge of computer science & engineering
2. Develop computer, software, network systems understanding

the importance of social, business, technical, environmental, and human context in which the systems would work

3. Articulate fundamental concepts, design underpinnings of computer/software/network systems, and research findings to train professionals or to educate engineering students
4. Contribute effectively as a team member/leader, using common tools and environment, in computer science and engineering projects, research, or education
5. Pursue life-long learning and research in selected fields of computer science & engineering and contribute to the growth of those fields and society at large

Program outcomes:

- Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the conceptualization of engineering models.
- Identify, formulate, research literature and solve complex engineering problems reaching substantiated conclusions using first principles of mathematics and engineering sciences.
- Design solutions for complex engineering problems and design systems, components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.
- Conduct investigations of complex problems including design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions.
- Create, select and apply appropriate techniques, resources, and modern engineering tools, including prediction and modeling, to complex engineering activities, with an understanding of the limitations.

- Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.
- Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- Demonstrate understanding of the societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to engineering practice.
- Understand and commit to professional ethics and responsibilities and norms of engineering practice.
- Understand the impact of engineering solutions in a societal context and demonstrate knowledge of and need for sustainable development.
- Demonstrate a knowledge and understanding of management and business practices, such as risk and change management, and understand their limitations.
- Recognize the need for, and have the ability to engage in independent and life-long learning.

QUALITIES OF BEHAVIOR EVERY SOFTWARE ENGINEER SHOULD HAVE

Technical Skills

1. Basic Computer Science Skills: Research skills, reading comprehension, the ability to know how to use library functions, and understanding computing problems, design patterns, and frameworks. develop skills in logical thinking, creative problem-solving and communication. Team approach, requiring clear communication among members to solve a problem and explain solution to others.
2. Passion for Code: Programmer must have a passion for code, developing it from a purely scientific skill into a craft or an art. Building code with the popularity of Open Source, don't think alone in code creation — the ability to work with software engineers and developers from around the world.
3. Fearless Refactoring: Refactoring is the ability to improve code without changing what it does. The old code can become unstable and incompatible over time. Refactoring enables the developer to own the code.
4. Develops Quality: Some engineers thought testing was lower them. Experienced engineers know and understand the value of tests, because their goal is to create a working system. Exposing bugs and eliminating them is the best way to develop quality code. Engineer knows not to waste time writing trivial or redundant tests, instead focusing on testing the essential parts of each component.
5. Willing to Leverage Existing Code: Life is too short to continuously invent new codes and libraries. Reuse of internal infrastructure, use of third-party libraries, and leveraging web-

scale known services.

6. Focus on Usable and Maintainable Code: Software always works better then it is well designed and user-centric. Good engineers work hard to make the system simple and usable. They think about customers all the time and do not try to invent convoluted stuff that can only be understood and appreciated by geeks. A disciplined engineer thinks about the maintainability and evolution of the code from its first line, as well. Expressive names for methods and variables can make the code self-explanatory.
7. Can Code in Multiple Languages: A willingness to learn new languages, new libraries and new ways of building systems goes a long way to creating a great software engineer.

Personal Traits

8. Vision: Visionaries create code and libraries that are open to refactoring in time, and easy to use in all code languages. Being able to see the impacts of present-day decisions is paramount to building great software.
9. Attention to Detail: If you get angry about misspelled database columns, “uncommented” code, projects that aren’t checked into source control, software that’s not unit tested, unimplemented features, and so on, then you probably try to avoid those issues yourself. Bad installation packages, sloppy deployments, or a misspelled column name can bring down entire systems. Be obsessive about details.
10. Business Acumen: Understand and value why your software development is so important to your clients’ livelihoods. “This software never crashes. It never needs to be re-booted. This

software is bug-free. It is perfect, as perfect as human beings have achieved. Consider these stats: the last three versions of the program — each 420,000 lines long-had just one error each. The last 11 versions of this software had a total of 17 errors. Commercial programs of equivalent complexity would have 5,000 errors." The ability to understand why all the coding is done, as it the fruit for any customer or client.

11. Curiosity: The best software engineers are curious about why something is done one way or another, yet with the added ability of being objective about the solutions. Many engineers we know got in trouble as kids for taking things apart to see how they worked. Putting together software is just a creative, and many software engineers also have artistic hobbies. This creativity and curiosity is required to think outside the box when designing programs. The thrill you get from making something work is what keeps you going.
12. Experience: If you've been tinkering with software programs since you were a kid, your abilities as an adult will be quadrupled. Beyond hands-on experience, you might also be addicted to math and science, and the ability to stay organized. At the same time, great software engineers also realize that they don't know it all…the ability to continue to learn is essential in a field where change is a constant.
13. Discipline: Although you may have passion for your job, this love for your work and for the next project doesn't mean that you can be sloppy. Attention to detail is important, but so is an ability to stay organized. So much bad code belongs to developers who don't do what they know should be done.
14. Patience: Bugs are natural. Design glitches are normal. Sloppy coding by other engineers occurs often. Patience is a key quality

for software engineers who want to work in this field.

15. Teamwork: Few projects are small enough or require so few skills that one person can do them well. Learning how to work as a team in college is one way to get over that "hermit" image…and working as a team online or in the office can only produce stellar projects. Successful engineers also become good communicators. They know how to write clear and concise reports and instructions, and know how to convey ideas to clients and customers

SUMMARY OF VARIOUS RECOMMENDATIONS ABOUT DESIRED COMPETENCIES OF SOFTWARE DEVELOPERS

SE thinking

1. Ability to accommodate himself to others, empathy, "be the customer" mentality – genuine interest in understanding what other people are trying to accomplish and based on this understanding think about creating technical solutions to help them reach their goals. Genuine interest in understanding "why to create software" and the broader context of software systems.
2. Cognitive task analysis. Appreciation of unstated requirement and ability to identify these. Listening skills, approachable, and respect for people.
3. Ability to work in homogeneous, multi-disciplinary, multi-locational and multicultural teams. Ability to work under supervision and constraints, Understanding of the impact of personal character and behaviors on others.

4. Ability to apply knowledge, ability to integrate the application of knowledge, skills, and sense of responsibilities to new settings and complex problems.
5. Ability to see the self as bound to all humans with ties of recognition and concern. Seek help from other, Ability to help and assist others, mentoring, commitment to others' success in their business. Sensitivity towards global, societal, environmental, moral, ethical and professional issues, and sustainability. Respect for the intellectual property of others. Work ethics.
6. Abstraction and transition between levels of abstraction, representation skills spatial and temporal modeling skills, structuring skills, and theorizing.
7. Algorithmic and structured thinking. Logic, pattern matching, logical what-if analysis, problem decomposition and synthesis, etc.
8. Analytical skills.
9. Communication skills.
10. Constructive criticism.
11. Curiosity, interest in 'how things work' and 'how to create things that work,' interest in the power of technology, humility, observation skills, ability to see things as they are, broader understanding and interests, respect for the classic authors of the great books, openness to constructive criticism, value and readiness for lifelong learning. Active listening skills. Ability to develop a very good understanding of domain specific vocabulary, its semantics, and established thinking patterns.
12. Decision making skills.
13. Design skills.
14. Domain competence.
15. Entrepreneurship, intrinsic motivation to create something, desire to improve things, initiative taking, enjoy challenges, sense of

mission, perseverance, concentration, result orientation, commitment, self motivation, dedication, and hard work. Adaptability, flexibility, open-mindedness, and ability to multi-task. Sense of urgency and stress management.

16. Experimentation skills.
17. Good grasping power and attention to detail: breadth, depth, clarity, accuracy, preciseness, specificity, relevance, significance, completeness, consistency.
18. Imagination: storyboarding, extrapolation, visualization, cognitive flexibly: ability to transfer and models of solutions of one situation/field to another, multi-perspective thinking, lateral thinking, inductive thinking, out-of-box thinking, unstructured thinking, creativity and idea initiation, and innovation.
19. Knowledge of contemporary issues and business practices.
20. Knowledge of physical and natural world. Intercultural knowledge.
21. Mentoring, coaching, and training skills.
22. Organizational skills.
23. Persuasion, negotiation, consensus building, and conflict resolution skills.
24. Problem orientation, problem definition and formulation, generations of alternatives. Ability to convert ill-defined problematic situations into software solvable problem. Ability of infusing different thinking patterns developed through their experience in other domains. Inclination for reuse and synthesis by integration. Emphasis on elegant and simple solutions.
25. Problem solving skills: solution implementation and verification.
26. Project planning and management, project scoping, estimation, process planning and management,
27. Quality, cost, and security consciousness, pursuit of excellence,

intellectual accountability and responsibility, intellectual integrity, intellectual courage, strength of conviction: assertive without being aggressive. Commitment to systematic documentation of the work. Recognize and act upon the need to consult other experts, especially in matters outside their area of competence and experience. Commitment to the fulfillment of needs of all users and persons who get affected by the technological solutions. Eagerness and inclination to understand the unintended consequences of creating software inappropriate or at odds to its real purposes. Commitment to health, safety, dignity, and welfare of the users and also the people who will be affected by their systems. Sensitivity towards constraints like economic disadvantage and physical disabilities that may limit software accessibility.

28. Reasoning: quantitative and verbal, and critical thinking: ability to question, validate, and correct the purpose, problem, assumptions, perspectives, methods, evidence, inference, reliability, relevance, criteria, and consequences. Numerical ability.
29. Reflection and transition between ladders of reflection. Meta-cognition.
30. Research skills: methods of mathematical research, engineering research, design research, and social science research.
31. Self-acceptance, self-regulation, self-awareness, self-improvement: strength to resist instant gratification in order to achieve better results tomorrow. Being honest and forthright about one's own limitations of competence. Tendency to avoid false, speculative, vacuous, deceptive, misleading, or doubtful claims. Faith in reason and review, inclination for verification and validation, respect for facts and data. Awareness and regulation of automatic thoughts.

32. Systems-level perspective, 'big picture' view, holistic and multi-perspective thinking, knowledge integration, consideration for multilateral viewpoint, and user-centeredness. Process and rule-oriented mindset. Tolerance to ambiguity and risk. Ability to understand and also build upon other's work. Ability to work such that others can easily understand and build upon.
33. Technical competence to solve the software solvable problems using tools and techniques, Use of open source software. Knowledge of industry's best practices and standards, appreciation of what is technically feasible. Identify the risk level of each piece of work.
34. Wealth creation skills.
35. Work load management.
36. Reversibility to safe state in any stage of project development: Responsibility for client who leave his safe stage/state and use new automation.

Safe mean, he has tools for full manipulation with system/data in a familiar way.

37. Invisible has visible images: models or prototypes.

Clients are very natural life oriented: in their mind invisible soft again has visible behavior. New software is new invisibility for client. Client should trust to safety of change/of new status. He does it by images linking or helping to link with reality. SE should give examples of reality, for example by presentations shows that developer understand that reality the same way like clients themselves.

Customer leave his safe state with trust to you. Give the evidences to him.

Customer trust to tangible things, but here it is even invisible. Psychological problems step-in.

COMPUTATIONAL THINKING

Typically, degree programs in computing include study of the nature of computation, with effective ways to exploit computation, and with the practical limitations of computation in application terms. The concept of computational thinking captures these concerns. Computational thinking is a basic analytical ability that has relevance (which every computing graduate should recognize) in many aspects of everyday life.

Its main characteristics include:

- algorithmic thinking including recursive, distributed and parallel possibilities and attention to the benefits and the limitations of these; the role of these in devising approaches to areas of system design, problem solving, artificial intelligence, simulation and modelling
- recognition of the relationships between the concepts of specification, program and data (in all its forms), as well as the power of the notion of transformation and proof, and the place of these in computing
- understanding the power behind abstraction, the potential of multiple levels of abstraction and the role this plays in computing
- understanding the opportunities for and the potential of automation, but also the proper balance between automation and how humans effectively interact with computers
- recognizing the role of redundancy, diversity and separation of concerns in achieving reliable, safe and critical systems - often in the presence of uncertainty and approaches to achieving this
- recognizing simplicity and elegance as useful concepts and principles on the one hand, but also bad and dangerous practices on the other.

Balance of theory and practice

- problem definition, specification, design, implementation and maintenance
- the necessary knowledge, including an understanding of the range of possible options for these tasks
- the data structures and algorithms
- the related practical and transferable skills, including relevant approaches to group activity
- access to the appropriate resources, including tools
- the necessary underpinning to guide practice to ensure the sustainability of their knowledge and to provide an appropriate framework that will accommodate rapid technological change. The necessary underpinning may come also from diverse disciplines including mathematics, empirical or experimental insight, engineering, psychology, aesthetics, organizational theory, management or graphical design.

Key Competencies

- the concept of computational thinking, the recognition of its main elements and the relevance of these to everyday life
- the computing system (and this includes systems such as information systems), and the process of developing or analyzing it is important; understanding of the system and its operation will go deeper than a mere external appreciation of what the system does or the way(s) in which it is used
- there is a balance of practice and theory, appropriate to the aims of the particular degree program, such that practical activity can be supported by an understanding of underlying principles.

Subject-related cognitive abilities

- Computational thinking including its relevance to everyday life.
- Knowledge and understanding: demonstrate knowledge and understanding of essential facts, concepts, principles and theories relating to computing and computer applications as appropriate to the program of study.
- Modelling: use such knowledge and understanding in the modelling and design of computer-based systems for the purposes of comprehension, communication, prediction and the understanding of trade-offs.
- Requirements, practical constraints and computer-based systems (and this includes computer systems, information systems, embedded systems and distributed systems) in their context: recognize and analyze criteria and specifications appropriate to specific problems, and plan strategies for their solution.
- Critical evaluation and testing: analyze the extent to which a computer-based system meets the criteria defined for its current use and future development.
- Methods and tools: deploy appropriate theory, practices and tools for the specification, design, implementation and evaluation of computer-based systems.
- Reflection and communication: present succinctly to a range of audiences (orally, electronically or in writing) rational and reasoned arguments that address a given information handling problem or opportunity. This should include assessment of the impact of new technologies.
- Professional considerations: recognize the professional, economic, social, environmental, moral and ethical issues involved in the sustainable exploitation of computer technology

and be guided by the adoption of appropriate professional, ethical and legal practices.

Subject-related practical abilities

- The ability to specify, design and construct computer-based systems.
- The ability to evaluate systems in terms of general quality attributes and possible trade-offs presented within the given problem.
- The ability to recognize any risks or safety aspects that may be involved in the operation of computing equipment within a given context.
- The ability to deploy effectively the tools used for the construction and documentation of computer applications, with particular emphasis on understanding the whole process involved in the effective deployment of computers to solve practical problems.
- The ability to operate computing equipment effectively, taking into account its logical and physical properties.

Additional transferable skills

- Effective information-retrieval skills (including the use of browsers, search engines and catalogues).
- Numeracy and literacy in both understanding and presenting cases involving a quantitative and qualitative dimension.
- Effective use of general information technology (IT) facilities.
- The ability to work as a member of a development team, recognizing the different roles within a team and different ways of organizing teams.
- Managing one's own learning and development including time management and organizational skills.

- Appreciating the need for continuing professional development in recognition of the need for lifelong learning

Professional thinking in Computer Science

What is Computational Thinking?

Computer scientists tend to approach problems in a particular way. Because programming is fundamental to computer science education, computer scientists tend to think like programmers. That is, they look for algorithmic solutions to problems, in terms of data manipulation and process control. In a widely cited paper in 2006, Jeanette Wing termed this computational thinking[1], and argued that this practice may be the most important contribution computer science makes to the world, and that it should be taught to all students in all disciplines.

This task force offers the following definition:

"Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data.
- Representing data through abstractions such as models and simulations.
- Automating solutions through algorithmic thinking (a series of ordered steps).
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
- Generalizing and transferring this problem solving process to a wide variety of problems."

[1] J. Wing, "Computational Thinking," Communications of the ACM, vol. 49, no. 3, pp. 33–35, 2006.

These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution

Computational Thinking (CT) is a process that generalizes a solution to open ended problems. Open-ended problems encourage full, meaningful answers based on multiple variables, which require using decomposition, data representation, generalization, modeling, and algorithms found in **Computational Thinking**.

BODY OF KNOWLEDGE

Body of knowledge [QAA, Computing07]

The following list of topics is seen as defining the scope of the broad area of computing. It is not intended to define curricula or syllabi; it is merely provided as a set of knowledge areas indicative of the technical areas within computing.

Architecture

The computer processing unit/memory/input-output (IO) model, representation of data and programs in memory, arithmetic/logic unit, fetch-execute cycle, registers, stacks, data-paths, special IO-support hardware, support for protection and virtual memory, instruction sets, implementation constraints and trade-offs, historical, current and future trends. Peripheral devices, interfacing. Cache memory and memory hierarchies.

Measuring performance. High performance and parallel architectures; pipeline processors, array processors and single instruction multiple data (SIMD) architectures, shared-memory multiple instruction multiple data (MIMD) machines (symmetrical multi-processor and non-uniform memory access), message-passing MIMD machines.

Artificial intelligence

This is a discipline with two strands. The scientific strand attempts to understand the requirements for and mechanisms enabling intelligence of various kinds in humans, other animals and information processing machines and robots. The engineering strand attempts to apply such knowledge in designing useful new kinds of machines and helping us to deal more effectively with natural intelligence, eg in education and therapy. Knowledge elicitation and representation. Constraint-based programming.

Uncertainty. Cognitive modelling. Logics. Reasoning. Deduction and theorem proving.

Search. Machine learning. Agent technology. Planning. Vision systems, robotics. Speech and language technology.

Comparative programming languages

The variety of languages and the motivation for this variety, including languages such as scripting languages. Design criteria for languages. Desirable properties of languages and their implementations. Different programming paradigms: imperative, object-oriented, functional, logic, visual. Concurrency, parallelism and distributed computing. Strengths, weaknesses of different language features including types and data modelling, control structures, structuring concepts, abstraction mechanisms, parameterization, exception handling, separate compilation, generics. Declarations, naming conventions, storage allocation strategies; parameter passing mechanisms.

Compilers and syntax directed tools

Aims in compiler construction. Variation in possible users. Features of languages. Phases of development. Lexical analysis. Parsing: LL grammars, LR parsing, parse trees and abstract syntax trees. Symbol tables. Type checking. Semantic analysis. Run-time storage organization. Code generation. Code optimization. Illustration of other syntax directed tools.

Computational science

Combinatory, integer and linear programming, optimization, operational research, e-science, computer algebra.

Computer-based systems

Definition of computer-based systems. Different kinds of systems: to include embedded systems, real time systems, distributed systems, client-server systems. Safety critical and other high-integrity systems: risk analysis and assessment. Systems approach. Modelling.

Needs, goals and objectives; requirements definitions; functional analysis and derivation of non-functional requirements; specification development; evaluation of trade-offs and alternatives leading to formulation of system architecture; allocation of responsibilities leading to sub-system design and interface definitions. Co-design issues. Problem of integration, configuration management, quality assurance, operations and maintenance.

Performance measures. Forensics.

Computer communications

Digital communication: standards, media, signaling, reliability, error handling and performance. Device management, input/output

considerations. Communications management. Communications software.

Computer hardware engineering

Specification, design (using electronic computer-aided design and hardware description languages), simulation, verification, construction and testing of the hardware of computer systems using appropriate technologies for logic, memory, storage and communication (with users and other machines). Understanding future technology trends and the requirements placed by software systems on computer hardware.

Computer networks

Networks: topologies, protocols and standards. Different communication media and data, and related requirements. Reference models, switching, access, security, compression, encryption, mobile operation, quality of service, performance, management, interconnection and architectural models, routing, congestion, firewalls, and proxy servers. Network operating system design. Future trends: emerging technologies and applications.

Computer vision and image processing

The design of computer algorithms and hardware to model the structure and properties of visual data. Modelling techniques and algorithms: human vision system based, engineering perspective based. The extraction and application of information from these models. Image processing: pattern recognition, the manipulation of the image signal to include image analysis: the extraction of semantic data, animation, manipulating images.

Concurrency and parallelism

Nature of concurrency, problems. Examples, including IO. Concurrent processes,

Inter-process communication. Low level synchronization primitives. Language primitives for shared memory. Concurrency at operating system, language level. Atomic actions.

Scheduling and real-time systems. Performance issues. Resource allocation and deadlock.

Concurrency control and recovery. Classification of parallel machines. Algorithms and algorithm design in the context of parallelism. Complexity and performance metrics associated with algorithms in the context of concurrent systems.

Databases

The concept of a database and database management. Database development.

Illustrations. Entity-relationship model. Database design: logical design and the relational model, physical design. Normalization; different normal forms. Client-server model.

Structured query language (SQL) and database servers. Database access and client applications. Extensible mark-up language (XML). Object oriented systems, multimedia database systems, distributed database systems. Spatial databases and geographic positioning systems. Database administration. Data mining, data warehousing.

Databases and websites. Tools.

Data structures and algorithms

Data types, structures and abstract data types. Efficiency measures (average and worst case), rates of growth, asymptotic behavior.

Algorithmic paradigms (including enumeration, divide-and-conquer, greedy, dynamic programming, tree search, probabilistic). Algorithm design and analysis with correctness proofs. Data processing algorithms (sorting, searching, hash tables etc.). Data mining. Numerical algorithms and analysis; statistical algorithms and simulation. Graph theory and graph theoretic algorithms (shortest paths, spanning trees, etc.). Symbolic computation. Other application areas, eg sequencing, scheduling and assignment. Parallel and distributed algorithms, implementation issues and efficiency measures.

Developing technologies

For example, quantum computing, bio-informatics, medical computing, ubiquitous computing, computer forensics, e-science, autonomic computing, grid computing, high performance computing.

Distributed computer systems

Characteristics of distributed systems, client-server model, interposes communication, remote procedure calls, distributed operating systems, naming and protection, file service design, shared data and transactions, concurrency and control, time coordination and time-stamping, replication, fault handling and recovery, distributed system security.

Computer supported collaborative work.

Document processing

Word processing systems, design and development. Related tools: editors, spelling checkers, grammar checkers. Mixed systems including tables, diagrams, and pictures.

Presentation systems. Web-based documents: the benefits and drawbacks. Design and development of web-based documents. E-Publishing, digital typography. Document engineering. Mark-up

languages. Multimedia presentation. Contents and index generation. Copyright and other legal issues.

E-Business

Nature of e-business: the particular cases of e-commerce, e-learning. Impact on practices and on organizations. Business process re-engineering. Levels of automation. Distributed transactions, security and privacy. Particular problems. Major components in such a system. Hard and soft e-commerce. Business-to-business and business-to-customer technologies. Digital signatures and authentication issues. Legal and ethical issues.

Need for deep knowledge in certain application areas.

Empirical approaches

Experiments: their role and purpose. Experimental design, hypothesis formulation and hypothesis testing. Empirical methods.

Games computing

Computer games programming, games design, interaction, relevant aspects of advanced computer graphics, relevant mathematics for computer games, and artificial intelligence for computer games, online gaming and networking, console architectures, professional issues for games. More generally, computers in entertainment.

Graphics and sound

Human perception of images, display and image-capture technology, storage formats and algorithms for the manipulation of two-dimensional (2D) and three-dimensional

(3D) representation, transformations on images, geometric

modelling, animation, rendering with realistic lighting and texture effects. Human perception of sound, frequency versus time domain representations, sound compression, synthesis, sound analysis. 2D and 3D modelling, animation, virtual reality, multimedia. Scientific and information visualization. Computational geometry. Object modelling. Animation.

Human-computer interaction (HCI)

User interface engineering: user-centered design and evaluation methodologies, architectures, IO modes (including multi-modal) and devices, development environments, interface managers, construction skills; HCI guidelines, principles and standards; interaction styles, metaphors and conceptual models. User models: human psychology and actions, ergonomics, human information processing. Human-computer applications: including virtual and connected environments (including mobile), games, visualization, multimedia, affective computing, systems for users with special needs.

Usability engineering and evaluation.

Information retrieval

Information and its management. Transmission issues. Text, graphics, speech, sound, documents and other kinds of content. Methods of retrieval for different content.

Methods based on logic, and probability theory, situation theory, computational logistics. Web-based considerations. Interactive considerations including feedback issues.

Case-based reasoning, hypertext, visualization.

Information systems

Theoretical underpinnings. Data, information and knowledge

management. Information in organizational decision making. Integration of information systems with organizational strategy and development. Information systems design. Systems approaches.

Compression technologies. Development, implementation and maintenance of information systems. Information and communications technologies (ICT). Decision support. Management of information systems and services. Content management systems. Organizational and social effects of ICT-based information systems. Economic benefits of ICT-based information systems. Personal information systems. Digital libraries.

Intelligent information systems technologies

Theory, design and development of database systems, database applications, data warehouses, data mining principles, decision support system development including intelligence density (quality, models, constraints, and organizational factors), decision trees, genetic algorithms, neural networks, fuzzy logic, case based reasoning, information presentation.

Management issues

Software project management, management and economic aspects of outsourcing and off-shoring. Aspects of managing IT systems, eg documents, resources, networks, websites. The management of risk.

Middleware

Examples of objects, and object libraries. Characteristics of well-designed and high-quality objects. Design guidelines. Methods of ensuring quality. Building new classes in accord with the guidelines. Design patterns. Design languages. Tool support. Mechanisms for interconnection of classes and modules. Integration as a concept

and as a vehicle for system enhancement. Mechanisms for achieving integration. Interconnection languages: scripting languages. Building systems in this environment, including distributed systems.

Verification and validation of such systems. Applications.

Multimedia

Multimedia seen as the capabilities of modern computer technology to employ multiple-media communication forms (including data, text, graphics, still and video images and sound) integrated into single applications. Distinguished from other forms of multiple-media by the fact that the computer reduces all information into a digital form that can be reproduced, manipulated, stored and transmitted electronically.

Consideration of the representation, storage and transmission issues for different digital forms, and the subsequent transformations of these forms. Operations. Design and development issues. User interface and presentational matters. Tools support.

Natural language computing

Advanced computing techniques to enhance the capabilities of systems providing text and speech. Communication. Language generation, language models, parsing and understanding, machine translation. Advanced models of interpersonal and human-computer dialogue; advanced methods for language processing by providing robust, accurate and efficient treatment of language in a range of applications and of user-situations. Speech recognition and synthesis. Text analysis.

Operating systems

Role, functions of operating systems. Characteristics, capabilities of single user systems, multi-user systems. Illustrations. Process concept. Architecture of an operating system: influences of networks,

multimedia, security. File processing systems. Resource management. Basic services - memory management, interrupt handling, process scheduling. Concurrency mechanisms. Scheduling. System processes - spooler, network interface; utilities. Security and protection issues including access control, virus protection. Shell programming. Relationship to window systems.

Professionalism

Ethics: consideration of the individual, organizational and societal context in which computing systems are planned, developed and used; deployment of technical knowledge and skills with a concern for the public good. Ethical responsibilities.

Techniques of ethical analysis. Social aspects of computing. Computer crime. Law: awareness of relevant law and processes of law eg data protection, computer misuse, copyright, intellectual property rights, basic company and contract law. Systems: development and operational costs; safety/mission criticality; consequences and liability issues of failure; risk analysis; security; recovery. Professional bodies: structure, function, restriction of title, license to practice, codes of ethics/conduct/practice.

Programming fundamentals

The nature of programming. Use of some well-designed and appropriate programming language. The idea of syntax and semantics, and related errors. Programming paradigms. Problem analysis, program design, coding including interface considerations.

Simple programs and simple algorithms. Abstraction mechanisms, parameter passing.

Simple quality considerations, including strategies for testing and debugging. Use of libraries. Different kinds of documentation serving different purposes.

Security and privacy

Security and privacy: the problems. Illustrations of how problems arise. Physical and logical security. Machine access. Protection mechanisms. Encryption and encryption building blocks. Virtual private networks. Legal issues. Firewalls and internet security.

Monitoring of traffic and computer use. Digital signatures. e-commerce, e-banking and related applications.

Simulation and modelling

Uses of modelling and simulation. Benefits and drawbacks. Model classification, systems theory. Continuous and discrete simulation. Applications. Simulation languages. Model building. Model validation. Different approaches and different types of simulation.

Software engineering

Development paradigms; requirements elicitation/specification; analysis and design (including architectural design and design patterns); system models; programming paradigms; prototyping and evolution; testing; verification and validation; assessment and evaluation; software reuse; software measurement and metrics; operation and maintenance; quality assurance and management; configuration management; formal description techniques; software dependability; tools (including CASE) and environments; software process models; implementation; documentation. Agile methods, open-source software development.

Systems analysis and design

Systems theory. Systems within an organization. Different kinds of systems serving different purposes, e.g. autonomic, distributed, ubiquitous. Systems in support of an enterprise which is potentially

complex and may have to adapt. Typical computer systems lifecycles. Systems requirements and specification. Feasibility concerns. System design: strengths and weaknesses of relevant methodologies and techniques. Fault tolerance. People and interface issues. Compliance with legal and ethical standards.

Development, implementation and maintenance. Quality considerations.

Theoretical computing

Models of computation, computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of programming, theorem proving, software specification, data types and data structures, coding theory, theory of databases and knowledge based systems, models of concurrency, statistical models of system performance, formal methods of system development. The subject also includes the development of the mathematical techniques used in the list above.

Web-based computing

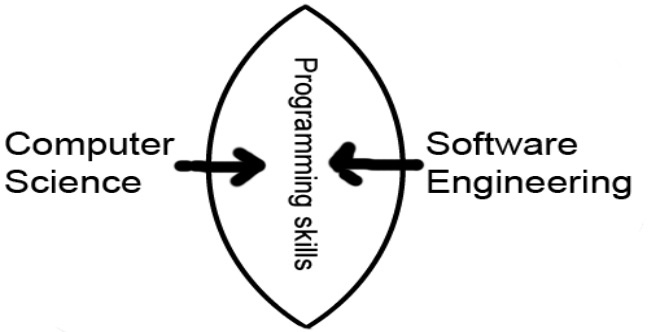
The specification, design, implementation and operation of web-based technologies and services: currently wired and wireless internet protocol (IP) protocol-based technologies, mark-up languages, HCI, branding and brand loyalty. Web scripting languages/systems.

Mobile computing. Enterprise systems: intranets and extranets: access control, security, authentication, encryption, intellectual property rights (IPR), costing, pricing, charging and funding. Tools. Server selection, installation, configuration and administration. Logs and traffic analysis. Searching and search engines. IPR and copyright. Impact of networked economy at regional, national and international levels.

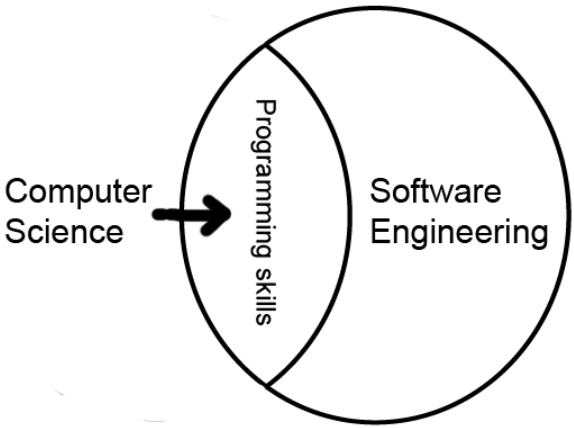
ORGANIZATION OF EDUCATION IN SECTOR ON BACHOLAR LEVEL

Different views

Industry views[1]

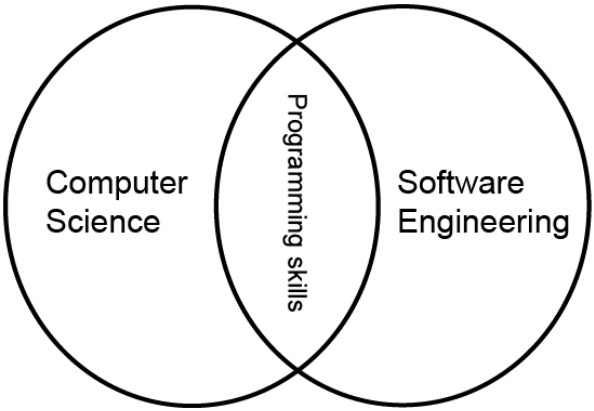


Software engineers view



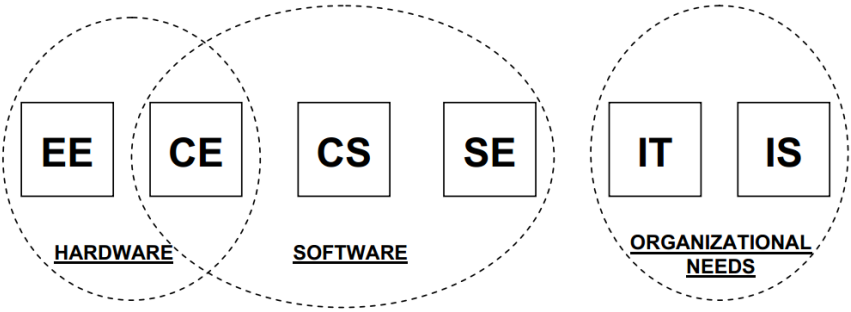
1 Source:
http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf

Reality of the situation



In practice of university education four different degrees on are where on bachelor level:

- Electrical engineering and Computer engineering
- Computer science
- Software engineering
- Information Technology and Information Systems.



VALUES IN SOFTWARE ENGINEERING

Valuing and motivations

- Reliability of systems behavior
- Covering all possible cases
- Autonomy of model behavior
- Restriction of used resources
- Productivity
- Higher interaction speed than in original object of modeling
- Prefer discrete state modeling
- Model is self-regulated in most of cases
- Portability from one basic set of actions to another
- Reusability of algorithms, states, data
- Covering universal cases
- No exceptions, planned and predictable of behavior
- Realistic reproduction of modeled reality
- Ideally reproduce human behavior, compete with human capacity, Ideal target is to have Self-reproduction
- Overcome human productivity
- Predict the findings of others, have secure systems
- Predict the possible behavior
- Interconnectivity
- Universality of connection
- Universality of functions
- High rate of asynchronous parallelism of participant, may be in geographically spread environment
- High density of information storage
- Openness of developed systems
- Concurrency control, concurrent management modeling

- Concurrency as tool to increase productivity
- Prototyping and developing
- Inheritance logic or principles
- Extension of human functions
- Translation and interpretation logic
- Continuing extension of human perception and action endpoints: eyes, ears, foots, arms, logic of thoughts, processing and decisions, talk
- Virtual spaces; Virtually of structure in calculation and addresses
- Abstract and concrete
- Mapping of abstraction levels or hierarchy
- State change automation
- Finite or Infinite; Calculation possible in infinite cycles
- Discrete and continuous
- Multidisciplinary thinking: logic of model developed in different sciences or useful for them
- Modeling of communication and extend its capacity
- Engineering logic: Natural science logic, not artificial discrete models logic when all possible
- Mathematical and human logic
- Repeatability, Valuing of reliability of processes
- Encapsulation idea, black-box of functions and data
- Atomicity of processing
- Open systems, extensions
- Open source code, reusability
- Synchrony, asynchrony, Parallelism
- Primitives and superposition of behavior
- Copying of continues logic in discrete
- Modeling of reality

- Real time, loose less
- Real time modeling by data storage
  - Storing data long term/infinite
  - Retrieving in real time
  - Universality
  - Permanent Modeling of real time change- updates of data and relations
- Entity-relationship models
- Universality of Inputs, outputs, actions
- Mass service
- Very big number of components, unimaginable for human
- Infinite and finite
- Nesting
- Stepwise refinement
- Loose-less of data, signals

QUALITIES OF GRADUATES OF AN UNDERGRADUATE SE PROGRAM

Graduates of an undergraduate SE program should be able to demonstrate the following qualities. [SE2014]

[**Professional Knowledge**] *Show mastery of software engineering knowledge and skills and of* the professional standards *necessary to begin practice as a software engineer*.

Students, through regular reinforcement and practice, need to gain confidence in their abilities as they progress through a software engineering program of study. In most instances, students acquire knowledge and skills through a staged approach in which they achieve different levels as each academic term progresses. In addition, graduates must have an understanding and appreciation of

professional issues and standards related to ethics and professional conduct, economics, and societal needs.

[**Technical Knowledge**] *Demonstrate an understanding of and apply appropriate theories, models, and techniques that provide a basis for problem identification and analysis, software design, development, implementation, verification, and documentation.*

Software engineering employs concepts that are unique to the nature of software and its development and also draws others from a range of reference disciplines. Students should both be aware of these concepts and of their limitations, whether inherent or arising from their adaptation to software engineering. Students should be able to evaluate and reflect on the processes that they follow as well as upon the solutions that they produce.

[**Teamwork**] *Work both individually and as part of a team to develop and deliver quality software artifacts.*

Students need to perform tasks that involve working as an individual, but also experience many other tasks that entail working with a group. For group work, students should be informed about the nature of groups and of group activities and roles as explicitly as possible. This must include an emphasis on the importance of such matters as a disciplined approach, adhering to deadlines, communication, and individual and team performance evaluations.

[**End-User Awareness**] *Demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment.*

A program of study should include at least one major activity that involves producing a solution for a client. Software engineers must take the view that they have to produce software that is of genuine utility. Where possible, a program should incorporate a period of industrial

experience as well as invited lectures from practicing software engineers and involvement in activities such as external software competitions. All this provides a richer experience and helps to create an environment that supports the development of high-quality software engineering graduates.

[**Design Solutions in Context**] *Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns.*

Throughout their study, students should be exposed to a variety of appropriate approaches to engineering design in the general sense and to examples of their use in developing software for different application domains. They must be able to understand the strengths and limitations of the available options and the implications of selecting specific approaches for a given situation. Their proposed design solutions must be developed within the context of ethical, social, legal, security, and economic concerns.

[**Perform Trade-Offs**] *Reconcile conflicting project objectives, finding acceptable compromises within the limitations of cost, time, knowledge, existing systems, and organizations.*

Students should engage in exercises that expose them to conflicting and changing requirements. There should be a strong real-world element present in such cases to ensure that the experience is realistic. Curriculum units should address these issues, with the aim of ensuring high-quality functional and nonfunctional requirements and a feasible software design.

[**Continuing Professional Development**] *Learn new models, techniques, and technologies as they emerge and appreciate the necessity of such continuing professional development.*

By the end of their program of study, students should show evidence of being self- motivated lifelong learners. Throughout a program of study, students should be encouraged to seek new knowledge and to appraise it for usefulness and relevance.

Չափսը 60x84 1/16:
Թուղթը օֆսեթ: Տպագրությունը՝ օֆսեթ: 8.5 տպ. մամուլ:
Տպաքանակը 125:

Տպագրված է «Գեոմափիս» ՍՊԸ-ի կողմից
http://www.geomapis.com
E-mail: info@geomapis.com

ARARAT

Armenian Coordination Agency
"University-Employer"

www.ararattempus.org

Universitat d'Alacant
Universidad de Alicante

Co-funded by the
Tempus Programme
of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.