КӘСІБИ БІЛІМ БЕРУ

А.В. РУДАКОВ

БАҒДАРЛАМАЛЫҚ ӨНІМДЕРДІ ДАЙЫНДАУ ТЕХНОЛОГИЯСЫ

ОҚУЛЫҚ

“Білім беруді дамытудың федералды институты” федералдық мемлекеттік мекемесі орта кәсіптік білім беру бағдарламаларын іске асыратын білім беру мекемелеріне оқу құралы ретінде ұсынылған.

Пікірдің тіркеу нөмірі
4 қазан 2010 жыл. ФММ «БДФИ»

10-шы басылым, қайта әзірленген және толықтандырылған

ACADEM'A

Мәскеу
«Академия» Баспа орталығы
2016

ӘОЖ 681.3.06(075.32)
КБЖ 32.973-018ші723 Р83

Бұл кітап Қазақстан Республикасының Білім және ғылым министрлігі және «Кәсіпқор» холдингі» КЕАҚ арасында жасалған шартқа сәйкес «ТжКБ жүйесі үшін шетел әдебиетін сатып алуды және аударуды ұйымдастыру жөніндегі қызметтер» мемлекеттік тапсырмасын орындау аясында қазақ тіліне аударылды. Аталған кітаптың орыс тіліндегі нұсқасы Ресей Федерациясының білім беру үдерісіне қойылатын талаптардың ескерілуімен жасалды.

Қазақстан Республикасының техникалық және кәсіптік білім беру жүйесіндегі білім беру ұйымдарының осы жағдайды ескеруі және оқу үдерісінде мазмұнды бөлімді (технология, материалдар және қажетті ақпарат) қолдануы қажет.

Аударманы «Delta Consulting Group» ЖШС жүзеге асырды, заңды мекенжайы: Астана қ., Иманов көш., 19, «Алма-Ата» БО, 809С , телефоны: 8 (7172) 78 79 29, эл. поштасы: info@dcg.kz

Пікір берушілер:

«Ғылыми аспаптар» ААО ғылыми-зерттеу бөлімінің басшысы ***Славянский О.Е;***
Санкт-Петербург электроника мектебінің (колледжінің) оқу-әдістемелік жұмыстары бойынша директордың орынбасары ***Васильева Н. А.***

Рудаков А. В.
Р83 Бағдарламалық өнімдерді дайындау технологиясы: орта кәсіби білім беру мекемелерінің студенттеріне арналған оқулық / Рудаков А.В. — 10-шы бас., қайта әзір. және тол. — М. : «Академия» баспа орталығы, 2016. — 208 б.

ISBN 978-5-4468-2655-1 (рус.)

ISBN 978-601-333-090-7 (каз.)

Оқулық орта кәсіби білім берудің Федералды мемлекеттік стандарттарының талаптарына сәйкес: «Компьютерлік жүйелердегі бағдарламалау» ПМ.01 «Компьютерлік жүйелер үшін бағдарламалық жасақтаудың бағдарламалық модульдерін дайындау» және ПМ.03 «Бағдарламалық модульдерді біріктіруге қатысу» (МДК.03.01 «Бағдарламалық жасақтаманы дайындау технологиясы») мамандықтары бойынша құрастырылған.

Оқулықта бағдарламалық өнімдерді дайындау технологиясының пайда болу тарихы, қазіргі жағдайы, ұйымдастыру қағидалары, жалпы ережелері мен болашағы қарастырылған.

Оқулық орта кәсіби білім беру мекемелерінің студенттеріне арналған.

ӘОЖ 681.3.06(075.32)
КБЖ 32.973-018ші723

ISBN 978-5-4468-2655-1 (рус.)
ISBN 978-601-333-090-1(каз.)

ҚҰРМЕТТІ ОҚЫРМАН!

Аталған оқулық «Компьютерлік жүйелерде бағдарламалау» мамандығы бойынша оқу-әдістемелік кешеннің бөлігі болып табылады және «Компьютерлік жүйелер үшін бағдарламалық жасақтаудың бағдарламалық модульдерін дайындаудың», «Бағдарламалық модульдерді біріктіруге қатысудың», (МДК.03.01 Бағдарламалық жасақтаманы дайындау технологиясының» кәсіби модульдерін игеруге арналған.

Жаңа буындағы оқу-әдістемелік кешендерге дәстүрлі және инновациялық материалдар жатады, олар жалпы білім беру және жалпы кәсіби пәндерді, кәсіби модульдерді игеруді қамтамасыз етеді. Әрбір кешен жалпы және кәсіби құзіреттіліктерді алуға қажетті және жұмыс берушінің талаптарына сәйкес оқулықтардан, оқу құралдарынан, оқыту және бақылау құралдарынан тұрады.

Оқу баспалары электронды білім беру ресурстарымен толықтырылады. Электронды ресурстар интерактивті жаттығулары және жаттықтырушылары, мультимедиялық объектілері, қосымша материалдарға сілтемелері, ғаламтор ресурстары бар тәжірибелік және теориялық модульдерден тұрады. Оларға оқу үрдісінің негізгі параметрлері бекітілетін терминологиялық сөздік және электронды журнал, жұмыс уақыты, бақылау және тәжірибелік жұмыстарын орындау нәтижелері кіреді. Электронды ресурстар оқу үрдісіне оңай кірістіріледі және әртүрлі оқу бағдарламаларына бейімделуі мүмкін.

КІРІСПЕ

Заманауи қоғамды компьютерсіз елестету мүмкін емес. Олардың біздің өмірімізге терең енгендігі соншалық, компьютерлер қолданылмайтын адамның қызмет ету саласын атау өте қиын. Осыған байланысты заманауи компьютерлердің аппараттық бөлігіне және қолданылатын бағдарламалық жасақтамаға жоғары талаптар қойылады. Негізінен бағдарламалық жасақтама немесе басқа сөзбен айтқанда бағдарламалық өнімдер компьютерді кеңінен пайдалану мүмкіндігін қамтамасыз етеді. Компьютердің бағдарламалық жасақтамасын қайта орнатсақ немесе жаңа бадарламалық өнімді орнатсақ біз осы компьютердегі жаңа тапсырмаларды орындай аламыз. Пайдаланылатын бағдарламалық өнімдер компьютердің сенімділігін және пайдаланушы жұмысының ыңғайлылығын қамтамасыз ететін белгілі критерийлерге сай болулары керек.

Алдымен компьютер аппаратурасы, тіпті ең қарапайым құраушылар дайындалды және бекітілген технологиялық үрдістерге сәйкес шығарылды, бағдарламалық өнімдерді дайындаудың белгілі бір технологиясы алғашқы кезде болған жоқ. Дайындаушылар өздерінің жеке тәжірибелеріне сүйенді, ол үшін дайындаудың майдагерлік тәсілдері қолданылады. Мұндай тәсіл бағдарламалық өнімдердің сапасында, дайындау мерзімдерінде, олардың бағасында көрініс тапты. Аталған жағдай бағдарламалау дағдарысы деп аталды. Дағдарыстан шығу үшін бағдарламалық өнімдерді дайындаудың индустриалды тәсілдерін құру қажет болды, яғни, алдыңғы қатарлы инженерлі әдістерден және бағдарламалық өнімдерді құру құралдарынан тұратын дайындау технологияларын құру.

Кейінірек бұл әдістер «бағдарламалық инженерия» деген түсінікпен біріктірілді (software engineering). Пайдалануды бағалау жүйесінің жиынтығында бағдарламалық өнімдерді дайындауда аталған технологияны құру бағдарламалық жасақтамалардың сенімділігін, оларды дайындау сапасын арттырды және құрастырушыларға қажетті бағдарламалық өнімді таңдауды жеңілдетті.

1 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМНІҢ ӨМІРШЕҢДІК ЦИКЛЫ

1.1. Бағдарламалық өнімнің өміршеңдік циклы түсінігі

Бағдарламалық өнім (БӨ) компьютерлік бағдарламалар, процедуралар және олармен байланысты құжаттар мен деректердің жиынтығын құрайды.

Бағдарламалық өнімнің өміршеңдік циклы — БӨ құру қажеттілігі туралы шешім қабылданған сәттен бастап, оны пайдаланудан толығымен шығаруға дейінгі мерзім аралығы.

БӨ-ді құру кезінде орындалуы тиіс БӨ-нің өміршеңдік циклының құрылымы, үрдістер құрамы, әрекеттер мен тапсырмалар ISO/IEC 12207 халықаралық стандартын анықтайды және реттейді:1995 «Information Technology — Software Life Cycle Processes» (ISO — International Organization for Standartization — Халықаралық стандарттау ұйымы; IEC — International Electrotechnical Commission — Электротехника бойынша халықаралық комиссия; стандарт атауы – «Ақпараттық технологиялар – Бағдарламаның өміршеңдік циклының үрдістері»).

Үрдіс дегеніміз - келген деректерді шығыс деректеріне түрлендіретін өзара байланысқан әрекеттер жиынтығы. Әрбір үрдіс белгілі тапсырмалармен және оларды шешу әдістерімен, сонымен бірге басқа үрдістерден алынатын бастапқы деректермен, нәтижелермен сипатталады.

Әрбір үрдіс әрекеттер жинағына, әрбір әрекет тапсырмалар жинағына бөлінеді. Үрдісті іске қосу және орындау, әрекеттер немесе тапсырмалар басқа үрдістермен жүзеге асырылады.

Ресейде 1970 жылдардан бастап БӨ-ді құру БҚБЖ стандарттарымен реттелген болатын (Бағдарламалық құжаттаманың бірыңғай жүйесі – МЕМСТ 19.ХХХ), олар жеке бағдарламашылармен құрылатын шағын көлемдегі қарапайым бағдарламаларға бағытталған. Қазіргі таңда аталған стандарттар формасы және тұжырымдамалық негізі бойынша ескірді және олардың қызмет ету мерзімдері аяқталды, бұл стандарттарды ары қарай пайдаланудың қажеті қалған жоқ. Әрбір маңызды жоба үшін нақты қолданбалы БӨ-ді құру үрдістерін реттейтін нормативті және әдістемелік құжаттар жиынтығын құруға тура келеді, сондықтан отандық дайындамаларда заманауи

халықаралық стандарттарды пайдаланған дұрыс.

ISO/IEC 12207 стандарттарына сәйкес, БӨ-нің өміршеңдік циклының барлық үрдістері үш негізгі топқа бөлінген: негізгі үрдістер, қосымша (қолдаушы) үрдістер, ұйымдастыру үрдістері

1.2. Бағдарламалық өнімнің өміршеңдік циклының негізгі үрдістері

Негізгі үрдістерге БӨ-нің өміршеңдік циклында орындалуы тиіс нақты әрекеттер жинағы және олармен байланысты тапсырмалар жатады.

Негізгілеріне алу, жеткізу, дайындау, пайдалану және сүйемелдеу үрдістері жатады.

Өнімді алу үрдістері (acquisition process) БӨ-ді алу бойынша тапсырыс беруші әрекеттерінен тұрады. Бұл әрекеттерге келесілер жатады:

- алынғандарға бастамашылық ету;
- өтінімдік ұсыныстарды дайындау;
- шартты дайындау және түзету;
- жеткізуші қызметін бақылау;
- қабылдау және аяқтау.

Алынғандарға бастамашылық ету келесі тапсырмалардан тұрады:

- БӨ немесе қызметтерді, жүйені дайындау, алу немесе жетілдірудегі тапсырыс берушінің қажеттіліктерін анықтау;
- жүйе талаптарын талдау;
- қолданылатын БӨ-ді дайындау, алу немесе жетілдіруге қатысты шешім қабылдау;
- БӨ-ді алған жағдайда қажетті құжаттамаларды, кепілдіктерді, сертификаттарды, лицензияларды, қолдауларды тексеру;
- жүйе талаптарынан, келісімшарт түрінен, тараптар жауапкершілігінен тұратын өнімді алу жоспарын дайындау және бекіту.

Нормативті құжаттарға сәйкес «жүйе» түсінігін екі түрлі түсіндіруге болады. Бірінші жағдайда жүйе ретінде аппаратты, бағдарламалық, материалды және адами ресурстар, қызметтер мен деректер, бір сөзбен айтқанда дайындау немесе сатып алуға қажетті заттар жиынтығы қарастырылады.

Екінші жағдайда жүйе дегеніміз – бұл бірігіп қызмет ететін шекті өнімдер мен дайындау, жеткізу, игеруге қажетті қосымша өнімдер жиынтығы.

Өтінімді ұсыныстарды дайындау төмендегідей ұсыныстарды дайындау және құрудан тұрады:

- дайындалатын немесе сатып алынатын жүйе талаптары;
- қажетті БӨ тізімі;
- шарттар мен келісімдер;

■ техникалық шектеулер (мысалы, жүйенің нақты қызмет ету жүйесін көрсету).

Өтінімдік ұсыныстар таңдалған жеткізушіге жіберіледі (немесе тендер кезінде бірнеше жеткізушілерге). Жеткізуші ретінде жүйені, БӨ-ді немесе бағдарламалық қызметті келісімшарттағы шарттар негізінде тапсырыс берушімен жеткізу келісімшартын орындайтын ұйым қарастырылады.

Келісімшартты дайындау және түзету мынадай тапсырмалардан тұрады:

■ болжамды жеткізушілер ұсынысын бағалау шарттарынан тұратын жеткізу таңдауы процедураларын тапсырыс берушімен анықтау;

■ ұсыныстарды талдау негізінде нақты жеткізушіні таңдау;

■ жеткізушімен шартты дайындау және орындау;

■ шартқа өзгерістер енгізу (қажет болған жағдайда).

Жеткізуші қызметін бақылау аудит және бірігіп бағалау үрдістерінде қарастырылатын әрекеттерге сәйкес жүзеге асырылады (1.3 бөлімді қараңыз).

Қабылдау үрдісінде қажетті тесттер дайындалады және орындалады. Шарт бойынша *жұмысты аяқтау* қабылдау шарттарын қанағаттандырған кезде жүзеге асырылады.

Жеткізу үрдісі (supply process) тапсырыс берушіні БӨ-мен немесе қызметпен жабдықтау кезінде жеткізушінің әрекеттері мен тапсырмаларын қамтиды. Бұл әрекеттерге төмендегілер жатады:

■ жеткізуге бастамашылық ету;

■ өтінімдік ұсыныстарға жауап дайындау;

■ келісімшартты дайындау;

■ жоспарлау;

■ орындау және бақылау;

■ тексеру және бағалау;

■ жеткізу және жұмысты аяқтау.

Жеткізуге бастамашылық етуге тапсырыс берушімен өтінімдік ұсыныстарды қарастыру жатады және шешімдер қабылдау қойылатын талаптармен немесе шарттармен расталады немесе өз нұсқаларын ұсынады.

Өтінімдік ұсыныстарға жауап дайындау жеткізуге бастамашылық ету нәтижесінде қабылданған шешімдерге сәйкес орындалады.

Шартты дайындау тапсырыс берушімен нақты жеткізушіні таңдағаннан кейін жүзеге асырылады.

Жоспарлау келісімшартқа отырғаннан кейін орындалады, олар төмендегідей тапсырмалардан тұрады:

■ өз күшімен немесе қосалқы мердігерді тарту арқылы жұмыстарды орындауға қатысты жеткізушінің шешім қабылдауы;

■ жобаның ұйымдастыру құрылымынан тұратын жобаны басқару жоспарын жеткізушінің дайындауы, жауапкершіліктерді

шектеу, дайындау орталарына және ресурстарына талаптар қою, қосалқы мердігерлерді басқару және т.б.

Қосалқы мердігер – бұл тапсырыс берушінің шарты бойынша жеткізуші орындауы тиіс жұмыстың бір бөлігін орындау бойынша жеткізушімен келісімшартқа отыратын корпорация, жеке тұлға немесе ұйым.

Орындау және бақылауға БӨ-ді, жүйені немесе қызметті жеткізу, дайындау немесе жетілдіру бойынша жауапкершіліктерді жеткізушінің өзіне алуына байланысты орындалатын тапсырмалар, осы үрдістерді бақылау жатады.

Тексеру және бағалау бірігіп бағалау және аудит үрдістерінде қарастырылған әрекеттерге сәйкес орындалады (1.3 бөлімді қараңыз).

Жеткізу және жұмысты аяқтау қабылдау және жұмысты аяқтау бойынша әрекеттерге бастамашылық ету үрдісіндегі ескертулерге сәйкес орындалады.

Дайындау үрдісі (development process) Бұл дайындаушының тапсырмалары мен әрекеттерінен қамтиды және келесі жұмыс бағыттарын қарастырады:

- БӨ және оның құраушыларын берілген талаптар бойынша құру, жобалық және пайдалану құжаттамаларын рәсімдеу;
- жұмыс қабілеттілікті және БӨ сапасын тексеруге қажетті материалдарды дайындау;
- Қызметкерді оқытуды ұйымдастыруға қажетті материалдарды дайындау және т.б.

Дайындау үрдісі 9-шы бөлімде толығырақ қарастырылады.

Пайдалану үрдісі (operation process) оператор тапсырмалары атқару барысын қамтиды - дайындалған БӨ немесе жүйені пайдаланумен айналысатын ұйым әрекеттері. Бұл әрекеттерге мыналар жатады:

- дайындау жұмысы;
- пайдалану тестілері;
- жүйені пайдалану;
- пайдаланушыларды қолдау.

Дайындау жұмысы операторлармен келесідей тапсырмаларды орындауды қамтиды:

- пайдалану үрдісінде орындалатын жұмыстарды жоспарлау және пайдалану стандарттарын орнату;
- шектеу процедураларын анықтау және пайдалану кезінде туындайтын мәселелерді шешу.

Пайдаланушылық тестілеу БӨ-нің әрбір кезекті нұсқасы үшін орындалады, одан кейін ол пайдалануға жіберіледі.

Жүйені пайдалану пайдаланушылық құжаттамаға сәйкес осы ортада жүзеге асырылады.

Пайдаланушыларды қолдау БӨ-ді пайдалану үрдісінде қателерді анықтау кезінде көмек және кеңес беруден тұрады.

Сүйемелдеу үрдісі (maintenance process) сүйемелдеуші ұйымның (сүйемелдеу қызметтері) тапсырмалары мен

әрекеттерінен тұрады.

Аталған үрдіс тиісті құжаттамада және пайда болған мәселелермен немесе БӨ-нің бейімделуін жаңартудағы қажеттіліктермен шақырылатын БӨ-нің өзгеруінде (түрленуінде) орындалады. IEEE-90 стандартына сәйкес, (IEEE — Institute of Electrical and Electronics Engineers — Электротехника және электроника бойынша инженерлер институты) сүйемелдеу ретінде БӨ-ге қателерді түзету, өнімділікті арттыру, жұмыстың өзгерген шарттарына бейімдеу шарттарында БӨ-ге өзгерістер енгізу жатады. Сүйемелдеу үрдісі толығырақ 11-ші бөлімде қарастырылады.

1.3. Бағдарламалық өнімнің өміршеңдік циклының қосымша (қолдаушы) үрдістері

Қосымша (қолдаушы) үрдістердің негізгі мақсатына сенімді, тапсырыс берушінің талаптарын шартпен белгіленген мерзімде толығымен қанағаттандыратын бағдарламалық өнімді құру жатады. Қосымша үрдістерге құжаттау үрдістері, конфигурацияны басқару, сапаны қамтамасыз ету, анықтау, аттестаттау, бірігіп бағалау, аудит, мәселелерді шешу жатады.

Құжаттау үрдісі (documentation process) БӨ-нің өміршеңдік циклында құрылған ақпаратты нысандандырылған сипаттауды қарастырады. Аталған үрдіс жобалауды, жоспарлауды, дайындауды, шығаруды, түзетуды, таратуды орындайтын әрекеттер жинағынан тұрады және басшылық, техникалық мамандар, жүйе пайдаланушылары сияқты мүдделі тұлғаларға қажетті құжаттарды ұсынып сүйемелдейді.

Құжаттау үрдісі келесі әрекеттерден тұрады:

- дайындау жұмысы;
- жобалау және құжаттаманы дайындау;
- құжаттаманы шығару;
- сүйемелдеу.

Дайындау жұмысы қажетті құжаттар тізімін анықтау және растау үшін, БӨ-нің өміршеңдік циклында орындалатын құжатты үрдістер үшін талап етіледі

Жобалау және дайындау БӨ жұмысының үрдісінде орындалады және оның өміршеңдік циклын бір уақытта орындаумен аяқталады.

Құжаттаманы шығару БӨ дайындау кезінде оның дайындалуы және ары қарай сүйемелденуі бойынша жүзеге асырылады.

Сүйемелдеу БӨ-нің өміршеңдік циклының үрдісінде құжаттаманы жаңарту және түзету бойынша әрекеттерден тұрады.

Конфигурацияны басқару үрдісі (configuration management process) БӨ-нің өміршеңдік циклындағы әкімшілік және техникалық процедураларды пайдаланудан тұрады және

келесідей әрекеттерді қамтиды:

- БӨ құраушыларының жағдайын анықтау;
- БӨ түрленуін басқару;
- БӨ құраушыларының және түрлену сұранымдарының жағдайы туралы есепті дайындау және сипаттау;
- БӨ құраушыларының жан-жақтылы үйлесімділігін және толықтығын қамтамасыз ету;
- БӨ жеткізуді және сақтауды басқару.

IEEE,0-90 стандартына сәйкес, *бағдарламалық өнім конфигурациясы* дегеніміз – жүзеге асырылған БӨ-де және техникалық құжаттамада бекітілген функционалды және физикалық сипаттамалар жиынтығы. Конфигурацияны басқару өміршеңдік циклдың барлық кезеңдерінде БӨ-ге өзгерістерді енгізуді басқаруға, оларды жүйелік есепке алуға, ұйымдастыруға мүмкіндік береді. БӨ конфигурациясын басқару бойынша жалпы қағидалар мен нұсқаулықтар ISO/IEC CD 12207-2: 1995 стандартында қарастырылады: «Information Technology — Software Life Cycle Processes. Part 2.
Management for Software» («Ақпараттық технологиялар — Бағдарламалардың өміршеңдік циклының үрдістері. 2.Бөлім Бағдарлама конфигурациясын басқару»).

Конфигурацияны басқару үрдісіне жатады:

- дайындау жұмыстары;
- конфигурацияны сәйкестендіру;
- конфигурацияны бақылау;
- конфигурация қалпының есебі;
- конфигурацияны бағалау;
- шығарылымды және жеткізуді басқару.

Дайындау жұмыстары конфигурацияны басқаруды жоспарлаумен түсіндіріледі.

Конфигурацияны сәйкестендіру БӨ құраушыларын және олардың нұсқаларын ажырату және сәйкестендіру арқылы ережелерді орнатумен түсіндіріледі. Одан басқа, әрбір құраушыға және нұсқасына құжаттаманың таңбаланатын кешені сәйкестендіріледі. Нәтижесінде БӨ құраушыларының нұсқасын таңдауға арналған база құрылады, ол бағдарламалық өнімнің әртүрлі нұсқаларын сәйкестендіретін символдардың шектелген және реттелген жүйесін пайдаланады.

Конфигурацияны бақылау БӨ түрленуін, орындау шығындарын және әрбір түрлену тиімділігі есебімен оларды үйлестірілген орындауды жүйелі бағалау үшін қолданылады. Сонымен қатар, БӨ құраушыларын дамыту және олардың қалпын бақылау, нақты өзгеретін құраушылар және кешенді құжаттама адекваттылығы қамтамасыз етіледі.

Конфигурация қалпының есебі БӨ құраушыларының күйін тіркеуден, БӨ құраушылар нұсқаларының қайтарылған және орындалған түрленулері туралы есептерді дайындаудан тұрады. Есеп жиынтығы жүйенің және оның құраушыларының

ағымдағы жағдайы туралы көріністен, сонымен қатар түрлендіру тарихын жүргізуден тұрады.

Конфигурацияны бағалау БӨ құраушыларының функционалды толық- тығын, сонымен қатар, олардың физикалық жағдайының ағымдағы техникалық сипаттамаға сәйкестігін бағалаумен түсіндіріледі.

Шығарылымды және жеткізуді басқару бағдарламалар мен құжат- тамалардың эталонды көшірмелерін дайындаудан, оларды сақтаудан және ұйымда қабылданған тәртіпке (үрдіс) сәйкес пайдаланушыларға жеткізуден тұрады.

Сапаны қамтамасыз ету үрдісі (quality assurance process) БӨ және оның өміршеңдік циклы үрдістерінің бекітілген талаптарға және жоспарға сәйкестігін кепілдендіруді қамтамасыз етеді. БӨ сапасы дегеніміз – БӨ-нің бекітілген талаптарды қамтамасыз ету қабілетін сипаттайтын қасиеттер жиынтығы.

Құрылған БӨ дұрыс бағалау үшін сапаны бағалау үрдісі БӨ-ді дайындаушы субъектілерге тәуелсіз орындалуы тиіс. Басқа тексеру, аттестаттау, бірігіп бағалау, аудит және мәселелерді шешу сияқты қосымша үрдістер нәтижелері қолданылуы мүмкін.

Сапаны қамтамасыз ету үрдісіне жатады:

- дайындау жұмыстары;
- өнім сапасын қамтамасыз ету;
- үрдіс сапасын қамтамасыз ету;
- жүйе сапасының басқа көрсеткіштерін қамтамасыз ету.

Дайындау жұмыстары басқа қосымша үрдістермен үйлестіруден, пайдаланылатын стандарттар, әдістер, процедуралар, құралдар есебімен сапаны қамтамасыз ету үрдісін жоспарлаудан тұрады.

Өнім сапасын қамтамасыз ету БӨ толық сәйкестігін, шартта қарастырылған тапсырыс беруші талаптарының құжаттамаларына сәйкестігін кепілдендіруден тұрады.

Үрдіс сапасын қамтамасыз ету БӨ өміршеңдік циклы үрдістерінің сәйкестігін, дайындау әдістерінің, дайындау және қызметкер квалификациясы құралдарының келісімшарттағы стандарттар мен үрдістерге сәйкестігін кепілдендіруден тұрады.

Жүйе сапасының басқа да көрсеткіштерін қамтамасыз ету келісімшарт талаптарына және ISO-9001 сапа стандарттарына сәйкес жүзеге асырылады.

Тексеру үрдісі (***verification process***) кейбір әрекеттер нәтижелері болып табылатын БӨ алдыңғы әрекеттерге тәуелді талаптар мен шарттарды толығымен қанағаттандырады.

Тексеру орындаушының өзімен немесе осы ұйымдағы басқа маманмен, сонымен қатар, үшінші ұйым маманымен жүргізілуі мүмкін.

Бұл жерде тексеру тәуелділіктерінің түрлі деңгейлері туралы айтуға болатын көптеген түрлендірмелер қолданылуы мүмкін. Егер тексеру үрдісі жеткізушіге, дайындаушыға, операторға

немесе сүйемелдеу қызметіне тәуелсіз ұйыммен жүргізілетін болса, онда ол тәуелсіз тексеру деп аталады.

Тексеру үрдісіне жататын барыстар:

- дайындау жұмыстары;
- тексеру үрдісінің өзі.

Дайындау жұмыстары басқа қосымша үрдістермен үйлестіруден, пайдаланылатын стандарттар, әдістер, процедуралар, құралдар есебімен тексеру үрдісін жоспарлаудан тұрады. Тексеру тиімділігін арттыру үшін ол онда қолданылатын үрдістермен бірігуі қажет (жеткізу, дайындау, пайдалану немесе сүйемелдеу).

Тексеру тар мағынада БӨ дайындығын ресми дәлелдеуді білдіреді. Аталған үрдіс талдаудан, бағалаудан және тестілеуден тұрады.

Тексеру үрдісінде мыналар тексеріледі:

- талаптардың жүйеге қайшылықсыздығы және пайдаланушы қажеттіліктерін ескеру деңгейі;
- жеткізушінің бекітілген талаптарды орындау мүмкіндіктері;
- БӨ өміршеңдік циклының таңдалған үрдістерінің келісімшарт талаптарына сәйкестігі;
- БӨ өміршеңдік циклы үрдістерінің, процедураларының,стандарттарының адекваттылығы;
- БӨ жобалық сипаттамаларының бекітілген талаптарға сәйкестігі;
- келген және шығыс деректерді жобалық сипаттамаларда сипаттау дұрыстығы, логика, интерфейс, оқиғалар кезектілігі;
- кодтың сипаттамалар мен талаптарға сәйкестігі;
- кодты тестілеу және түзету, оның кодтаудың қабылданған стандарттарға сәйкестігі;
- БӨ құраушыларын жүйеге біріктіру дұрыстығы;
- Құжаттама адекваттылығы, толықтығы, қайшылықсыздығы.

Аттестаттау үрдісі (validation process) құрылатын жүйеге немесе БӨ-ге қойылатын талаптар толықтығын, функционалды бағытталуын анықтаудан тұрады. Аттестаттау БӨ-ді тестілеу дұрыстығын бағалау және бекітуді білдіреді. Аттестаттау БӨ-нің сипаттамаларға, талаптарға және құжаттамаларға толық сәйкестігін кепілдендіруі керек, сонымен қатар пайдаланушымен сенімді және қауіпсіз қолданылу мүмкіндігін қарастырады. Аттестаттауды барлық мүмкін жағдайларда тестілеу арқылы және тәуелсіз мамандарды пайдалану арқылы орындау ұсынылады. Аттестаттау бағдарламалық өнімнің өміршеңдік циклының бастапқы кезеңдерінде жүргізілуі немесе бағдарламалық өнімді қабылдау бойынша жұмыс бөлігі ретінде орындалуы мүмін. Аттестаттау тексеру сияқты тәуелсіздіктің әртүрлі деңгейлерімен жүзеге асырылуы мүмкін.

Егер аттестаттау үрдісі жеткізушіге, дайындаушыға,

операторға немесе сүйемелдеу қызметіне тәуелсіз ұйыммен орындалатын болса, онда ол тәуелсіз аттестаттау деп аталады.

Аттестаттау үрдісіне жатады:

- дайындау жұмыстары;
- аттестаттаудың өзі.

Дайындау жұмыстары басқа қосымша үрдістермен үйлестіруден, пайдаланылатын стандарттар, әдістер, процедуралар, құралдар есебімен аттестаттау үрдісін жоспарлаудан тұрады.

Аттестаттау құрылған БӨ-ге немесе жүйеге қойылатын талаптардың функционалды бағытталуына толық сәйкестігін анықтауға мүмкіндік береді.

Бірігіп бағалау үрдісі (joint review process) аталған жұмыстарды орындау кезінде құрылатын жоба және БӨ бойынша жұмыстар жағдайын бағалау үшін қолданылады. Ол негізінен ресурстарды, қызметкерлерді, аппаратураны және жобаның құралдарын басқару және жобалауды бақылаумен түсіндіріледі.

Бағалау жобаны басқару деңгейінде, сонымен қатар техникалық орындау деңгейінде орындалады және келісімшарт мерзімі ішінде жүргізіледі. Аталған үрдіс шартқа қатысушы кез-келген екі тәсілмен орындалуы мүмкін, бір тарап екінші тарапты тексереді.

Бірігіп бағалау үрдісіне жатады:

- дайындау жұмысы;
- жобаны басқаруды бағалау;
- техникалық бағалау.

Дайындау жұмыстары басқа қосымша үрдістермен үйлестіруден, пайдаланылатын стандарттар, әдістер, процедуралар, құралдар есебімен бағалау үрдісін жоспарлаудан тұрады.

Жобаны басқаруды бағалау бағаланатын жоба бойынша жұмыстар орындау кезінде ағымдағы жағдайды анықтауға мүмкіндік береді.

Жобаны техникалық бағалау жобаны техникалық дайындау бойынша жұмыстар орындау кезінде ағымдағы жағдайды анықтауға мүмкіндік береді.

Аудит үрдісі (***audit process***) БӨ-ді құру бойынша жұмыстар орындау кезінде келісімшарт талаптарына, жоспарларына, талаптарына сәйкестігін анықтаудан тұрады. Аудит шартқа қатысушы кез-келген екі тәсілмен орындалуы мүмкін, бір тарап екінші тарапты тексереді.

Аудит нақты жұмыстар мен есептердің жоспарларға, талаптарға және шартқа сәйкестігін анықтау үшін жүргізіледі. Аудиторлар (тексерушілер) БӨ дайындаушыларына тікелей тәуелді болмауы керек. Олар жұмыс жағдайын, ресурстарды пайдалануды, құжаттамалардың сипаттамалар мен стандарттарға сәйкестігін, жүргізілетін тестілеу дұрыстығын бағалайды.

Аудит үрдісіне жататындар:

- дайындау жұмыстары;
- аудиттің өзі.

Дайындау жұмысы басқа қосымша үрдістермен үйлестіруден, пай- даланылатын стандарттар, әдістер, процедуралар, құралдар есебімен аудиторлық тексерулерді жүргізуді жоспарлаудан тұрады.

Аудит — БӨ-нің немесе жүргізілетін жұмыстардың бекітілген талаптарға, жоспарларға, келісімшарт талаптарына, шартқа сәйкестігін тәуелсіз бағалау мақсатында құзіретті органмен жүргізілетін тексеріс (ревизия).

Мәселелерді шешу үрдісі (problem resolution process) дайындау, пайдалану, сүйемелдеу кезінде анықталған мәселелерді шешуді және талдауды және пайда болу көзіне тәуелсіз басқа үрдістерді қарастырады (анықталған сәйкессіздіктерді қоса алғанда). Әрбір анықталған мәселе сәйкестендірілген, сипатталған, талданған және шешілген.

Мәселені шешу үрдісіне жататындар:

- дайындау жұмыстары;
- мәселелерді шешу үрдісінің өзі.

Дайындау жұмыстары пайда болған мәселелерді шешу, анықтау, талдау бойынша жұмыстар жоспарлаумен және басқа қосымша үрдістерді сүйемелдеумен түсіндіріледі.

Мәселелерді шешу БӨ-нің өміршеңдік циклының бүкіл мерзімінде жүргізіледі және түрлі мәселелерді анықтау бойынша әрекеттерден немесе оларды талдау, жою бойынша сәйкессіздіктерден тұрады.

1.4. Бағдарламалық өнімнің өміршеңдік циклының ұйымдастырушылық үрдістері

Ұйымдастыру үрдістерінің негізгі мақсатына сенімді, тапсырыс берушінің талаптарын шартпен белгіленген мерзімде толығымен қанағаттандыратын бағдарламалық өнімді дайындау үрдісін ұйымдастыру жатады. Ұйымдастыру үрдістеріне басқару, инфрақұрылымды құру, жетілдіру, игеру жатады.

Басқару үрдісі (***management process)*** өз үрдістерін басқарушы кез-келген тараппен орындалуы мүмкін әрекеттер мен тапсырмалардан тұрады. Аталған тарап (менеджер) өнімнің, жобаның шығуын басқаруға, өнімді алу, жеткізу, дайындау, пайдалану, сүйемелдеу сияқты тиісті үрдістер тапсырмаларына жауап береді.

Басқару үрдісіне жататын барыстар:

- басқару саласын анықтау және бастамашылық ету;
- жоспарлау;

- БӨ-ді құру бойынша жұмыстарды басқару және оларды орындауды бақылау;
- тексеру және бағалау;
- жұмысты аяқтау.

Басқару саласын анықтау және бастамашылық етуде менеджер басқаруға қажетті ресурстарды анықтауы қажет (персонал, құрылғы және технология), сонымен бірге, осы ресурстардың жеткілікті екеніне көз жеткізуі керек.

Жоспарлау келесі тапсырмаларды орындаудан тұрады: жұмыстарды орындау бойынша графиктер құру; қажетті ресурстарды бөлу; жауапкершілікті бөлу; нақты тапсырмалары бар тәуекелдерді бағалау; басқару инфрақұрылымын құру.

Жоспарлау тапсырмалары 6 бөлімде қарастырылған.

БӨ құру бойынша жұмыстарды басқару және оларды *орындауды бақылау* жоспарлау нәтижелеріне сәйкес жүзеге асырылады.

Жұмыстарды орындау кезінде міндетті түрде оларды орындалуын *тексеріп*, алынған нәтижелерді *бағалау* қажет. Тексеру және бағалау нәтижелері бойынша, қажет болған жағдайда түзетулер енгізілуі мүмкін.

Жұмысты аяқтау алдын-ала дайындаған үрдістерге сәйкес, тапсырыс берушінің жеткізушіден алған барлық міндеттерді орындағаннан кейін болады.

Инфрақұрылымды құру үрдісі (infrastructure process) БӨ-ді сүйемелдеу, пайдалану, дайындау үшін қолданылатын технологиялар, стандарттар мен құралдар, аппараттық және бағдарламалық құралдарды қолдау (сүйемелдеу) және таңдауды қамтиды. Инфрақұрылым тиісті үрдістерге қойылатын талаптар өзгерісіне сәйкес сүйемелденуі және түрленуі қажет. Инфрақұрылым, өз кезегінде конфигурацияны басқару объектілерінің бірі болып табылады.

Инфрақұрылымды құру үрдісіне мыналар жатады:

- дайындау жұмыстары;
- инфрақұрылымды құру;
- инфрақұрылымды сүйемелдеу.

Дайындау жұмыстары басқа ұйымдастыру үрдістерімен үйлестіру және аппараттық, бағдарламалық құралдар, стандарттар, технологиялар, таңдалған технологиялар есебімен инфрақұрылым құру бойынша жұмыстарды жоспарлаумен түсіндіріледі.

Инфрақұрылымды құру таңдалған тұжырымға және БӨ құру бойынша жұмыстарды орындауға арналған инфрақұрылымды құру жоспарына сәйкес дайындау бойынша барлық әрекеттерден тұрады.

Инфрақұрылымды сүйемелдеу БӨ-ді сүйемелдеу қажеттілігінен және өнімді өзгертілген талаптарға сәйкес түрлендіруден туындайды.

Жетілдіру үрдісі* (*improvement process) БӨ өміршеңдік циклы үрдістерін бақылау, бағалау, өлшеуді, жетілдіруді қарастырады. Аталған үрдіске жатады:

- үрдістерді құру;
- үрдістерді бағалау;
- БӨ өміршеңдік циклы үрдістерін жетілдіру.

Үрдістерді құру БӨ өміршеңдік циклының үрдістерін жетілдіру өміршеңдік цикл үрдістерін орындауды бақылау негізінде, сипаттамаларды өзгерту және алынған нәтижелерді бағалау өңделетін БӨ-нің сапасын жақсартуға және оның құрылу мерзімін қысқартуға мүмкіндік береді.

Үрдістерді бағалау БӨ дайындау оның күшті және әлсіз жақтарын анықтауға және алынған нәтижелер негізінде қажетті жақсартулар жүргізуге мүмкіндік береді.

БӨ өміршеңдік циклы үрдістерін жетілдіру барлық қатысушы мамандардың еңбек өнімділігін арттыруға бағытталады, ол қолданылатын технологияларды, басқару әдістерін, құралдарды таңдау, қызметкерді оқыту арқылы орындалады. Жетілдіру әрбір үрдістің артықшылықтары мен кемшіліктеріне негізделеді. Мұндай талдау көп жағдайда ұйымдағы жоба бойынша тарихи, техникалық, экономикалық және басқа ақпараттардың жиналуы әсер етеді.

Оқу үрдісі (training process) қызметкердің алғашқы оқытылуын және квалификациясының тұрақты көтерілуін қамтиды. Алу, жеткізу, дайындау, пайдалану және бағдарламалық өнімді айтарлықтай деңгейде сүйемелдеу қызметкердің білім деңгейіне және квалификациясына тәуелді болады. Мысалы, БӨ дайындаушылары бағдарламалық инженерияның әдістері мен құралдарына оқытудан өтуі керек. Оқу үрдісінің мазмұны жоба талаптарымен анықталады. Бұл үрдіс үшін қажетті ресурстар мен оқытудың техникалық құралдары жоспарлануы керек. Одан басқа, пайдаланушыларды оқу жоспарына сәйкес оқытуға қажетті әдістемелік материалдар дайындалып, ұсынылуы қажет.

Оқу үрдісіне төмендегілер жатады:

- дайындау жұмыстары;
- оқу материалдарын дайындау;
- оқу жоспарын орындау.

Дайындау жұмыстары басқа ұйымдық үрдістермен үйлестіруден және квалификацияны арттыру, оқыту жоспарын құру жұмыстарын жобалаудан тұрады.

Оқу материалдарын дайындау оқытудың ажырамас бөлігі болып табылады, себебі оның тиімділігі мен сапасын арттыруға мүмкіндік береді.

Оқу жоспарын орындау жоспар дайын болған уақытта үздіксіз жүзеге асырылуы тиіс.

1.5. Бағдарламалық өнімнің өміршеңдік циклы үрдістері арасындағы өзара байланыс

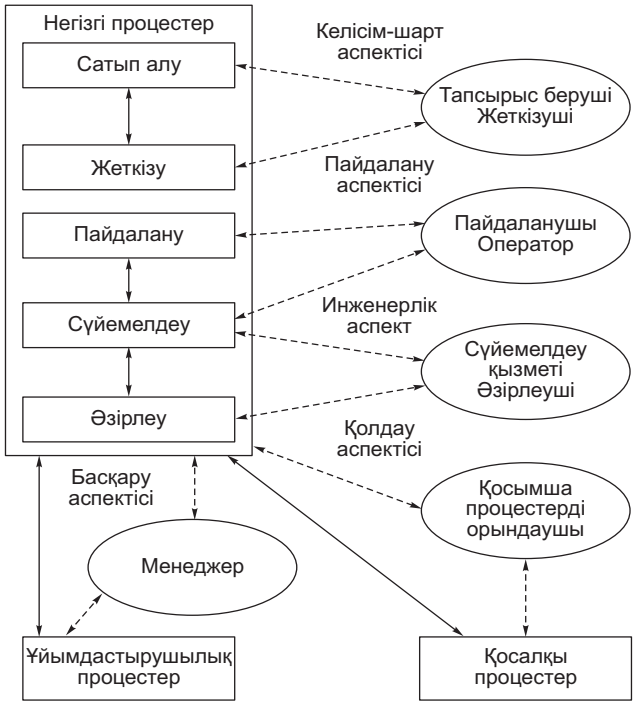
ISO/IEC 12207 стандартымен реттелетін БӨ өміршеңдік циклының үрдістері нақты жобаларда әртүрлі ұйымдармен қолданылуы мүмкін. Одан басқа, стандарт әртүрлі көзқарас тұрғысынан немесе әртүрлі аспектілерде үрдістер арасындағы өзара байланыс жиынтығынан тұрады (шартты, басқару, пайдалану, инженерлі қолдау). Бұл 1.1 суретте көрсетілген. Штрихты тілшелер нақты үрдістері бар үрдістердің қызмет етуші тұлғалары арасындағы байланысты көрсетеді (тапсырыс беруші, жеткізуші), ал тұтас тілшелер - үрдістер және үрдістер тобы арасындағы байланыс.

Шартты аспектіде тапсырыс беруші және жеткізуші келісімшарттық қатынастарға түседі және алу және жеткізу үрдістерін жүзеге асырады.

Басқару аспектісінде тапсырыс беруші, жеткізуші, дайындаушы, оператор, сүйемелдеу қызметі және БӨ өміршеңдік циклында қатысатын басқа тарапар өз үрдістерін орындауды басқарады. Менеджер ұйымдық және негізгі үрдістер арасындағы байланыстырушы буын болып табылады.

Пайдалану аспектісінде жүйені пайдаланушы оператор пайдаланушы- ларға тиісті қызметтерді ұсынады.

Инженерлі аспектіде дайындаушы немесе сүйемелдеу қызметі тиісті техникалық тапсырыстарды шешеді, БӨ-ді түрлендіріп, дайындайды.



Сурет 1.1. Бағдарламалық өнімнің өміршеңдік циклы үрдістері арасындағы байланыс

Қолдау аспектісінде қосымша үрдістерді орындайтын қызметтер барлық қызметкерлерге қажетті қызметтерді ұсынады. Қолдау аспектісі шеңберінде БӨ-нің сапасын басқару аспектісін бөліп қарастыруға болады.

Ұйымдастыру үрдістері БӨ-нің өміршеңдік циклының қалған үрдістерін тұрақты жетілдіру және жүзеге асыруға арналған базаны құру арқылы корпоративті деңгейде орындалады.

Үрдістер және оларды жүзеге асырушы ұйымдар (немесе тараптар) өзара функционалды байланысқан. Ішкі құрылым және ұйым мәртебесі реттелмейді. Бір ұйым әртүрлі рөлдерді атқаруы мүмкін (жеткізуші, дайындаушы және т.б.) және керісінше, бір рөлді бірнеше ұйым орындауы мүмкін. Стандартта сипатталған үрдістер арасындағы өзара байланыс статикалық сипатқа ие. Тараптармен жүзеге асырылатын үрдістер арасындағы маңызды динамикалық байланыстар нақты жобаларда бекітіледі.

Бақылау сұрақтары:

1. «Бағдарламалық өнімнің өміршеңдік циклы» түсінігін анықтауды тұжырымдау.

2. Бағдарламалық өнімнің өміршеңдік циклы қандай құжаттармен реттеледі?

3. Ресейдегі бағдарламалық өнімді құру алдымен қандай стандарттармен реттеледі?

4. Бағдарламалық өнімнің өміршеңдік циклы үрдістерін қандай топтарға бөлуге болады?

5. Қандай үрдістер әрбір топ құрамына кіреді?

6. Алу үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

7. Алу үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

8. Дайындау үрдісінде қандай әрекеттер мен тапсырмалар орындалады?

9. Пайдалану үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

10. Сүйемелдеу үрдісі дегеніміз не?

11. Құжаттау үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

12. Конфигурацияны басқару үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

13. Сапаны қамтамасыз ету үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

14. Тексеру үрдісінің аттестаттау үрдісінен айырмашылығы қандай?

15. Тексеру үрдісінде қандай шарттар тексеріледі?

16. Тәуелсіз аттестаттау үрдісі дегеніміз не?

17. Бірігіп бағалау үрдісінің аудит үрдісінен айырмашылығы қандай?

18. Мәселелерді шешу үрдісінде қандай тапсырмалар орындалады?

19. Басқару үрдісінің құрамына қандай әрекеттер кіреді және олардың мақсаты қандай?

20. Инфрақұрылымды құру үрдісінде қандай тапсырмалар орындалады?

21. Оқыту үрдісінің мақсаты қандай?

22. Бағдарламалық өнімнің өміршеңдік циклы үрдістері арасындағы өзара әрекеттестіктің негізгі аспектілерін атаңыз.

23. Сіздердің ойларыңыз бойынша, бағдарламалық өнімнің өміршеңдік циклының әртүрлі үрдістері арасындағы өзара байланыстар немен түсіндіріледі?

2 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ҚҰРУ БОЙЫНША ЖҰМЫСТЫҢ НЕГІЗГІ КЕЗЕҢДЕРІ

2.1. Негізгі кезеңдердің ұзақтығы

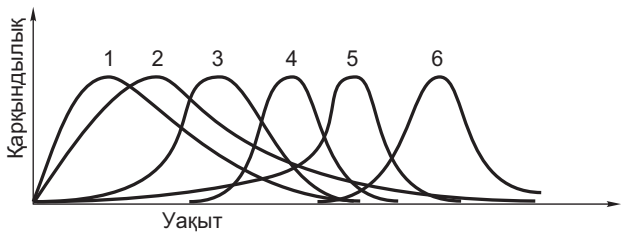
Құрылатын БӨ-нің сапасы мен жұмыс тиімділігін айтарлықтай арттыру бірыңғай ережелер мен технологияларды пайдалануға алып келеді. БӨ-ді құрудың бірыңғай үрдісі жоғары сапалы БӨ-ді құру бойынша барлық жұмыстарды орындау кезінде пайдаланылуы тиіс әдістемелер, әдістер, стандарттар мен құралдардан тұрады.

БӨ-ді құру кезінде жұмыстың алты негізгі кезеңін бөлуге болады:

1) Бағдарламалық жобаны жоспарлау;
2) Тапсырыс беруші талаптарын құру;
3) БӨ-ді жобалау;
4) БӨ-ді дайындау;
5) БӨ-ді тестілеу;
6) БӨ-ді сүйемелдеу.

БӨ өміршеңдік циклының әрбір кезеңіндегі сипатты ұзақтылық 2.1 суретте көрсетілген.

БӨ-ді құрудың алғашқы екі кезеңі бір уақытта басталады, бағдарламалық жобаны жоспарлау кезеңі (*1*) тапсырыс беруші талаптарын құру кезеңіне қарағанда (2) әрдайым ерте аяқталады. Осы екі кезеңде болашақ БӨ-ді құру бойынша жұмыстар мерзімі мен мазмұны анықталады. Бұл кезеңдердің ұзақтығы БӨ-нің жұмыс үрдісінде жұмыс жоспарына түзетулер, тіпті БӨ-ге талаптар қоюға алып келуі мүмкін.



Сурет 2.1. Бағдарламалық өнімнің өміршеңдік циклы кезеңдерінің ұзақтығы:
1 — 6 — кезек нөмірлері. Уақыт

Тестілеу кезеңі (5) *1* және *2.*кезеңдерімен бір уақытта басталады. Тесттің бұлай ерте басталуы алғашқы кезеңдердегі қателерді анықтауға мүмкіндік береді, бұл уақытты үнемдеуге және қателіктерді қалпына келтіруге көмектеседі. Ерте кезеңдерде БӨ-нің өзі емес, дайындалатын жобалық құжаттама тесттен өтеді.

2.2. Негізгі этаптар сипаттамасы

Бағдарламалық өнімді жобалау. Жоспарлау кезеңінде дайындау үрдісінде орындалуы тиіс барлық негізгі тапсырмалар анықталады, қаржылық, адами, техникалық және техникалық емес ресурстар, дайындалатын БӨ-нің көлемдері мен қиындықтары бағаланады, тестілеу әдістері және БӨ-ді қабылдау критерийлері, жұмыстың әдістері мен технологиялары анықталады, жұмысты орындаудың уақытша графиктері құрылады.

Тапсырыс беруші талаптарын құру. Осы кезең кезінде дайын- даушылар БӨ-нің талаптарын талдайды (ақпаратты жеткізу формасы, қажетті функциялар, қалаулы интерфейстер, тиісті шектеулер және т.б).

Аталған кезең талаптардың белгісіздігін жою үшін, болашақ БӨ-ге қатысты барлық бөлшектерді анықтау және оларды анық түсіну үшін, оны тестілеу үшін, БӨ-ге қойылатын талаптарға қатысты дайындаушылар мен тапсырыс беруші арасындағы өзара түсіністікті құру үшін қолданылады. Талаптар БӨ-нің талаптар сипаттамасына сәйкестігін анықтайтын «ия» немесе «жоқ» жауабын беретін тестті дайындаушы анық тест құра алатындай тестілеуден өтеді. Тестілеу үшін сипаттама анық, бір мәнді болуы және мүмкіндігі бойынша сандық сипаттарға ие болуы керек.

Бағдарламалық өнімді жобалау. Жобалау кезеңі дайындалатын БӨ-нің моделін өңдеу және анықтау үшін қолданылады. Мұндай модель БӨ-нің құрылымын, деректер мен интерфейстер, модульдердің ұйымдастырылуын және жүзеге асырудың кезекті кезеңіне қажетті сипатты анықтайды. Жобалау кезеңі жобалау кешендерінің жиынтығын ұсынуы мүмкін, олардың әрқайсы үшін қасиеттер жинағы және басқа кешендермен байланыстары анықталады. Жобалау үрдісі жоба жоспарында анықталған тәсілдер мен технологияларға сәйкес орындалуы керек. Жобалау екі бөлімнен тұруы мүмкін: жоғары деңгейлі және төмен деңгейлі (егжей-тегжейлі) жобалау.

Бағдарламалық өнімді дайындау. Бұл кезеңді орындау үрдісінде дайындаушылар жобалау кезеңінің нәтижелерін бағдарламада қолданылатын тілде бағдарлама кодына кодтау

стандарттарына сәйкес түрлендіреді. Дайындаушылар тестілеу бойынша тиісті шарттарды құру үшін БӨ тестілеу үшін инженермен өзара әрекеттеседі.

Одан басқа, дайындаушылар техникалық құжаттаманы құру бойынша жұмыстар жүргізеді және БӨ-ді біріктіруді орындау және жоспарлауды бастайды.

Бағдарламалық өнімді тестілеу. Тестілеу кезеңі нақты бір бастаудан тұрмайды, бірақ неғұрлым ерте басталса, соғұрлым дайындалатын БӨ тапсырыс беруші талаптарына сәйкес болады. Тестілеу бойынша барлық әрекеттер арнайы жұмысшы – тестілеушімен орындалуы керек. Ол тесттерді дайындайды, тестілеу процедурасын орындайды, тестілеу нәтижелері туралы есептер құрайды.

Тесттердің жалпы жинағы тестіленетін БӨ-ді максималды қамтуды қамтамасыз етуі қажет, яғни жеке модульдермен бірге, барлық өнім тестілеуден өтуі керек, сонымен қатар, жүйелік қасиеттері талаптарда бекітілген сипаттарға сәйкес тесттен өтеді.

Бағдарламалық өнімді сүйемелдеу. Сүйемелдеу кезеңінде негізгі БӨ-ге өзгерістер енгізуге негізгі көңіл бөлінеді. Бұл өзгерістер тапсырыс берушінің қосымша қалауларымен, БӨ-мен жұмыс істеу нәтижесінде пайда болатын қателіктерді түзеумен байланысты болуы мүмкін.

Егер өзгеріс қажетті болып танылатын болса, онда аталған өзгерісті енгізу бойынша жұмыстарды жоспарлау қажет, оларды құжаттандырып, БӨ-нің өзгерістері бойынша нәтижелерге шолу жүргізу қажет.

Бақылау сұрақтары

1. Бағдарламалық өнімді құру бойынша негізгі кезеңдерді олардың орындалу тәртібі бойынша атаңыз.
2. Бағдарламалық өнімді құру бойынша кезеңдер ұзақтығының сипаттамалы арақатынасын көрсетіңіз.
3. Дайындалатын бағдарламалық өнім қиындықтарын, көлемдерін, қажетті ресурстарын бағалау қандай кезеңде орындалады?
4. Тестті дайындауда сипаттамаға қандай талаптар қойылады?
5. Жобалау кезеңін орындауда қандай мақсат қойылады?
6. Жобалау кезеңдері қандай бөліктерге бөлінуі мүмкін?
7. Жобалау нәтижелерін бағдарламалық өнімге айналдыру қай кезеңде орындалады?
8. Пайдаланудағы бағдарламалық өнімге өзгерістер енгізу қажеттілігі неден пайда болады және бұл жұмыс қай кезеңде орындалады?

3 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ДАЙЫНДАУДЫҢ ӨМІШЕҢДІК ЦИКЛЫНЫҢ МОДЕЛЬДЕРІ

3.1. Бағдарламалық өнімді дайындаудың өміршеңдік циклының модельдері түсінігі. Бар модельдерді шолу.

БӨ-ді дайындаудың өміршеңдік циклының моделіне үрдістердің орындалу кезектілігін және өзара байланысын, БӨ дайындаудың өміршеңдік циклында орындалатын әрекеттер мен тапсырмаларды анықтайтын құрылым жатады. Өмрлік цикл моделі орындалатын жобаның сипатына және күрделілігіне, сонымен қатар, БӨ қызмет ететін шарттарға тәуелді болады.

ISO/IEC 12207 стандарты өміршеңдік циклдың нақты модельдерін және БӨ-ді дайындау әдістерін ұсынбайды. Стандарт ережелері өміршеңдік циклдың, БӨ дайындау технологиялары мен әдістерінің кез-келген моделі үшін ортақ болып табылады.

Стандарт БӨ-нің өміршеңдік циклы үрдістерінің құрылымын сипаттайды, бірақ осы үрдістерге жататын әрекеттер мен тапсырмаларды қалай орындау керек екенін анықтамайды.

Кез-келген нақты БӨ-нің өміршеңдік циклының моделі оны құру үрдісінің сипатын анықтайды, ол тиісті талаптарға сәйкес келетін БӨ-ді құруға қажетті өзара байланысқан және біріккен жұмыс кезеңдерін, уақыт бойынша реттелген жиынтықты ұсынады. БӨ-ді дайындау кезеңіне БӨ-ді құру үрдісі жатады, бұр үрдіс кейбір шектеулермен шектеледі және аталған кезең үшін бекітілген талаптармен анықталған нақты өнімді шығарумен аяқталады (БӨ модельдерін, бағдарламалық кешендерді, құжаттамаларды). БӨ-ді құру кезеңдері рационалды жоспарлау, бекітілген нәтижелермен аяталатын жұмыстарды ұйымдастыру түсінігі бойынша бөлінеді (2 бөлімді қараңыз).

БӨ-ді дайындаудың өміршеңдік циклының кең тараған модельдеріне келесілер жатады:

- каскадты модель немесе «сарқырама» (Waterfall model);
- V-образды модель (V-shaped model);
- прототиптеу (түпүлгілеу) моделі (Prototype model);
- бағдарламаларды тез дайындау моделі немесе RAD-модель

(RAD — Rapid Application Development model);

Кесте 3.1

Бағдарламалық өнімді дайындаудың өміршеңдік циклының модельдері

Атауы	Сипаттамасы
Каскадты модель	Түзу сызықты және пайдалануда қарапайым. Жұмысты тұрақты қатаң тексеруді қажет е Дайындалатын бағдарламалық жасақтамаға өзгерістер енгізілмейді.
V-образды модель	Пайдалануда қарапайым . Ерекше көңілге қонымды, тестілеуге және тестілеу фазасының нәтижелерін салыстыруға және жобалауға бөлінеді.
Прототиптеу моделі	Шекті талаптарды құрғанға дейін «тез» ішінара орындау құрылады. Жобаны орындау үрдісінде пайдаланушы мен дайындаушы арасында кері байланыс қамтамасыз етіледі. Пайдаланылатын талаптар толық емес.
Бағдарламаларды тез дайындау моделі	Жобалық топтар шағын (3....7 адам) және жоғары квалификацияланған мамандардан құралған. Дайындау циклының мерзімі қысқартылған (3 айға дейін) және өнімділігі жақсарылған. Кодты қайта пайдалану және дайындау үрдісін автоматтандыру.
Көп өтпелі модель	Жұмыс жүйесі тез құрылады. Дайындау үрдісіне өзгеріс енгізу мүмкіндігі азаяды. Ағымдық ішінара орындау кезінде ағымдағыдан жаңа нұсқаға өту мүмкін емес.
Спиральді модель	Каскадты модельді қамтиды. Фазаларды майда бөлшектерге бөледі. Жобалауды икемді орындауға мүмкіндік береді. Тәуекелдерді талдайды және басқарады. Пайдаланушылар БӨ-мен прототиптердің арқасында ерте кезеңде танысады.

- көп өтпелі модель (Incremental model);
- спиральді модель (Spiral model). Әрбір аталған модельдің қысқаша сипаттамалары 3.1 кестеде көрсетілген.

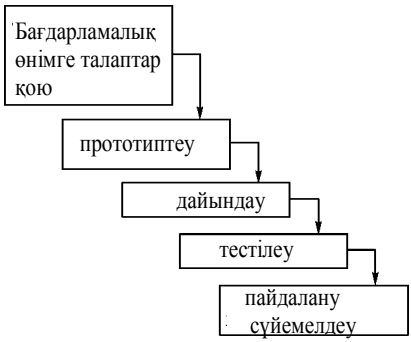
3.2 КАСКАДТЫ МОДЕЛЬ

1970 және 1980 жылдардағы біртекті ақпараттық жүйелерде қолданбалы БӨ біртұтас болды. БӨ-нің мұндай түрін дайындау үшін каскадты модель немесе «сарқырама» (waterfall) (сурет 3.1) қолданылған болатын.

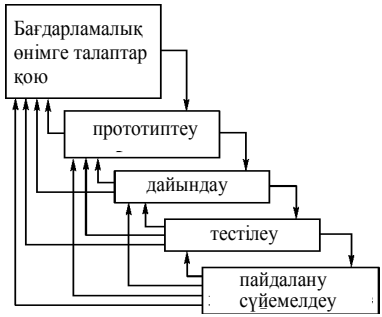
Каскадты тәсілдің қағидалы ерекшелігі: келесі кезеңге өту ағымдағы кезеңдегі жұмыс толығымен аяқталғаннан кейін жүзеге асырылады, өткізілген кезеңдерге қайту қарастырылмайды. Әрбір кезең келесі кезеңге бастапқы деректер ретінде қызмет ететін кейбір нәтижелерді алумен аяталады. Талаптарды құру кезеңінде анықталатын дайындалатын БӨ-ге қойылатын талаптар техникалық тапсырма ретінде қатаң құжатталады және жобаның дайындалу мерзімінде бекітіледі. Әрбір кезең құжаттаманың толық кешенін шығарумен аяқталады, бұл құжаттама дайындаушылардың басқа тобымен дайындаманың жалғасуы үшін жеткілікті болып табылады. Мұндай тәсілде дайындама сапасының талаптарына техникалық тапсырма сипаттамасын орындау дәлдігі жатады. Дайындаушылардың негізгі назары дайындалатын БӨ-нің техникалық сипаттамаларының оңтайлы мәндеріне қол жеткізуге бағытталады - өнімділігіне, толтырылатын жад көлеміне және басқаларға. Каскадты тәсілдің артықшылықтары:

- әрбір кезеңде жоба құжаттамасының аяқталған жинағы құрылады, ол толықтық пен келісушілікке жауап береді;
- логикалық кезектілікте орындалатын жұмыс кезеңдері барлық жұмыстарды аяқтау мерзімдерін және тиісті шығындарды жоспарлауға мүмкіндік береді.

Каскадты тәсіл ақпараттық жүйелерді құруда өзін жақсы көрсетті, дайындаушыларға техникалық түрде еркін орындауға мүмкіндік беру мақсатында дайындаудың басында барлық талаптарды дәл әрі толық қалыптастыруға болады.



Сурет 3.1. Каскадты модель



Сурет 3.2. Бағдарламалық өнімді дайындаудың нақты үрдісінің сызбасы

Бұл категорияға есептеу сипатындағы тапсырмалары көп күрделі жүйелер, нақты уақыт жүйелері және т.б жатады. Сонымен бірге, аталған тәсіл айтарлықтай кемшіліктерден тұрады: ең алдымен БӨ-ді дайындаудың нақты үрдісі ешқашан мұндай қатаң сызбада болмайды. Бұл үрдіс итерациялық сипатқа ие: кезекті кезең нәтижелері ерте кезеңде орындалған жобалық шешімдерде өзгерістер тудырады. Осылайша, өткен кезеңдерге қайту және бұрын қабылданған шешімдерді қарастыру қажетілігі туындайды.

Нәтижесінде дайындаудың нақты үрдісі 3.2 суретте көрсетілген.

3.2 суретте кескінделген сызбаны көбінесе аралық бақылауы бар модельге жатқызады, мұнда аралық түзетулер каскадты модельге қарағанда дайындаудың барлық кезеңін арттырғанымен, үлкен сенімділікті қамтамасыз етеді.

Каскадты тәсілдің негізгі кемшіліктеріне нәтижелерді алудың айтарлықтай кешігуі, пайдаланушылардың өзгерген қажеттіліктерін қанағаттадырмайтын жүйені құру тәуекелділігінің жоғары болуы жатады. Тәжірибе көрсеткендей, жобаның бастапқы кезеңінде барлық талаптарды нақты дәне толығымен болашақ жүйеге қалыптастыру мүмкін емес. Бұл келесі себептермен түсіндіріледі:

- пайдаланушылар барлық талаптарын бірден ұсына алмайды және олардың дайындау кезінде қалай өзгеретінін болжай алмайды;
- дайындау кезінде жүйе талаптарына әсер ететін сыртқы ортада өзгерістер орын алуы мүмкін.

Каскадты тәсіл шеңберінде БӨ-ге қойылатын талаптар құру кезінде техникалық тапсырмалар түрінде бекітіледі, ал алынған нәтижелерді пайдаланушылармен растау тек әрбір кезеңді аяқтағаннан кейін орындалады (сонымен бірге техникалық тапсырмада көрсетілген талаптарды қозғамаса, пайдаланушылардың ескертулері бойынша түзетулер енгізілуі мүмкін). Сонымен, пайдаланушылар жүйе жұмысы толығымен аяқталғаннан кейін, айтарлықтай ескертулер енгізулері мүмкін. Талаптарды анық емес жеткізуде немесе БӨ-ді құрудың ұзақ мерзімінде өзгерістер енгізуде пайдаланушылар олардың қажеттіліктерін қанағаттандырмайтын жүйені қолданады. Нәтижесінде осы жағдайды аңғаратын жаңа жоба бастауға тура келеді.

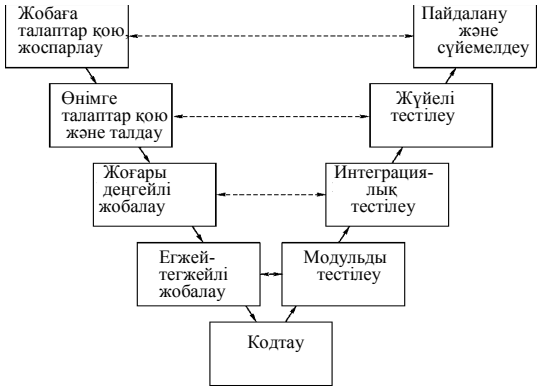
3.3. V-образды модель

Бұл модель (сурет 3.3) каскадты модельдің өзге түрі ретінде дайындалды, бұл жерде ерекше көңіл БӨ-ді тексеруге және аттестаттауға бөлінеді. Модель, көрсеткендей, өнімді тестілеу дайындаудың өміршеңдік циклының ерте кезеңінен бастап талқыланады, жобаланады, жоспарланады (3.3 суретте бұл үрдіс штрихты тілшелермен бейнеленген).

Каскадты модельден V-образды модель тізбекті құрылымды алды, ол арқылы әрбір тізбекті фаза алдыңғы фазаны сәтті аяқтағаннан кейін ғана басталады.

Аталған модель мәселені жүйелік тәсілмен шешуге бағытталады, оның шешімі төрт қадаммен анықталады: талдау, жобалау, дайындау және шолу. Талдау жүргізу кезінде жобаны жоспарлау, талаптарды құру жүзеге асырылады. Жобалау жоғары деңгейлі және егжей-тегжейлі (төменденгейлі) болып бөлінеді. Дайындау кодтаудан тұрады, ал шолу тестілеудің әртүрлі типтерінен тұрады.

Модельде кодтау мен тестілеуден бұрын болатын талдау фазалары мен жобалау фазалары арасындағы өзара байланыс жақсы көрсетіледі.



Сурет 3.3. V-образды модель

Штрихты тілшелер бұл фазаларды параллельді қарау керек екендігін көрсетеді.

Модель келесідей фазалардан тұрады:

жоба талаптарын құру және жоспарлау – жүйелік талаптар анықталады және жұмыс жоспары орындалады;

өнім талаптарын құру және оларды талдау – бағдарламалық өнім талаптарының толық сипаттамасы құрылады;

жоғары деңгейлі жобалау – БӨ-нің құрылымы, оның негізгі құраушылары арасындағы өзара байланыстар және функциялар анықталады;

егжей-тегжейлі жобалау – әрбір құраушы жұмысының алгоритмі анықталады;

кодтау — алгоритмдерді дайын бағдарламалық қамтамасыз етуге айналдыру;

модульді тестілеу — БӨ-нің әрбір модулін немесе құраушысын тексеру орындалады;

интеграциялық тестілеу – БӨ-нің интеграциясы немесе оны тестілеу жүзеге асырылады;

жүйелік тестілеу — БӨ-ді талап сипаттамасына сәйкес аппараттық ортаға орнатқаннан кейінгі оның қызмет етуі тексеріледі;

пайдалану және сүйемелдеу — БӨ-ді өндіріске жіберу. Бұл фазада БӨ-ге түзетулер енгізілуі және жаңартылуы мүмкін.

У-образды модель артықшылықтары мынадай:

- негізгі назар БӨ-ді тексеру мен аттестаттауға аударылады, оны дайындаудың ерте кезеңдерінен бастап барлық әрекеттер жоспарланады;
- БӨ-ді тексеру және аттестаттаумен бірге, барлық алынған сыртқы және ішкі деректерді тексеру жүргізілуі мүмкін;
- жұмысты орындау тәртібі оңай тексеріледі, себебі әрбір фазаның аяқталуы қорытынды нүкте болып табылады.

Артықшылықтардан басқа, модельдің кемшіліктері де бар:

- фазалар арасындағы итерация есепке алынбайды;
- өміршеңдік циклдың әртүрлі кезеңдерінде өзгерістер енгізуге болмайды;
- талаптарды тестілеу өте кеш орындалады, сондықтан өзгерістер енгізу жұмыс графигін орындауға әсер етеді.

Аталған модельді басты талабы жоғары сенімділік болып табылатын бағдарламалық өнімдерді дайындауда қолдану қажет.

3.4. Прототиптеу (түпүлгілеу) моделі

Прототиптеу моделі (сурет 3.4) БӨ-нің прототипін БӨ-нің талаптарын құру кезеңінде немесе кезеңіне дейін қалыптастыруға мүмкіндік береді. Әлеуетті пайдаланушылар осы прототиппен оның әлсіз және күшті жақтарын анықтайды, нәтижелерді БӨ дайындаушыларына жеткізеді. Осылайша, пайдаланушы мен дайындаушылар арасындағы кері байланыс қамтамасыз етіледі.

Сурет. 3.4. Прототиптеу моделі

Бұл БӨ талаптарының сипаттамасын түзету немесе өзгерту үшін қолданылады. Осындай жұмыс нәтижесінде өнім пайдаланушының нақты қажеттіліктерін көрсететін болады.

БӨ дайындаудың өміршеңдік циклы жобаны дайындаудан басталады (3.4 суретте жоспарлау кезеңіне эллипс ортасы сәйкес келеді), одан кейін тез талдау жүргізіледі, деректер базасы құрылады (әрине гер ол БӨ-де қолданылса), пайдаланушылық интерфейсі құрылып, тиісті функцияларды дайындау орындалады. Осындай жұмыс нәтижесінде БӨ талаптарының жеке сипаттамасынан тұратын құжат пайда болады. Аталған құжат ары қарай тез прототиптеудің итерациялық циклына негіз болады.

Прототиптеу нәтижесінде дайындаушы пайдаланушыларға дайын прототипті көрсетеді, ал пайдаланушылар оның қызмет етуін бағалайды. Одан кейін пайдаланушылар мен дайындаушылар бірігіп жоятын мәселелер анықталады. Бұл үрдіс пайдаланушылардың қажеттіліктері БӨ сәйкестік деңгейімен қанағаттандырылғанша жалғаса береді. Одан кейін прототип пайдаланушыларға жетілдіру ұсыныстарын алу мақсатында көрсетеді, олар жұмыс моделі қанағаттанарлық болғанға дейін тізбекті итерацияларға қосылады. Одан кейін пайдаланушылардан прототиптің функционалды мүмкіндіктерінің ресми мақұлдауын алады (бекіту) және оны дайын БӨ-ге айналдырады.

Прототип моделі төмендегідей артықшылықтарға ие:

- тапсырыс берушінің дайындалатын жүйемен өзара әрекеттестігі ерте кезеңде басталады;

■ тапсырыс берушінің реакциясына сәйкес, прототип талаптардағы дәлсіздіктер санының минимумына түсіріледі;

■ түсінбестіктердің, ақпараттың бұрмалануының немесе БӨ талаптарын анықтаудағы түсінбестіктердің пайда болу ықтималдығы азаяды, бұл сапалы БӨ-нің құрылуына алып келеді;

■ дайындау үрдісінде тапсырыс берушінің жаңа, тіпті күтілмейтін талаптарын есепке алуға болады;

■ прототип БӨ жүзеге асқан ресми сипаттамаға ие;

■ прототип жобалауды және дайындауды икемді орындауға мүмкіндік береді, оған дайындаудың өміршеңдік циклындағы барлық фазалардағы бірнеше итерациялар да жатады;

■ тапсырыс беруші әрдайым БӨ дайындау үрдісінде прогресті көреді;

■ тапсырыс беруші мен дайындаушы арасындағы түсінбестіктердің пайда болу мүмкіндіктері минимумға түсірілген;

■ түзетулер саны азаяды, бұл түзету құнын төмендетеді;

■ пайда болған мәселелер ерте кезеңдерде шешіледі, бұл оларды жою шығындарын қысқартады;

■ тапсырыс берушілер өнімнің өміршеңдік циклы кезеңінде дайындау үрдісіне қатысады және нәтижесінде жұмыс нәтижелеріне риза болады.

Аталған артықшылықтардан басқа, прототиптеу модельдері келесідей кемшіліктерге ие:

■ күрделі тапсырмаларды шешу келешекке қалдырылады;

■ тапсырыс беруші БӨ толық аяқталған нұсқасын емес, прототипті алуды таңдауы мүмкін;

■ прототиптеу қисынсыз созылып кетуі мүмкін;

■ жұмысты бастамас бұрын, қанша итерация орындау керек екендігі белгісіз болады.

Прототиптеу моделін келесідей жағдайларда пайдалану ұсынылады:

■ БӨ талаптары белгісіз;

■ талаптар тұрақсыз немесе сәтсіз қалыптасқан;

■ талаптарды анықтау қажет;

■ тұжырымдаманы тексеру қажет;

■ пайдалану интерфейсінің қажеттілігі талап етіледі;

■ жаңа дайындау орындалады (аналогтары жоқ);

■ дайындаушылар қандай шешімді таңдау керек екендігіне сенімсіз.

3.5. Бағдарламаны тез дайындау моделі (RAD-модель)

RAD –моделінде (сурет 3.5) шекті пайдаланушы негізгі рөлді атқарады. Дайындаушылармен тар өзара әрекеттестікте ол талаптар қалыптастыруға және оларды жұмыс прототиптерінде анықтауға қатысады. Осылайша, өміршеңдік циклдың басында шекті пайдаланушы жұмыстың көп бөлігін алады, бірақ осының нәтижесінде құрылатын жүйе тез қалыптасатын болады.

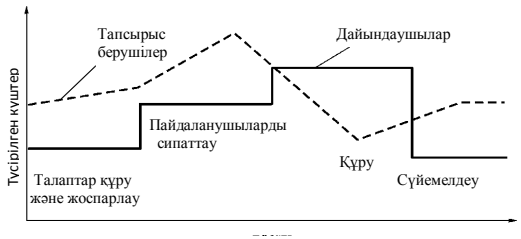
Дайындаудың дәстүрлі өміршеңдік циклында жұмыстың көп бөлігін бағдарламалау және тестілеу алады. Бағдарламалауды автоматтандыруда және RAD-модельдерінде қолданылатын кодты қайта пайдалануда жұмыстың көп бөлігін жоспарлау мен жобалау алады.

RAD моделінің қағидасымен түсіндірілетін 3.5 суретте дайындау үрдісінің кезеңдері және тапсырыс берушілердің қатысуы көрсетіледі (штрихтық сызық).

Модельге келесідей фазалар жатады: *талаптарды құру және жоспарлау* – талаптарды бірігіп жоспарлау әдісін қолдану арқылы жүзеге асырылады (БӨ құру бойынша жұмыстарды жоспарлау және БӨ талаптар құру бір уақытта орындалады). Ол құрылымдық талдауда және шешілетін тапсырмаларды талқылауда көрініс табады;

пайдаланушының сипаты - тапсырыс берушінің тікелей қатысуымен БӨ-ді жобалау;

құру — егжей-тегжейлі жобалау, кодтау және БӨ-ді тестілеу, сонымен қатар оны тапсырыс берушіге жеткізу;



Сурет 3.5. RAD-модель

сүйемелдеу— қабылдау зерттеулері, БӨ орнату және пайдаланушыларды оқыту.

Модель төмендегідей артықшылықтарға ие:

■ заманауи құралдарды пайдалану дайындау циклының уақытын қысқартуға мүмкіндік береді

■ жұмысқа тапсырыс берушіні тарту оның дайын БӨ-ге қанағаттанбау тәуекелін минимумға түсіреді;

■ қолда бар бағдарламалардың құраушылары қайта пайдаланылады;

Модельдің кемшіліктері:

■ егер тапсырыс берушілер әрдайым дайындау үрдісіне қатыса алмаса, онда бұл БӨ-ге кері әсер етуі мүмкін;

■ жұмысқа заманауи құралдарды пайдалана алатын жоғары квалификацияланған кадрлар қажет;

■ БӨ жұмысы аяқталмайды, себебі ол шырғалануы мүмкін, сондықтан уақытынды тоқтату қажет.

Қарастырылатын RAD-модельді БӨ-ді дайындау кезінде пайдалануға болады, БӨ талаптары белгілі болса, олар модельдеуге жақсы қолданылады, ал тапсырыс беруші дайындау үрдісіне тікелей қатысады.

3.5. Көп өтпелі модель

Көп өтпелі модель (сурет 3.6) — бұл әрбір кезекті итерацияға жаңа функционалды мүмкіндіктерді қосу арқылы немесе БӨ –нің тиімділігін арттыру арқылы БӨ-нің прототипін құру үрдісінің бірнеше итерациясы.

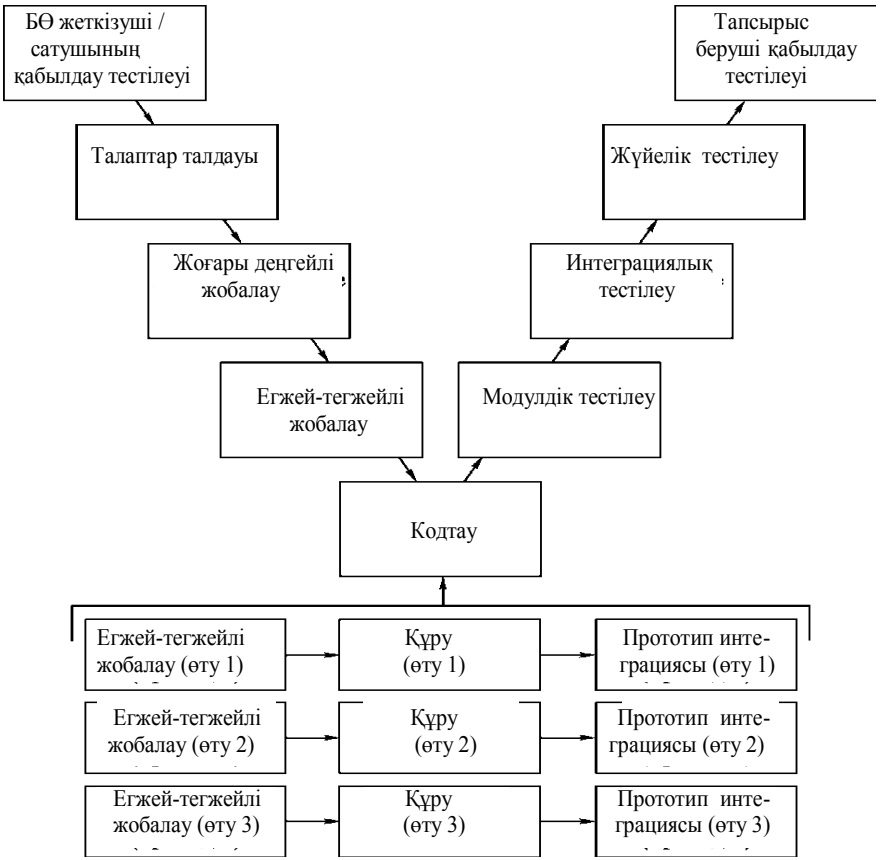
Дайындаудың өміршеңдік циклының ерте кезеңдерінде (жоспарлау, талаптарды талдау және жобаны дамыту) БӨ-ді құрастыру орындалады. Қажетті инкременттер саны және оларға тиісті функциялар анықталады. Әрбір инкремент өміршеңдік циклдың қалған фазалары арқылы өтеді (кодтау және тестілеу). Алдымен БӨ-нің негізін құрайтын базалық функцияларды орындау және тестілеу, құрастыру орындалады. Одан кейінгі итерациялар БӨ-нің функционалды мүмкіндіктерін жақсартуға бағытталады. Көп өтпелі модельдің артықшылықтары:

■ дайындау барысында БӨ негізгі функцияларын жүзеге асыруға және дайындауға қажетті құралдар талап етіледі;

■ әрбір инкременттен кейін функционалды өнім алынады;

■ сәтсіздік тәуекелі және талаптар өзгерісі азаяды;

■ ең кеш итерациялар үшін дайындаушылармен және пайдаланушылармен БӨ талаптарын түсіну жақсартылады;



Сурет 3.6. Көп өтпелі модель

■ функционалды мүмкіндіктер инкременттері тестілеуге оңай қолданылады.

Көп өтпелі модельдің кемшіліктері:

■ әрбір инкремент ішінде итерациялар қарастырылмайды;

■ толық функционалдылықты анықтау дайындаудың өмірлің циклының басында жүзеге асырылуы тиіс;

■ қиын тапсырмаларды шешудің созылу тенденциясы туындауы мүмкін;

■ БӨ-ді құрудың жалпы шығындары басқа модельдермен салыстырғанда азаймайды;

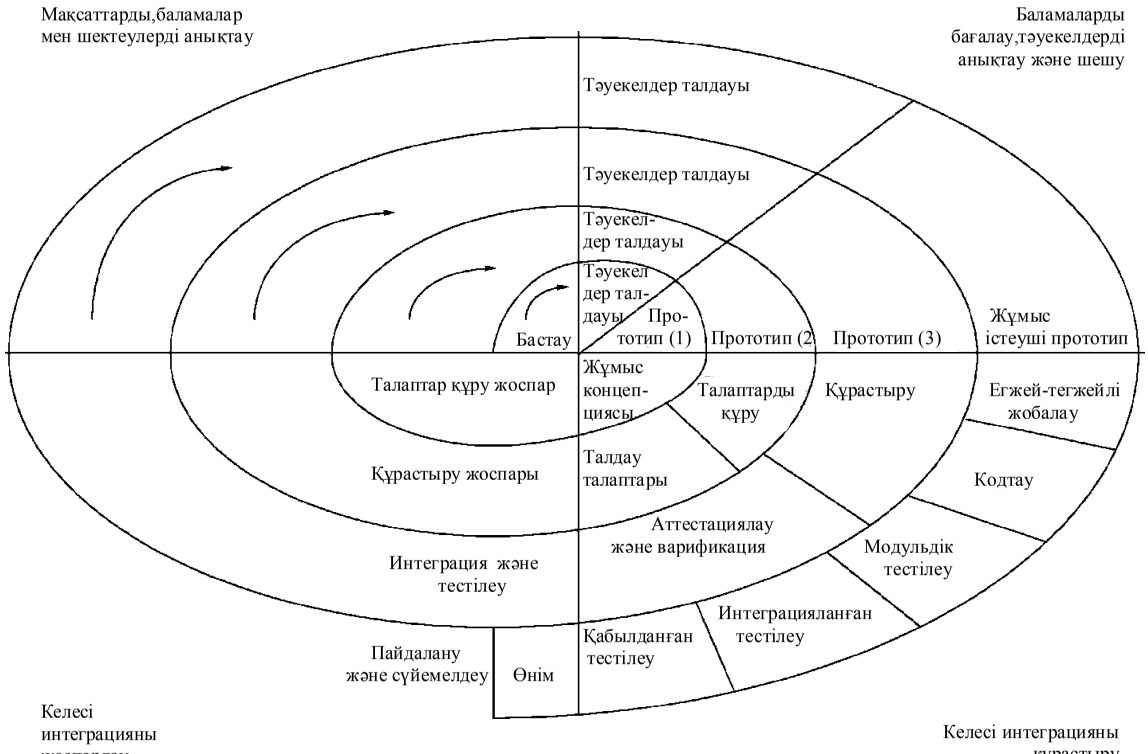
■ міндетті шарт ретінде жақсы жоспарлау және жобалаудың болуы қарастырылады.

Егер БӨ талаптары алдын-ала қалыптасатын болса көп өтпелі модель қолданылады, жобаны орындау үшін уақыттың үлкен мерзімі бөлінеді.

3.7. Спиральді модель

Көп өтпелі модельді пайдаланумен байланысты мәселелерді шешу үшін 1980 жылдардың ортасында өміршеңдік циклдың спиральді моделі ұсынылды (сурет 3.7). Оның қағидалы ерекшелігі қолданбалы БӨ бірден құрылмайды, каскадты тәсілдегі сияқты прототиптеу әдісін пайдалану арқылы бөлшектеп құрылады. Прототип ретінде жеке функцияларды орындайтын бағдарламалық кешен және дайындалатын БӨ-нің сыртқы интерфейстері қарастырылады. Прототиптерді құру бірнеше итерациялармен немесе спираль орамдарымен жүзеге асырылады. Әрбір итерация фрагмент құруға немесе БӨ нұсқасына сәйкес келеді, онда жобаның мақсаты мен сипаттары анықталады, алынған нәтижелер сапасы бағаланады және келесі итерация жұмысы жоспарланады. Әрбір итерацияда тағы бір итерацияның орындалу қажеттілігін анықтау мақсатында мерзімді және жоба құнын, толықтың деңгейін және жүйе талаптарын түсіну дәлдігін арттыру тәуекелін егжей-тегжейлі бағалау орындалады, сонымен бірге жобаны тоқтату жөнділігі қарастырылады. Спиральді модель пайдаланушылар мен дайындаушыларды бастапқы кезеңде жүйе талаптарын толық және нақты қалыптастырудан арылтады, себебі олар әрбір итерацияда анықталады. Осылайша, жоба бөлшектері тереңдетіледі және тізбекті нақтыланады және нәтижесінде жүзеге асырылатын негізгі нұсқа таңдалады.

Итерациялармен дайындау жүйені құрудың спиральді циклын объективті көрсетеді, келесі кезеңге жұмыстың толық аяқталуын күтпей-ақ өтуге мүмкіндік береді, себебі итеративті тәсілде жетіспей тұрған жұмысты келесі итерацияда орындауға болады. Мұндай дайындаманың басты мақсаты - пайдаланушыларға жұмысқа қабілетті өнімді тезірек көрсету, сонымен бірге, талаптарды анықтау және толықтыру үрдісін белсендіру. Егер жүйе талаптары толығымен анықталған болса, спиральді модель жобаның қорытынды кезеңдерінде каскадты тәсілді қолданады. Спиральді циклдың басты мәселесі – келесі кезеңге өту сәтін анықтау. Оны шешу үшін өміршеңдік циклдың әрбір кезеңіне уақыт шектеулерін енгізу қажет.



Сурет 3.7. Спиральді модель

Ауысу тіпті барлық жоспарланған жұмыс аяқталмаса да, жоспарға сәйкес жүзеге асырылады. Жоспар алдынғы жобаларда алынған статистикалық деректер және дайындаушының жеке тәжірибесі негізінде құрылады.

Спиральді модель төмендегідей артықшылықтарға ие:

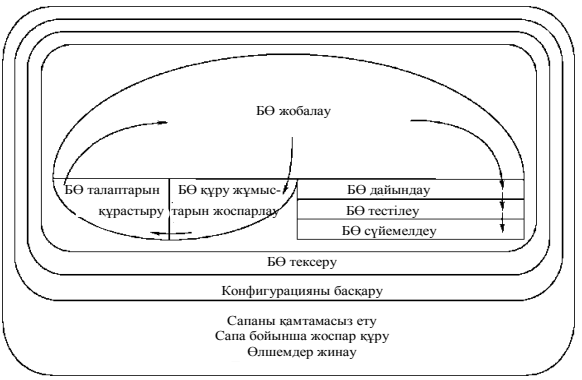
- тапсырыс беруші дайындалатын БӨ-ді дайындаманың ерте кезеңдерінде көре алады;
- тапсырыс берушілер БӨ-ді дайындауда белсенді қатысады;
- модельде каскадты және көп өтпелі модель артықшылықтары болады.

Спиральді модельдің кемшіліктері:

- күрделі құрылым;
- спираль шексіздікке дейін жалғасуы мүмкін, себебі тапсырыс берушінің әрбір жауаптық реакциясы жаңа циклды тудыруы мүмкін. Бағдарламалық өнімді дайындаудың өміршеңдік циклының моделі ретінде жақсартылған спиральді модель кеңінен таралды (сурет 3.8). Бұрын қарастырылған спиральді модельдерден айырмашылығы, бұл модель БӨ дайындаудың қорытынды кезеңдерінде каскадты тәсілді пайдаланады.

Егер келесі себептердің кемінде біреуі болса, спиральді модельді пайдаланған абзал:

- прототипті құрған жөн;
- ұйым модельді бейімдеуге қажетті дағдыларға ие;
- тәуекелділігі жоғары және орташа жобаларды орындау талап етіледі;
- тапсырыс берушілер өз қажеттіліктерінде сенімсіз;
- талаптар өте күрделі;
- жоба өте үлкен.



Сурет 3.8. Қосымша үрдістері көрсетілген жақсартылған спиральді модель

3.8. Қосымша (қолдаушы) үрдістер

3.8 суретте көрсетілгендей, БӨ-ді құру үрдісі қолдаушы үрдістерге негізделеді: БӨ-ді тексеру; конфигурацияны басқару; сапаны қамтамасыз ету.

Қолдаушы үрдістердің негізгі мақсаты – қысқа мерзімде тапсырысберушінің қажеттілігін толығымен қанағаттандыратын, сенімді бағдарламалық өнімді құру.

Бағдарламалық өнімді тексеру. БӨ-ді тексерудің басты мақсаты – дайындалатын өнімнің сапасын және дайындаушылардың өнімділігін арттыру. Тексерудің әртүрлі түрлері дайындаушыларға БӨ-ді құру бойынша жұмыс кезеңдерінде ақаулықтарды анықтауға көмектеседі. Одан басқа, тексеру бойынша іс-шараларды жүргізу компания ішіндегі жобалық топтардың өкілдері арасындағы серіктестікті жақсартуға ықпал етеді.

Тексерудің төмендегідей түрлерін бөліп қарастыруға болады. Жоғары басшылықпен, сапа тобымен, басқа жобалар өкілдерімен достық тексеру.

Жоғары басшылықпен тексеру компанияда бекітілген уақыт аралығынан кейін ұдайы жүргізілуі керек (мысалы, кварталды бір рет).

Осы тексеруді жүргізуде жобаның әрбір қызметкері жасаған жұмысы туралы, анықталған мәселелер, қызықты шешімдер туралы ақпараттарды баяндайды. Жоба басшысы алынған нәтижелер мен пайда болған мәселелерді белгілей отырып, жұмыстың орындалуы туралы айтып береді.

Сапа тобымен тексеру әдетте БӨ-ді құру бойынша әрбір кезең аяқталғаннан кейін орындалады. Сонымен қатар, есептік және жобалық құжаттамалардың болуы, осы құжаттамалардың стандарттар мен компания процедураларына сәйкестігі тексеріледі.

Басқа жоба өкілдерімен тексеру компанияның жалпы ұйымдық жиналысында жүзеге асырылады, ол әдетте әр апта жүргізіледі. Жиналысқа барлық жоба өкілдері қатысады және өз жұмыстарының орындалуы, алынған нәтижелер, пайда болған қиындықтар туралы баяндайды, озық тәжірибемен бөліседі немесе оны басқалардан қабылдайды. Қандай-да бір жобада мәселе пайда болса, оны бірігіп шешеді. Жаңа қолданбалы немесе аспаптық бағдарламалық жасақтаманы БӨ-ді дайындау үрдісіне енгізуде әдетте жаңалықты бейімдеуді жобалардың біреуінде орындайды (сынамалы).

Компанияның жалпы жиналысында осы жобаның өкілдері жаңалықты енгізу нәтижелерімен бөліседі және оны ортақ пайдалануға ұсыну, ұсынбауды қарастырады.

Басқа жоба өкілдерімен тексеру тексерілетін жоба өкілдерінің өтініштерімен орындалады. Тексерудің мұндай түрі пайда болған мәселелерді бірге шешуге және жоғары квалификацияланған мамандардың тәжірибелері мен білімдерін максималды пайдалануға мүмкіндік береді.

Достық тексеру бір жоба қызметкерлерінің арасында жүргізіледі. Кез-келген қызметкер басқа қызметкерге осы жобаны қарауға және өз жұмысының нәтижелерін тексеруге өтініш жасауы мүмкін. Бұл кейде қателерді табуға мүмкіндік береді.

Барлық тексерулер нәтижесі бойынша міндетті түрде анықталған мәселелер, олардың жою тәсілдері мен мерзімдері, сонымен бірге оларды жоюға жауапты тұлғалар көрсетілген есеп құрылады.

Тексеру қателерді жою бойынша барлық мәселелерді шешпесе де, оларды уақыты анықтауға мүмкіндік береді, ал жою көп күш пен құралдарды талап етпейді. Тексерудің негізгі мақсаттары:

- БӨ жұмысының ерте кезеңдеріндегі қателерді анықтау;
- жұмыс тәртібінің бекітілген стандарттар мен компания процедурасына сәйкестігін тексеру;
- БӨ құру бойынша жұмыс үрдісіндегі негізгі және аралық кезеңдерді орындауды бақылау;
- кейбір техникалық тапсырмалардың ресми аяқталуын тексеру;
- қызметкерлер жұмысындағы келісімділікті тексеру;
- жұмысты ары қарай жақсарту жолдарын анықтау.

Конфигурацияны басқару. Конфигурацияны басқару мәселесі 1.2 бөлімшеде толығырақ көрсетілген. Айтылғандарға тағы бірнеше ескертулер қосуға болады.

Конфигурацияны басқару өміршеңдік циклда БӨ-нің толықтығын қолдау үрдісі болып табылады.

Әрбір жобалық топ конфигурацияны басқаруға жауапты тұлғадан және компания конфигурациясын басқарудың жалпы жоспары құрамында конфигурацияны басқарудың өзіндік жобасынан тұрады.

Конфигурацияны басқарудың жобалық жоспары көрсетілген жоспарға сәйкес құрылады.

Сапаны қамтамасыз ету. Сапаны қамтамасыз ету мәселесі 1.2 бөлімде толығырақ көрсетілген. Айтылғандарға бірнеше ескертулерді қосуға болады.

БӨ сапасын қамтамасыз ету компанияда қабылданған барлық стандарттар мен процедуралардың барлық қызметкерлермен орындалуын тексерумен түсіндіріледі.

БӨ-нің сапасын қамтамасыз ету жұмыстары келесілерді

қамтамасыз етуі тиіс:

- тапсырыс беруші талаптарының орындалуын тексеру;
- БӨ сапасы бойынша тапсырыс берушімен өзара байланыс;
- БӨ сапасын арттыратын процедураларды дайындау және орындау;
- дайындаушылардың квалификацияларын арттыру.

Әрбір жобалық топ сапаны қамтамасыз ету бойынша жауапты тұлғадан және сапаны қамтамасыз ету бойынша жоспардан тұруы керек. Жекелеген жобалық топтар жоспарының негізінде барлық компанияның сапаны қамтамасыз ету жоспары құрылады, ол сапаны арттыру бойынша әрекеттерді және осы әрекеттерге жауап беретін жауапты тұлғаларды қалыптастырады.

БӨ құру бойынша жұмыстар орында үрдісінде дайындау кезеңіне тәуелсіз, әртүрлі статистикалық деректер (метрикалар) жиналады, олар жұмыс істеу тәртібін сандық бағалауға және жұмыс орындау кезіндегі қателіктерді анықтауға мүмкіндік береді. Жиналған метрикалар жоба жұмысын дәл жоспарлауға, тар жерлерді анықтауға және оларды жою бойынша шаралар қабылдауға көмектеседі.

Метрикалар БӨ-ді құру бойынша ұйымдық үрдістің сәйкестік деңгейінің, бағларламалық өнімді немесе БӨ өзін құру бойынша жеке жобаның сандық бағалауы болып табылады. Жұмыста өнім, жоба және үрдіс метрикалары қолданылады. Үрдіс метрикалары компанияның ұйымдық үрдісін жетілдіру үшін және орындауды қадағалау үшін, жобаның жұмысын жақсарту және қадалағау үшін, БӨ-нің сапасын жетілдіру үшін қолданылады (5 бөлімді қараңыз).

3.9. Бағдарламалық өнімді дайындаудың өміршеңдік циклының қолайлы моделін таңдау

БӨ-ді дайындаудың өміршеңдік циклының қолайлы моделін таңдауды келесі алгоритмді пайдалану арқылы орындау қажет.

1. Жобаның айрықша категорияларын талдау төмендегідей болады.

- БӨ пайдаланушыларына қойылатын талаптар;
- жоба қатысушыларының құрамы;
- БӨ болжамды пайдаланушылары;
- жоба түрі және болжамды тәуекелдер.

2. Жобаның әрбір категориясы үшін келтірілген тиісті кестелердегі сұрақтарға «ия» немесе «жоқ» сөздерімен жауап беру.

Кесте 3.2

БӨ талаптарын талдау негізінде өміршеңдік цикл моделін таңдау

Сұрақ	Өміршеңдік цикл моделі					
	Каскадты	V-образды	Прототипті	Спиральді	RAD	Инкрементті
Талаптар оңай анықталған немесе жақсы таралған ба?	Ия	Ия	Жоқ	Жоқ	Ия	Жоқ
Талаптар алдынала анықтала ал ма?	Ия	Ия	Жоқ	Жоқ	Ия	Ия
Талаптар жиі өзгере ме?	Жоқ	Жоқ	Ия	Ия	Жоқ	Жоқ
Анықтау мақсатында талапты көрсету қажет пе?	Жоқ	Жоқ	Ия	Ия	Ия	Жоқ
Мүмкіндіктерді көрсету үшін тұжырымдаманы тексеру қажет пе?	Жоқ	Жоқ	Ия	Ия	Ия	Жоқ
Талаптар жүйе күрделілігін көрсете ме?	Жоқ	Жоқ	Ия	Ия	Жоқ	Ия
Талаптар ерте кезеңдерде функционалды сипатқа ие бола ала ма?	Жоқ	Жоқ	Ия	Ия	Ия	Ия

3. Категорияларды маңыздылық деңгейі бойынша орналастыру, қажет болған жағдайда қолайлы модель таңдалатын жобаға қатысты сұрақтарды да қою.

4. Егер алынған көрсеткіштер бірдей немесе ұқсас болса, модельдерді салыстыру кезінде пайда болатын түсініспеушіліктерді шешуге арналған реттеу категорияларын пайдалану..

Кесте 3.3

Жоба қатысушыларын талдау негізінде өміршеңдік цикл моделін таңдау

Сұрақ	Өміршеңдік цикл моделі					
	Каскадты	V-образды	Прототипті	Спиральді	RAD	Инкрементті
Көптеген дайындаушылар үшін жобаның пән саласының мәселелері жаңа болып табылады ма?	Жоқ	Жоқ	Ия	Ия	Жоқ	Жоқ
Көптеген дайындаушылар үшін жобаның пән саласының технологиясы жаңа болып табылады ма?	Ия	Ия	Жоқ	Ия	Жоқ	Ия
Жобамен қолданылатын құралдар көптеген дайындаушылар үшін жаңа болып табылады ма?	Ия	Ия	Жоқ	Ия	Жоқ	Жоқ
Жоба қатысушыларының рөлдері өміршеңдік циклда өзгереді ме?	Жоқ	Жоқ	Ия	Ия	Жоқ	Ия
Жоба дайындаушылары оқытылады ма?	Жоқ	Ия	Жоқ	Жоқ	Ия	Ия
Дайындаушылар үшін құрылым икемділікке қарағанда маңызды болып табылады ма?	Ия	Ия	Жоқ	Жоқ	Жоқ	Ия
Жоба менеджері команда прогресін қатаң қадағалайды ма?	Ия	Ия	Жоқ	Ия	Жоқ	Ия
Ресурстарды бөлу жеңілдігі маңызды ма?	Ия	Ия	Жоқ	Жоқ	Ия	Ия
Команда тек құқықты шолулар мен тексерулерді, тапсырыс беруші шолуларын/кезеңдерді қабылдайды ма?	Ия	Ия	Ия	Ия	Жоқ	Ия

БӨ пайдаланушыларына қойылатын талаптарды талдау. 3.2 кестеде аталған жоба талаптарына қатысты сұрақтар келтірілген.

Жоба қатысушыларының құрамын талдау. Мүмкіндік бойынша, жобаны дайындаушылар командасының құрамына өміршеңдік цикл моделі таңдалғанға дейін қызметкерді алу қажет.

Кесте 3.4

Болжамды пайдаланушыларды талдау негізінде өміршеңдік цикл моделін таңдау

Сұрақ	Өміршеңдік цикл моделі					
	Каскадты	V-образды	Прототипті	Спиральді	RAD	Инкрементті
Өміршеңдік циклда пайдаланушылар шектеледі ме?	Ия	Ия	Жоқ	Ия	Жоқ	Ия
Пайдаланушылар жүйе анықтамасымен танысады ма?	Жоқ	Жоқ	Ия	Ия	Жоқ	Ия
Пайдаланушылар пән саласының мәселелерімен танысады ма?	Жоқ	Жоқ	Ия	Жоқ	Ия	Ия
Пайдаланушылар өміршеңдік циклдың барлық фазаларына тартылады ма?	Жоқ	Жоқ	Ия	Жоқ	Ия	Жоқ
Тапсырыс беруші жұмыстың орындалу тәртібін қадағалайды ма?	Жоқ	Жоқ	Ия	Ия	Жоқ	Жоқ

Кесте 3.5.

Болжамды тәуекелдер және жоба түрі негізінде өміршеңдік цикл моделін таңдау

Сұрақ	Өміршеңдік цикл моделі					
	Каскадты	V-образды	Прототипті	Спиральді	RAD	Инкрементті
Жоба өнімнің жаңа бағытын ұйым үшін анықтайды ма?	Жоқ	Жоқ	Ия	Ия	Жоқ	Ия
Жоба жүйелік интеграция түріне ие болады ма?	Жоқ	Ия	Ия	Ия	Ия	Ия
Жоба тиісті жүйенің кеңейтілуі болып табылады ма?	Жоқ	Ия	Жоқ	Жоқ	Ия	Ия
Жобаны қаржыландыру өмірік циклда тұрақты болады ма?	Ия	Ия	Ия	Жоқ	Ия	Жоқ
Ұйымда өнімді ұзақ уақыт пайдалану күтіледі ме?	Ия	Ия	Жоқ	Ия	Жоқ	Ия
Сенімділіктің жоғарғы деңгейі болу керек пе?	Жоқ	Ия	Жоқ	Ия	Жоқ	Ия
Күтілмеген әдістерді пайдалану арқылы жүйе сүйемелдеу кезеңінде өзгереді ме?	Жоқ	Жоқ	Ия	Ия	Жоқ	Ия
График шектеулі болып табылады ма?	Жоқ	Жоқ	Ия	Ия	Ия	Ия

Интерфейсті модульдер мөлдір болып келеді ме?	Ия	Ия	Жоқ	Жоқ	Жоқ	Ия
Қайта пайдаланылатын құраушылар қолжетімді ме?	Жоқ	Жоқ	Ия	Ия		Жоқ
Ресурстар жеткілікті ме (уақыт, ақша, құралдар, қызметкерлер)?	Жоқ	Жоқ	Ия	Ия	Жоқ	Жоқ

Жоба қатысушыларын талдау (3.3 кесте) модельді таңдауда маңызды рөл атқарады, себебі ол жобаны сәтті орындауға жауап береді және модельді таңдау үрдісінде көмектеседі

.

Болжамды пайдаланушыларды талдау. Жобаның бастапқы фазаларында болжамды пайдаланушылар туралы нақты ұғымды және жобаның қызмет ету уақытындағы дайындаушылар командасымен өзара байланысын алуға болады (кесте 3.4). М+ұндай ұғым тиісті модельді таңдауға көмектеседі, себебі кейбір модельдер жобаны дайындау және игеру үрдісінде пайдаланушылардың қатысуын талап етеді.

Болжамды тәуекелдерді және жоба түрін талдау. БӨ өміршеңдік циклының кейбір модельдерінде тәуекелді басқару қарастырылады, ал кейбір модельдерде тәуекелді басқару қарастырылмайды. Тәуекелді басқаруды қарастырмайтын модельді таңдау анықталған тәуекелдерді азайтуға бағытталған әрекеттер жоспарын құру және талдау керек емес дегенді білдірмейді. Мұндай модель әрекеттер жоспарын орындауға және талқылауға болатын шеңбердегі сызбаны қамтамасыз етеді (кесте 3.5).

Сәйкес модельді таңдау – бұл нақты жоба үрдісіндегі өміршеңдік циклдың моделін пайдаланудың бірінші кезеңі. Келесі кезең осы жобадағы қажеттіліктер мен мүмкіндіктерге сәйкес оны үйлестірумен түсіндіріледі. Бұл дегеніміз, таңдалған нақты фазалар мен әрекеттер жоба жетекшісіне жобаны таңдалған модельге үйлестіруге көмектесуі қажет.

SEI CMM модельге сәйкес, бұл есепке қандай-да бір нақты жазылулар жоқ. Өміршеңдік циклдар, олардың фазалары, сонымен қатар тиісті әрекеттерді циклдарды, фазаларды, әрекеттерді анықтаудың жіберу нүктесі ретінде пайдалануға болады, олар қазіргі уақыт қажет болып табылады. Үйлестіруді аяқтағаннан кейін, модель дайындаушылар командасымен

негізге алынады және жобаны орындауда қолданылады.

Егер жобаны орындау кезінде командаға басқа модель әрекетті болатын еді деген ойды тудыратын қандай-да бір өзгерістер орын алса не болады? Жобаны орындау үрдісінде модельді өзгертуге болады ма? Бұл сұраққа жауап әрдайым оң болады, бірақ жобаның мұндай өзгерістерінің болжамды салдарын міндетті түрде есепке алу қажет. Жоба қажеттіліктеріне тиісті деңгейде сәйкес келмейтін модельді пайдаланғанша, оны өзгерткен дұрыс.

3.10. Бағдарламалық өнімді құрудың өнеркәсіптік технологиялары

БӨ құрудың өнеркәсіптік технологиялары әртүрлі бағдарламалық қамтамасыз етуді құрудың үлкен тәжірибесін жинаған фирмалармен дайындалуы мүмкін. Бұл технологиялар бұрын қарастырылған модельдердің буданы болып табылатын өміршеңдік цикл модельдерінің негізіндегі қағидалар, әдістер, пайдаланылатын үрдістер мен операциялар сипаттарымен ұсынылады. Әдетте мұндай технологиялар CASE-құралдарын жинаумен түсіндіріледі және өнімнің өміршеңдік циклының барлық кезеңдерін қамтиды және тәжірибелік тапсырмаларды шешуде сәтті пайдаланылады.

Үлкен қызығушылықты өміршеңдік циклдың келесі модельдері ұсынады:

Microsoft Solution Framework (MSF) — тапсырмалардың кең аясын шешуге арналған Microsoft фирмасының дайындамасы. Технология масштабталған, яғни кез-келген қиындықтағы мәселені ұжыммен шешуге бағытталады;

Rational Process (RUP) — ұзақ уақыт бойы CASE-құралдарын құрумен айналысқан Rational фирмасының дайындамасы, олар өміршеңдік циклдың әртүрлі кезеңдерінде қолданылады: талдаудан тестілеуге және құжаттандыруға дейін.

MSF, RUP – әмбебап, масштабты және нақты шарттарда бапталады;

Extreme Programming (XP) — қатаң шектелген уақыт шарттарында шағын тапсырмаларды кәсіби дайындаушылардың шағын ұжымымен шешуге арналған, соңғы кездері белсенді дамыған технология.

Осы технологиялардың әрқайсы БӨ өміршеңдік циклы моделін ұйымдастырудың өзіндік ерекшеліктеріне ие.

Microsoft Solution Framework моделі. MSF технологияларының негізгі ерекшеліктерінің біріне оның жоғарыдағы талаптарды қанағаттандыратын БӨ құруға ғана емес, сонымен бірге тапсырыс беруші алдындағы мәселелерді

шешуге, талдауға, іздеуге бағытталады.

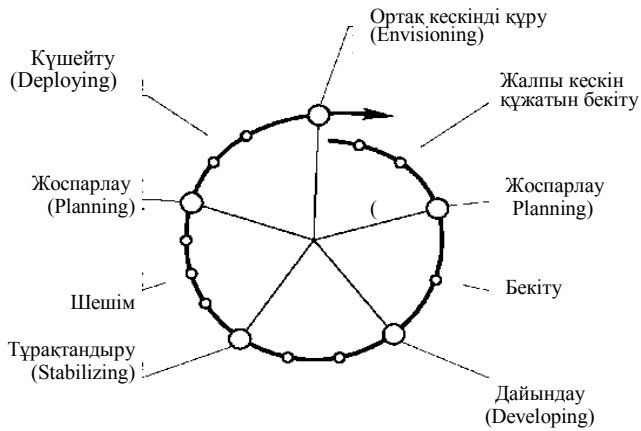
Оның айырмашылығы - тапсырыс берушінің қоятын талаптары терең мәселелер мен дәл келмейтін көрінісі болып табылады. Оның толық болмауы талаптардың дайындау үрдісінде өзгеруі - мәселелерді түсінбеу салдары. Сондықтан, MSF технологиясында көп көңіл тапсырыс беруші мәселесін талдауға және осы мәселелерді шешу және іздеуге арналған жүйе нұсқаларын дайындауға бөлінеді.

MSF өміршеңдік циклының моделі каскадты және спиральді модельдің кейбір буданы болып табылады, каскадты модельдің басқару қарапайымдылығын спиральді модельдің икемділігімен үйлестіреді. MSF өміршеңдік циклының модель сызбасы 3.9 суретте көрсетілген.

MSF өміршеңдік циклының моделі «утамырға» бағытталады (milestones) – бұл қандай-да бір нәтижеге қол жеткізуді сипаттайтын жобаның басты нүктелері. Бұл нәтиже келесі сұраққа жауап ретінде бағаланады және талданады: «Осы қадамда қойылған мақсаттарға біз жеттік пе?». Модельде негізгі фазаның ішкі кезеңдерін көрсететін негізгі және аралық утамырлардың болуын қарастырады (модельдің негізгі фазаларының аяқталуы).

MSF негізгі фазалары:

бағдарламаның жалпы кескінін құру (Envisioning). Бұл кезеңде келесідей негізгі тапсырмалар шешіледі: ағымдағы жағдайды бағалау; команда құрамын, жоба құрылымын, бизнес-мақсаттарды, пайдаланушылардың талаптары мен бейімдерін анықтау; тәуекелді бағалау тұжырымын дайындау.



Сурет 3.9. MSF өміршеңдік циклы моделінің сызбасы

Екі аралық утамыр орнатылады: «Команда ұйытқысы ұйымдастырылған» және «Шешімнің жалпы кескіні құрылған».

Ол жобаны жоспарлау және жобалаудан тұрады. Талаптарды талдау негізінде жоба және негізгі сәулет шешімдері, жүйенің функционалды сипаттары, жоспарлар мен күнтізбелі графиктер, дайындау орталары, тестілеу және сынамалы пайдалану дайындалады. Бұл кезең үш деңгейден тұрады: тұжырымдамалық, логикалық және физикалық жобалау. Тұжырымдамалық жобалау деңгейінде тапсырма пайдаланушылық және бизнес талаптарынан қарастырылады және жүйені пайдалану сценарийін жинауды анықтаумен аяқталады. Логикалық жобалауда тапсырма жобалық команда тарапынан қарастырылады, оның шешімі сервис таңдау түрінде ұсынылады. Және физикалық жобалау деңгейінде тапсырма бағдарламашылар тарапынан қарастырылады, пайдаланылатын технологиялар мен интерфейстер анықталады;

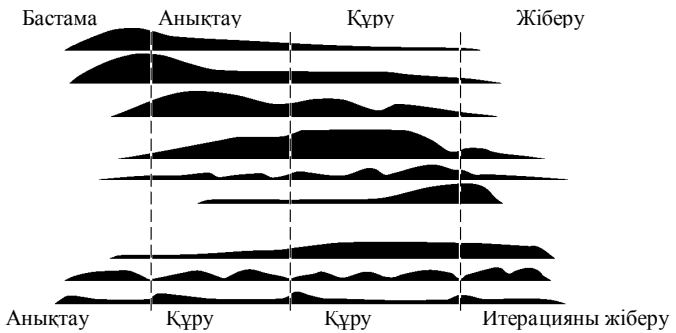
дайындау (Developing). Код және кезекті прототип құжаттамасы түріндегі мәселені шешу нұсқасы құрылады, оған сипаттама және тестілеу сценарийі жатады. Кезеңнің негізгі утамыры – «Жоба әрекеті саласын қорытынды бекіту». Өнім сырқы тестілеуге және тұрақтандыруға дайын. Одан басқа, тапсырыс берушілер, пайдаланушылар, қолдау және сүйемелдеу қызметінің қызметкерлері, сонымен қатар жобаның негізгі қатысушылары өнімді алдын-ала бағалап, жеткізуге дейін жойылуы тиіс барлық кемшіліктерді көрсетуі керек.

Тұрақтандыру (Stabilizing). Өнімнің қорытынды нұсқасын шығаруға дайындау, оны сапаның белгіленген деңгейіне жеткізу. Мұнда тестілеу бойынша жұмыстар кешені орындалады (қателерді табу және жою), өнімді күшейту сценарийі тексеріледі. Шешім жеткілікті деңгейде тұрақты болғанда, оны тестілік ортада сынамалы пайдалану орындалады және пайдаланушылар тартылып, жұмыстың нақты сценарийі қолданылады;

Күшейту (Deploying). Тиісті қоршау құралдарының шешімін қалып- тастыру орнатылады, өнеркәсіптік шарттардағы оны тұрақтандыру және жобаны сүйемелдеу тобына өткізу жүргізіледі. Одан басқа, жоба тапсырыс берушінің қанағаттану деңгейіне талданады.

Rational Unified Process моделі. RUP өміршеңдік цикл моделі күрделі, каскадты модель элементтері бар итеративті-инкрементті модельмен егжей-тегжейлі дайындалған. RUP моделінде 4 негізгі фаза, қызметтің (үрдістің) 9 түрі бөлінеді. Одан басқа, модельде жобаны сәтті орындауға басшылық ететін және қолданылатын тәжірибелер қатары сипатталады. Бұл модель UML тілінің көмегімен құрылатын өнімді кезеңді модельдеуге бағытталады. RUP өміршеңдік цикл моделі 3.10 суретте көрсетілген. RUP негізгі фазалары:

Жобаның басталу фазасы (Inception). Жобаның негізгі мақсаттары, жоба бюджеті, оны орындаудың негізгі құралдары-технологиялар, құралдар, негізгі қызметкер анықталады.



Сурет 3.10. RUP өміршеңдік циклы моделінің сызбасы

Бизнес-модельдеу Талаптарды анықтау Талдау және жобалау

Өткізу
Тестілеу
Күшейту

Конфигурация мен өзгерістерді басқару Жобаны басқару Ортаны басқару

Жобаның алдын-ала дайындалған жоспары анықталады. Бұл фазаның негізгі мақсаты — жоба тапсырмасына қатысты барлық мүдделі тұлғалар арасындағы келісімге қол жеткізу;

Анықтау фазасы (*Elaboration*). Бұл фазаның негізгі мақсаты – негізгі базалар, талаптар негізінде жүйе тапсырмаларын орындауға және оны ары қарай жүйені дайындау үшін пайдалануға мүмкіндік беретін өнімнің тұрақты базалық құрылысын дайындау;

Құру фазасы (Construction). Бұл фазаның негізгі мақсаты — талаптарды егжей-тегжейлі түсіндіру және осы талаптарды қанағаттандыратын жүйені бұрын жобаланған құрылыс негізінде дайындау;

Жеткізу фазасы (Transition). Фаза мақсаты — жүйені шекті пайда- ланушыларға толығымен қолжетімді ету. Бұл жерде жүйені оның жұмыс ортасында шекті күшейту, шағын бөлшектерді пайдаланушылар қажет- тіліктеріне үйлестіру орындалады.

Әрбір фаза шеңберінде саны орындалатын жоба күрделілігі-

мен анықталатын бірнеше итерациялар жүргізу орындалуы мүмкін.

RUP негізгі үрдістері бес жұмыс және төрт қолдау үрдістеріне бөлінеді. Жұмыс үрдістеріне жатады:

Пән саласын модельдеу (бизнес-модельдеу, Business Modeling). Бұл үрдістің мақсаты —жүйе жұмыс істеуі тиіс бизнес-мәтінді түсіну, болжамды мәселелерді түсіну, оларды шешу нұсқаларын және ұйым бизнесінің болжамды салдарын бағалау.

Талаптарды анықтау (Requirements). Бұл үрдістің мақсаты — жүйенің не істеу керек екенін түсіну, жүйе шектерін және жобаны жоспарлау негізін, тиісті ресурстарды бағалауды анық*Талдау және жобалау (Analysis and Design)*. Бұл үрдістің мақсаты – негізгі талаптар негізінде жүйе құрылысын дайындау, өнімді түрлі көзқарастан сипаттайтын UML диаграммасы түрінде ұсынылған жоба моделін құру.

Өткізу(Implementation). Бұл үрдістің мақсаты — бастапқы кодты, жүйе кешенін дайындау, тестілеу және құраушыларды біріктіру.

Тестілеу (Test). Бұл үрдістің мақсаты — өнімнің кемшіліктерін және сапасын жалпы бағалау. Бастапқы талаптарға сәйкестік деңгейін бағалау.

Қолдаушы үрдістер:

Күшейту (Deployment). Үрдіс мақсаты — жүйені оның жұмыс щеңберінде күшейту және оның жұмыс қабілеттілігін бағалау.

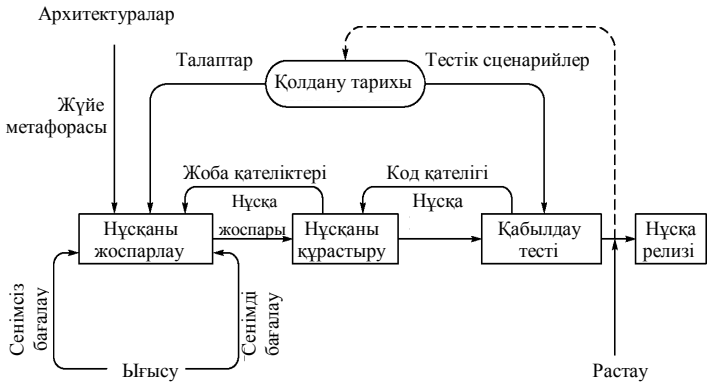
Конфигурациялар мен өзгерістерді басқару (Configuration and Change Management). Үрдіс мақсаты – сақталатын элементтерді және келісімді конфигурацияны құру ережелерін анықтау, жүйенің ағымдағы қалпының толықтығын сақтау, енгізілетін өзгертулерді тексеру.

Жобаны басқару (Project Management). Үрдіс мақсаты — жоспарлау, қызметкерлерді басқару, басқа мүдделі тұлғалармен байланысты қамтамасыз ету, тәуекелдерді басқару, жобаның ағымдағы жағдайын қадағалау.

Жоба ортасын басқару (Environment). Үрдіс мақсаты — үрдісті нақты жобаға баптау, жобада қолданылатын технологиялар мен құралдарды таңдау және ауыстыру.

Extreme Programming моделі. XP төтенше бағдарламалау моделі итерациялық-инкрементті типтегі өміршеңдік цикл моделі болып табылады, ол тапсырыс берушінің кезекті талабын қамтамасыз ететін өнім прототипін тез құруға және түрлендіруге қолданылады. Бұл модельдің ерекшеліктері 3.11 суретте көрсетілген.

XP моделінің негізгі фазалары:



Сурет 3.11. XP өміршеңдік циклы моделінің сызбасы

«*Құрылымды шығару*» — өнім көрінісі құрылатын жобаның бастапқы фазасы, құрылым және пайдаланылатын технологиялар бойынша негізгі шешімдер қабылданады. Бұл фазаның нәтижесіне жүйенің метафорасы (metaphor) жатады, ол қарапайым және түсінікті командада жүйе жұмысының негізгі механизмін сипаттайды;

Пайдалану тарихы (User Story) — жекелеген фунцияларды орындау сценарийі ретінде арнайы карточкаларда жазылатын талаптар жинағының фазасы. User Story кезекті нұсқаны жоспарлау және қабылдау тестілерін (Acceptance tests) бір уақытта дайындау талаптарынан тұрады;

Нұсқаны жоспарлау — фаза тапсырыс берушінің қатысуымен және келесі нұсқа құрамына кіретін User Stories таңдау арқылы орындалады. Нұсқаны жүзеге асырумен байланысты шешімдер қабылданады. Жоспарлау мақсаты – 1-3 аптада өнімнің келесі нұсқасын қалай құру керектігін бағалау;

Дайындау — бұл фаза жоспарға сәйкес орындалады және жоспарлау кезеңінде таңдалған функциялардан тұрады;

Тестілеу — бұл фаза тесттерді құрауға қатысатын тапсырыс берушінің қатысуымен орындалады;

Релиз шығару — бағдарламалық өнімнің дайындалған нұсқасы пайдалану немесе бета-тестілеу үшін тапсырыс берушіге жіберілетін фаза.

Цикл аяқталғаннан кейін дайындаманың келесі итерациясына өту орындалады.

XP өміршеңдік циклы моделінің ерекшеліктері бұл әдістің

келесідей қағидаларын анықтайды. Ең алдымен бұл - «жанды» дайындама мағлұматында бекітілген БӨ «жанды» дайындама қағидалары.

1. Адамдар және олардың байланысы үрдістер мен құралдарға қарағанда өте маңызды.
2. Жұмыс істеуші бағдарлама түпкілікті құжаттамаға қарағанда маңыздырақ.
3. Шартты талқылаумен салыстырғанда, тапсырыс берушімен серіктестік маңыздырақ.
4. Өзгерістерді жетілдіру жоспарға сәйкес орындауға қарағанда маңызды.

Одан басқа, XP моделінде өміршеңдік цикл моделінің ерекшелігін сипаттайтын бірнеше ережелер бар.

Жанды жоспарлау (planning game) — бағдарламалық өнімнің келесі нұсқасына дейін жасалуы тиіс жұмыстар көлемін тезірек анықтау. Шешім ең алдымен тапсырыс берушінің бизнес-басымдылықтарының негізінде, екіншіден техникалық бағалау негізінде қабылданады. Жоспарлар тапсырыс беруші қалауына немесе ақиқаттылығына байланысты өзгереді.

Нұсқаларды жиі ауыстыру (small releases) — бірінші жұмыс істейтін нұсқа тезірек пайда болуы тиіс және ол бірден қолданыла бастайды. Келесі нұсқалар қысқа уақыт аралығынан кейін дайындала бастайды.

Қарапайым жобалық шешімдер (simple design) — әрбір уақыт мезетінде жүйе барынша қарапайым қалыптасуы керек. Жаңа функциялар тек анық өтініштен кейін қосылады. Барлық артық қиындықтар анықталған сәттен бастап жойылады.

Тестілеу негізінде дайындау (test-driven development) — алдымен тесттер жазылады, одан кейін тесттер орындалуы үшін модульдер өткізіледі. Тапсырыс берушілер алдымен жүйенің негізгі мүмкіндіктерін көрсететін тесттерді жазады, сол арқылы жүйенің іске қосылғандығын көре алады.

Тұрақты қайта өңдеу (refactoring) — артық қиындықтарды жою, кодтың түсініктілігін, оның икемділігін арттыру жүйелері. Басқа шешімдермен салыстырғанда, қажетті нәтижені беретін икемді және сыпайы шешімдерге басымдылық беріледі.

Жұппен бағдарламалау (pair programming) — барлық кодтар екі бағдарламашымен бір компьютерде жазылады, бұл оның сапасын арттырады.

Тұрақты интеграция (continuous integration) — жүйе жиналады және бағдарламашылар жұбы тиісті функцияны орындауды аяқтаған кезде интеграциялық тестілеуден күніне бірнеше рет өтеді.

40-сағаттық жұмыс аптасы — мерзімнен тыс жұмыс

жобадағы үлкен мәселелер белгісі ретінде қарастырылады. 2 апта қатарынан орындалатын мерзімнен тыс жұмысқа жол берілмейді, бұл бағдарламашыларды шаршатады және олардың жұмыс өнімділіктерін төмендетеді.

Бақылау сұрақтары:

1. Бағдарламалық өнімнің өміршеңдік циклының моделі дегеніміз не?
2. Бағдарламалық өнімді құру үрдісіне не жатады?
3. Бағдарламалық өнімді дайындаудың өміршеңдік циклы моделінің кезеңдері дегеніміз не?
4. Бағдарламалық өнімді дайындаудың өміршеңдік циклы құрамына қандай кезеңдер кіреді?
5. Бағдарламалық өнімді дайындаудың қандай негізгі модельдерін білесіз?
6. Осы модельдердің қағидалы айырмашылықтары неде?
7. Төмендегі модельді түсіндіріңіз және сипаттаңыз: а) V-образды; б) RAD-модель; в) көп өтпелі; г) прототиптеу; д) каскадты; е) спиральді.
8. Төмендегі модельдер арасындағы айырмашылық қандай?

а) аралық бақылауы бар және каскадты;

б) спиральді және каскадты;

в) дәстүрлі спиральді және жетілдірілген, немесе өзгертілген және спиральді?

9. Келесі үрдістердің мақсаттары қандай?

а) қосымша;

б) бағдарламалық өнімді тексеру;

в) бағдарламалық өнім конфигурациясын басқару;

г) сапаны қамтамасыз ету;

10. Тексеру түрлерін санап шығып, сипаттаңыз.
11. Бағдарламалық өнімді дайындау үрдісіндегі метриканың рөлі қандай?
12. Үрдіс, жоба, өнім метрикасының мақсатын түсіндіріңіз.

4 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ДАЙЫНДАУ ҮРДІСІН ҰЙЫМДАСТЫРУ

4.1. Бағдарламалау дағдарысы және одан шығу тәсілдері

Бағдарламалар тіпті ең жақсы әдістемелер мен үлгілерге сүйенсе де, өздігімен пайда болмайды. Бағдарлама - адамдардың еңбектері қалай ұйымдастырылуына байланысты олардың қызметтерінің нәтижелері, көп деңгейде дайындалатын БӨ-нің сапасына тәуелді болады.

1970 жылдардың басында АҚШ-та бағдарламалау дағдарысы орын алды (software crisis). Бұл көптеген жобалардың графиктен қалуымен немесе шығындар сметаларының өсуімен түсінідірілді, дайындалған БӨ талап етілетін функционалды мүмкіндіктерге ие болған жоқ, оның өнімділігі төмен болды және алынған бағдарламалық жасақтаманың сапасы пайдаланушылардың көңілінен шыққан жоқ.

Жетекші шетелдік талдаушылармен орындалатын аналитикалық зерттеулер мен шолулар үміттендіретін нәтижелер берген жоқ. Мысалы, 1995 жылы Standish Group компаниясы 364 американдық корпорацияның жұмысын талдады, сонымен бірге, БӨ дайындамасымен байланысты 23 мың жобаның орындалу нәтижелеріне талдау жүргізді және келесідей қорытындығы келді:

- жобалардың тек 16,2 % ғана уақытында аяқталды, жоспарланған бюджеттен асқан жоқ және барлық талап етілетін функциялар мен мүмкіндіктерді қамтамасыз етті;
- жобалардың 52,7 % кешігіп аяқталды, шығындар жоспарланған бюджеттен асып кетті, талап етілетін функциялар толығымен орындалған жоқ;
- жобалардың 31,1 % аяқталғанға дейін күшін жойды.

Кешігіп аяқталған немесе күшін жойған жобалар үшін бюджет орта есеппен 89 %-ға, ал орындау мерзімі 122 %-ға артты.

1998 жылы жобалардың пайыздық қатынасы жақсы жаққа бірнеше есе өзгерді (26, 46 және тиісінше 28 %).

Болжамды сәтсіздіктердің негізгі себептері: БӨ талаптарының дәл емес және толық емес қалыптасуы, жоба жұмысына пайдаланушыларды жеткіліксіз тарту, қажетті ресурстардың болмауы, қанағаттанарлықсыз жоспарлау, талаптар мен

сипаттамаларды жиі өзгерту, пайдаланылатын технологияның жаңашылдығы, жобаны сауатты басқарудың болмауы, жоғарғы басшылық тарапынан қолдаудың аз болуы.

Соңғы уақытта жетекші шетелдік талдаушылардың пікірі бойынша, жобалардың сәтсіз болуының бірден-бір себебі - олардың төтенше жағдайларда орындалуы. Ағылшынтілді әдебиетте бағдарламалық инженерия саласындағы әлемдік жетекші мамандардың бірі Эдвард Иордан келесідей өрнекті бекітті «death march» немесе «қатерлі марш». Бұл параметрлері қалыпты мәндерден ауытқитын (тым болмағанда 50 %-ға) жобаны білдіреді. БӨ құру жобалары бойынша бұл келесі шектеулерді біреуінің болуымен түсіндіріледі:

1) жоба жоспары қалыпты есептік жоспармен салыстырғанда жартысына ықшамдалған, яғни, қалыпты шарттарда 12 айды талап ететін жұмыс 6 айда немесе одан аз уақытта орындалуы керек. Әлемдік нарықтағы қатаң бәсекелестік бұл жағдайды кеңейтеді.

2) дайындаушылар саны аталған масштабтағы жобаға қажетті мөлшермен салыстырғанда жартысына дейін қысқартылады, әдетте, дағдарыс немесе қайта ұйымдастыру нәтижесінде компания штатын қысқартудан туындайды.

3) бюджет және онымен байланысты ресурстар жартысына дейін қысқартылған (компанияның қысқаруы немесе басқа шығынға қарсы шаралар, табысты шартқа бәсекелі күрес), бұл жалданатын жұмысшылар санын қысқартады немесе төмен жалақыдағы тәжірибесіз жас жұмысшылар санын тартады.

4) функцияларға, өнімділікке, басқа техникалық сипаттамаларға қойылатын талаптар қалыпты шарттардағы мәннен екі есе асады.

БӨ дайындау үрдісін бақылау қажеттілігі, дайындау құнын, нәтижелер мерзімі мен сапасын болжау және кепілдеу 1970 жылдардың аяғында майдагерлік тәсілден индустриалды тәсілге өтуге және БӨ құрудың инженерлік әдістері мен құралдарының құрылуына алып келді, олар «бағдарламалық инженерия» деген ортақ атаумен біріктірілді (software engineering). Ең алғаш бұл термин 1968 жылы НАТО қамқорлығымен өткізілген конференция тақырыбы ретінде қозғалды. Жеті жыл өткеннен кейін, 1975 жылы Вашингтонда бағдарламалық инженерия бойынша бірінші халықаралық конференция өтті. Сол кезде бағдарламалық инженерияға арналған бірінші баспа пайда болды — «IEEE. Transactions on Software Engineering» («IEEE. Бағдарламалық инженерия бойынша еңбектер»).

Бағдарламалық инженерияны құру және дамыту үрдісінде екі кезеңді бөлуге болады: 1970-1980 жылдар – БӨ құру үрдістерін

жүйелендіру және стандарттау (құрылымдық тәсіл негізінде); 1990 жылдар – БӨ құрудың құрастыру, индустриалды тәсіліне өтудің бастамасы.

Бағдарламалық инженерия негізінде іргелі идея жатыр: БӨ жобалау игеруге және жетілдіруге болатын ресми үрдіс болып табылады. БӨ құрудың әдістері мен құралдарын дұрыс игеру және пайдалану оның сапасын арттырады, БӨ жобалау үрдісін басқаруды қамтамасыз етеді және оның өмір сүру мерзімін ұзартады.

БӨ дайындау үрдісіндегі кез-келген өзгерістерде жүргізілген өзгерістерден алынған әсерді бағалауға мүмкіндік беретін критерийлер қажет болады. Басқа сөздермен айтқанда, жұмыс тиімділігі мен сапасын өлшеуге мүмкіндік беретін қажетті әдістер. Ұйым– жобаны дайындаушылардың барлық топтарының жиынтығы (әрбір жоба өз БӨ-ін құруға бағытталған) және осы үрдіске қатысты әкімшілік.

Қазіргі таңда бағдарламалық жасақтаманы дайындау үрдісінің жағдайын бағалайтын екі жалпы қабылданған әдістеме бар: ISO 9001 халықаралық стандарты және CMM-SEI моделі Capability Maturity Model (ұйымдағы технологиядық үрдістердің жетілуін бағалау моделі), ол Software Engineering Institute бағдарламалық инженерия институтында дайындалған.

4.2. CMM-SEI моделі

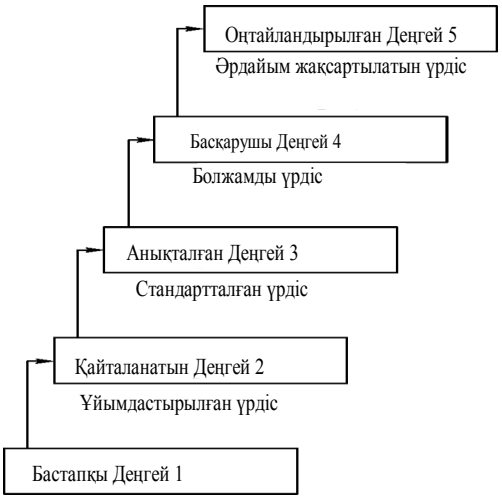
1986 жылы SEI институты (Карнеги – Меллон университетінің бөлімшесі) Mitre корпорациясының көмегімен бағдарламаларды дайын- даудың тиімді үрдісінің модельдерін дайындауды бастады. Бұл үрдісті сипаттай отырып, модель авторлары «жетілу» түсінігін пайдаланды, бұл түсінік үрдіс тиімділігін ғана емес, сонымен бірге, тұрақтылықты, сенімділікті білдірді.

«CMM» атауына ие болған модельдің алғашқы нұсқасы 1987 жылдың аяғында шығарылды. Одан кейін модель бірнеше рет қайта өңделді, қазіргі таңда оның кезекті нұсқасы дайындалуда.

Осы модельге сәйкес, ұйым жетілудің бес деңгейінің біреуінде болады (сурет 4.1).

Бірінші немесе *бастапқы* деңгейге бағдарламаны дайындаудың кенеттен немесе ретсіз үрдісі сипатты болып табылады. Дайындау процедуралары анықталмаған, жетістік жұмысшылардың жеке күш салуларына тәуелді болады.

Қайталанатын деп аталатын екінші деңгейде БӨ функционалды сипаттарын және оның құнын қадағалауға мүмкіндік беретін басқарудың негізгі үрдістері пайдаланылады.



Сурет 4.1.
Ұйымның жетілу деңгейлері

Ұйым жаңа жоба дайындамасын ұқсас мүмкіндіктермен қайта сәтті орындай алады.

Үшінші деңгей *анықталған* деп аталады.

Осы деңгейдегі ұйым БӨ дайындаудың барлық басқарушылық және инженерлік тапсырмаларын құжаттайды, стандарттайды және басқарудың бір үрдісіне біріктіреді, сонымен бірге ұйымның стандартты үрдісін дайындайды. Ұйымдағы барлық жобалар дайындаудың бекітілген тәсілдерін және нақты жобаға бейімделген бағдарлама қолдауларын пайдаланады.

Төртінші деңгей *басқарылатын* деп аталады. Осы деңгейдегі ұйымда дайындалатын БӨ мен үрдіс сапасын егжей-тегжейлі өлшеу тәсілдері болады. Дайындау үрдісінің және дайындалатын БӨ-нің сандық сипаттары жақсы оқытылған және басқарылған, үрдіс - белгілі.

Ұйымның ең жоғары деңгейі *оңтайландырылған* деп аталады. Онда орындалған және орындалатын жобалардың сандық сипаттарына және жаңа технологиялар мен идеяларды енгізуге негізделген дайындау үрдісін үздіксіз жақсарту жүзеге асырылады.

Жетілудің жоғарылауы бірінші деңгейден бесінші деңгейге қарай орындалады. БӨ дайындаумен айналысатын көптеген ұйымдар бірінші деңгейден бастайды (кейбіреулері сол деңгейде қалады). Жетілмеген ұйым дайындаудың аяқталуы мерзімін және шекті өнім сапасын болжай алмайды. Жетістік тікелей басшының тәжірибесіне және дайындаушылардың квалификациясы мен талантына тәуелді болады. Басқару үрдісін жиі болатын дағдарыстарда емес тек жобаны орындау кезінде құру қажет.

Мұндай ұйым БӨ дайындауды сәтті орындай алмайды деп ойлауға болмайды, дегенмен дайындау графигі орындалмайды және жүйенің құны жоспарға қарағанда жоғарырақ болады.

Сәттіліктің қайталануы тек дайындауда сол қызметкерлер қатысқан жағдайда орындалады. Екінші немесе жоғары деңгейлерге қол жеткізу үшін ұйымдар негізгі анықталған үрдістерден тұратын үрдісті құруы керек.

■ Оңтайландырылған деңгейдегі негізгі үрдістер:
- үрдіс өзгерістерін басқару;
- жаңа заманауи технологияларды пайдалану;
- қателерді жою.

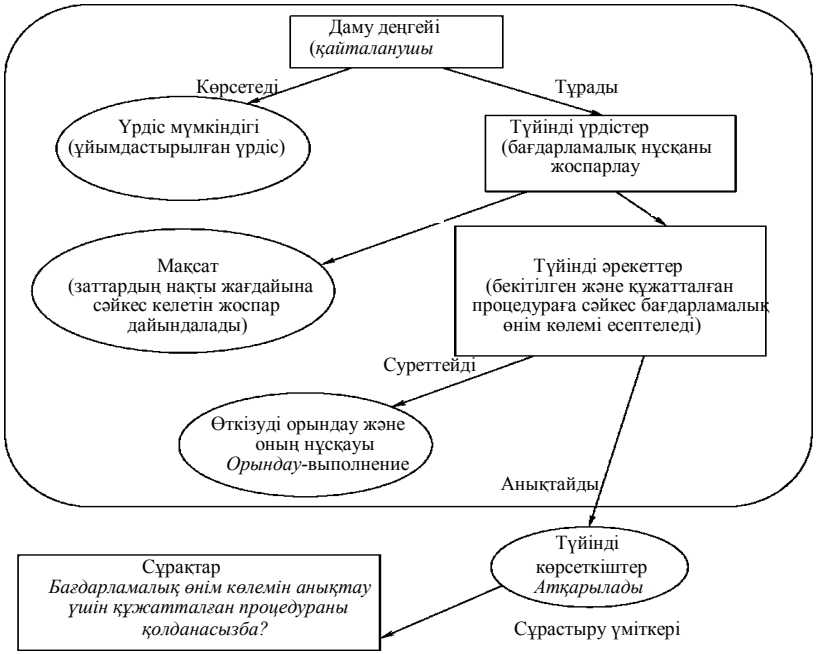
■ Басқарылатын деңгейдегі негізгі үрдістер:
- үрдіс сапасын басқару;
- үрдісті өлшеу және талдау.

■ Анықталған деңгейдегі негізгі үрдістер:
- жұмыс нәтижелерін ұжымдастармен бағалау және талқылау;
- жобалар арасындағы үйлесімділік және өзара әрекеттестік;
- қызметкерлердің квалификациясын арттыру;
- ұйымдық үрдістерді анықтау;
- ұйымдық үрдістерге ерекше көңіл бөлу;
- БӨ дайындау және жобалаудың индустриалды тәсілі;
- ұйымның стандартты үрдісіне негізделетін барлық жобаларды бірігіп басқару.

■ Қайталанатын деңгейдегі негізгі үрдістер:
- БӨ конфигурациясын басқару (нұсқаларын);
- БӨ сапасын қамтамасыз ету;
- қосалқы мердігерлер жұмысын басқару;
- бағдарламалық жобаны басқаруды бақылау;
- бағдарламалық жобаны жоспарлау;
- БӨ талаптарын басқару.

Бастапқы деңгейде негізгі үрдістер жоқ.

Кез-келген негізгі үрдіс осы үрдіске қатысты негізгі әрекеттерді ұйымның орындауымен түсіндіріледі, бұл өз кезегінде ұйымға осы үрдіспен орындалатын белгілі мақсаттарға жетуге мүмкіндік береді. Негізгі үрдістер аталған үрдісті орындауға мүмкіндік берумен қатар, оны жүзеге асырудың нұсқаулығы болып табылады. Әртүрлі негізгі үрдістердің орындалуы немесе орындалмауы ұйымның жетілу деңгейін талқылауға болатын көрсеткіш болып табылады. СММ моделіндегі осы деңгейді анықтау үшін ұйым қызметкерлерімен қойылатын сұрақтар тізімі қарастырылады. Бұл сұрақтар жауабының талдамасы жетілу деңгейі туралы қорытынды жасауға мүмкіндік береді. Ұйым жетілуінің екінші деңгейіндегі «Бағдарламалық жобаны жоспарлау» негізгі үрдісіне арналған СММ құрылымының мысалы 4.2 суретте көрсетілген.



Сурет 4.2 СММ құрылымының мысалы

СММ моделі қызметкерді таңдау мәселесін қарастырмайды. Ол БӨ дайындамасын таза өндірістік үрдіс ретінде қарастырады. Ұйымдық құрылым және басқару үрдістері дайындаушылардың қабілеттіліктерінің тиімді қолдануына көмектеседі. Әртүрлі ұйымдардағы үрдістер пайдаланылатын технологиялар мен әдістер сияқты әртүрлі болады. Барлық үрдістердің жалпы талаптарына БӨ дайындауды басқару жүйесінің негізін құру жатады.

4.3. ISO 9001 стандарттар жүйесінің көмегімен бағдарламалық өнімді дайындау сапасын басқару

Халықаралық стандарттау ұйымы (ХСҰ) сапаны басқару мәселесін реттейтін ISO 9001 стандартының жүйесін дайындады. Бұл стандарттар тауарлар мен қызметтердің, БӨ-нің барлық салаларында қолданылады.

СММ модельдері мен ISO 9001 стандарттары арасындағы өзара әрекеттестік келесідей: ISO 9001 талаптар тізімінен тұрады, ал СММ сапаны басқару бойынша құжаттарға қосу үшін дайындау үрдісінің талаптарын анықтайды. Ресей ХСҰ

мүшесі бола отырып, ISO 9001 стандарттар жүйесін өзінің ұлттық стандарты ретінде қабылдады. Стандарттаудың Жалпы ресейлік ғылыми-зерттеу институты оны орыс тілінде аударды.

ISO 9001 мақсаты - дайындаудың барлық кезеңдерінде сапаны қамтамасыз ететін толассыз басқару жүйесін құру (TQM — Total Quality Management).

ISO 9001 және CMM-SEI стандарттарында жүйеде көрсетілген стандарттар мен модельдерге ұйым сәйкестігін сертификаттау процедуралары көрсетілген. Мұндай сертификаттаудан өту қиын болып табылады, көптеген компаниялар өтуге тырысады. Стандарттар мен модельдерге сәйкестікті ресми сертификаттау бәсекелестер алдында бәсекелік артықшылықтар береді. Орындаушыны таңдау кезінде тапсырыс беруші жұмыстың жоғары сапада орындалуына сенімді болады. Кейбір ірі компаниялар байқаулар мен тендерлер жүргізгенде байқауға қатысатын компаниялар қандай деңгейде ISO 9001 стандарттарына және CMM-SEI моделіне сәйкес екендігі туралы ақпараттарды талап етеді.

ISO 9001 стандарттарының жүйесі сапаны басқару талаптарының минималды жинағын анықтайды. Шартты түрде бұл жинақ үшке бөлінеді: компания менеджментіне, дайындау үрдісіне, өнімді бақылауға қойылатын талаптар.

Егер компания менеджменті оның мәнін түсінбесе немесе оны құру мақсатын қоймаса, сапаның тиімді жүйесі мүмкін болмайды. Компания басшылығы өнімдер мен қызметтердің жоғары сапасын сақтау бейілділігін растайтын ресми хатқа қол қоюы керек.

Хатта сапаны бақылау кезінде негізделуге болатын негізгі құжаттар көрсетіледі. Компания басшылығы сапаны бақылауды қамтамасыз ететін бөлімшені құрайды. Одан басқа, мерзімдік тексеріс процедуралары және сапаны басқару жүйесінің тиімділігін талқылаулар анықталады.

Өнімді басқаруға дайын пакеттер мен бағдарламаларды, өнімдерді алумен жүйе нұсқаларын бақылау жатады, қазіргі таңда ол сапа талаптарына жауап бермейді. Стандарттың көп ережелері өнімді басқару бөлімінде бағдарламалық жасақтамаға жатпайды немесе екінші деңгейлі болып табылады (мысалы, қаптамаға немесе сақтауға қатысты ережелер).

Дайындау үрдісін басқару - бағдарламаумен айналысатын ұйымдар үшін ISO 9001 стандартының басты бөлігі. Ол БӨ дайындау үрдісінің құжаттамасынан және құру талаптарынан тұрады – келісімшартқа отырған сәттен бастап дайын өнімді шығаруға дейінгі мерзімді қамтиды (бұл кезеңде дайындаманы басқару өнімді басқаруға өтеді).

ISO 9001 классификациясы бойынша, бағдарламаны дайындау өнім ақаулықтары оны пайдаланғанға дейін байқалмайтын арнайы үрдістерге жатады.

ISO 9001 стандартының сипаттамасы түрлі ұйымдарда әртүрлі болатын дайындау үрдісінің өзін реттемейді. Ол үрдістің сапаны толассыз бақылау талаптарына сәйкестік критерийлерін стандарттайды. Бірінші қажетті шарт ретінде нақты ұйымдағы нақты үрдісті реттейтін құжаттаманың болуы қарастырылады.

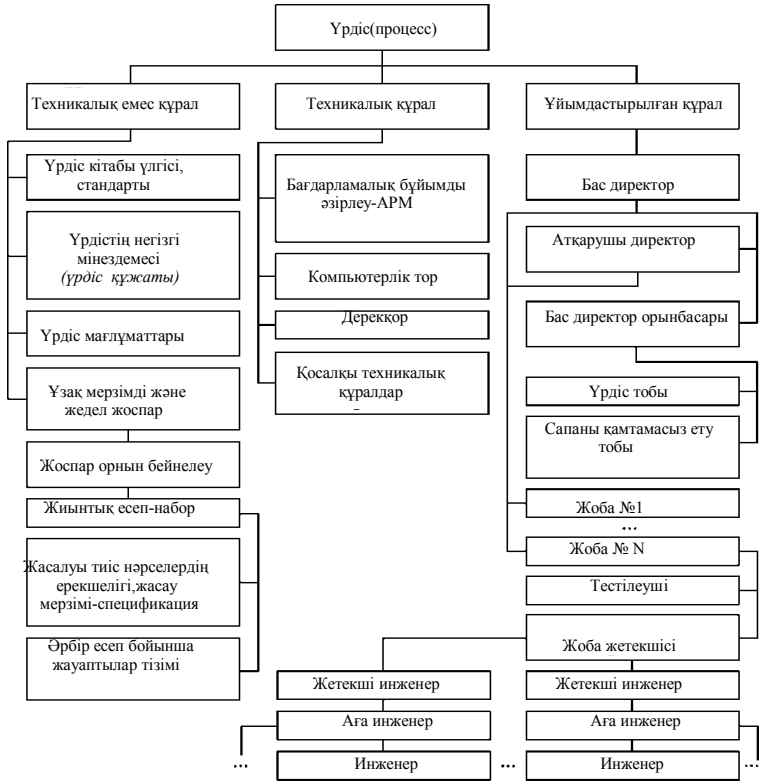
4.4. Бағдарламалық өнімді дайындаумен айналысатын ұйымның және үрдістің болжамды құрылымы

Компанияның болжамды және басқарылатын үрдісін ұйымдастыру үшін ұйымдық, техникалық және техникалық емес құралдар қажет (сурет 4.3).

Ұйымдық құралдарға әртүрлі қызмет тізімдері және жоғары тұрған басшылыққа қызметкерлердің бағыну иерархиясы жатады.

Компания жұмысын ортақ басқаруды бас директор орындайды. Әртүрлі жобалардың орындалуымен байланысты сұрақтарды атқарушы директор, ал ұйым және компания үрдісін қамтамасыз ету мәселелерін (яғни, компания әрекет ететін нұсқаулықтарды, ережелер жинағын, процедураларды, басқа басқарушылық құжаттарды) және БӨ сапасын қамтамасыз ету жұмыстарын бас директордың орынбасары басқарады. Жұмыстарды бұлай бөлу компаниядағы үрдіс құру маңыздылығын және БӨ сапасын қамтамасыз ету жұмыстарын өткізуді білдіреді.

Қажет болған жағдайда компаниядағы екі топтың орнына (үрдістер тобы және сапаны қамтамасыз ету тобы) үрдістің бір тобы ғана қалуы мүмкін, бірақ бұл топ БӨ сапасын қамтамасыз ету жұмыстарын да орындауы керек. Одан басқа, әрбір жобада БӨ сапасына жауап беретін тұлға тағайындалуы керек. Әдетте бұл жоба жетекшісі немесе жетекші инженердің біреуі. Сапаға жауап беретін тұлға үрдіс тобының немесе сапаны қамтамасыз ету (егер бұл топ өздігінен қызмет етсе) тобының өкілі болып табылады және компания үрдісіне, сапаны қамтамасыз етуге қатысты барлық әрекеттердің орындалуына жауап береді.



Сурет 4.3. Бағдарламалық өнімді дайындаумен айналысатын ұйым мен үрдістің болжамды құрылымы

4.3 суретте көрсетілген, тәуелсіз тесттен өткізуші жоба жұмысына қатысады, бірақ жоба жетекшісіне тәуелді болмайды. Бұл құжаттамаға және аталған жобада дайындалатын БӨ-ге объективті тестілеуді өткізуге мүмкіндік береді.

Тесттен өткізуші егер жобаның ағымдағы кезеңдері сәйкес болмаса, бір уақытта бірнеше жобаларға қатысады. Компанияның барлық тесстен өткізушілері кіретін тестілеудің жеке тобын жиі құрып жатады.

Техникалық құралдар жобамен жұмыс істеудің және компания үрдісін қолдаудың тиісті шарттарын ұйымдастыру үшін, сонымен қатар, бағдарламалық өнімнің сапасын қамтамасыз ету үшін қолданылады. Мысалы, автоматтандырылған жұмыс орны (АЖО) бағдарламашы жұмысының өнімділігін және дайындалатын БӨ сапасын арттырады, ал компьютерлік желі компаниядағы электронды құжат айналымын және қызметкерлер арасындағы байланысты қамтамасыз етеді. Деректер базасы бұрын орындалған және ағымдағы жобаларды орындауға

қатысты барлық ақпаратты сақтауға мүмкіндік береді.

Техникалық емес құралдар пайдалануға қабылданған немесе дайындалған стандарттар мен жоспарларды, сонымен қатар компания үрдісін егжей-тегжейлі сипаттайтын үрдіс кітабын қамтиды. Үрдіс метрикасы бойынша оның негізгі сипаттамаларын бағалайды (негізгі үрдістері) және бағалау нәтижелерін үрдіс төлқұжатына енгізеді. Бұл төлқұжат үрдістің сақталуын қадағалайды, сонымен бірге, оны жетілдіру бойынша әрекеттерді жоспарлайды.

Бақылау сұрақтары:

1. Бағдарламалау дағдарысы неден пайда болды?
2. Бағдарламалау дағдарысынан шығу тәсілі қандай?
3. Қандай жобаны «қатерлі марш» ретінде анықтайды?
4. Бағдарламалық өнімді дайындау жағдайын бағалаудың қандай әдістерін білесіз?
5. СММ моделіне сәйкес ұйымның жетілуінің негізгі деңгейлерін атаңыз және сипаттаңыз.
6. СММ моделінің әрбір бес деңгейінде қандай негізгі үрдістер орындалуы керек?
7. СММ құрылымы қандай?
8. ISO 9001 стандарты жүйесінің мақсаттары мен қолданылу бағытын анықтаңыз.
9. ISO 9001 стандарты жүйесіне сәйкес, сапасы басқару талаптарының минималды жинағын атаңыз және сипаттаңыз.
10. Қандай талаптар басқаруға қойылады: а)компаниямен; б) өніммен; в)дайындамамен?
11. Бағдарламалық өнім дайындамасымен айналысатын ұйым және үрдістің болжамды құрылымын түсіндіріңіз.
12. Құралдарға жатады: а)ұйымдық; б)техникалық; в) техникалық емес?

5 БӨЛІМ

МЕТРИКАЛАР

5.1. Бағдарламалық өнімдерді дайындау үрдісіндегі метрикалардың рөлі

БӨ дайындау үрдісінде орындалатын өлшеулер ұйымда қабылданған дайындау үрдісін, жобаның орындалуын, БӨ сапасын бағалауға мүмкіндік береді. Үрдістің өлшемі оны ары қарай жетілдіру үшін орындалады, жобаның өлшемі – жұмысты ұйымдастыруды жетілдіруді, ал БӨ өлшеу – оның сапасын арттыруды мақсат етеді. Өлшеу нәтижесінде өлшеу объектісінің қандай-да бір сипатының сапалы сипаттамасы анықталады. Тікелей өлшемдер арқылы объектінің тірек қасиеттері, яғни, тірек метрикалары анықталуы мүмкін. Барлық қалған метрикалар тірек метрикалары мәндеріндегі функцияларды есептеу нәтижесінде бағаланады. Бұл есептеулер тиісті формулалар бойынша орындалады.

«IEEE. Standard Glossary of Software Engineering Terms» баспасында ((«IEEE. Бағдарламалық инженерияда қолданылатын стандартты терминдер тізімі») метрика сандық мәннен тұратын қасиетке ие болу деңгейіндегі шама ретінде анықталды.

Метрика ретінде БӨ-ді, жобаны немесе үрдісті сандық бағалау жатады, ол басқа өлшемдер немесе болжамдар орындалатын негізде тікелей қолданылады.

БӨ дайындау саласында әлемдік танылған сарапшы Д-р Барри У. Боэм (Dr. Barry W. Boehm) белгілегендей, метриканың маңыздылығы олардың шешім қабылдауға қаншалықты әсер ететітіне байланысты анықталады. Егер бағдарламалық жоба жетекшісі осыны есіне сақтайтын болса, онда ол ақпараттың көп бөлігін жинақтап, күрделі пайдаланылатын метрикаларды кездейсоқ жинамай, пайдалы және маңызды метрикаларды пайдаланады.

«Бағдарламалық өнімді дайындаудағы өлшемдер бағдарламалық үрдіске жататын және оған сәйкес өнімдер бойынша деректерді анықтаудың, жинаудың, талдаудың үздіксіз үрдісі болып табылады. Бұл қызметтің мақсаты – үрдіс туралы ұғымдарды алу, оларды және бағдарламалық өнімді бақылау, сонымен бірге, бағдарламалық өнімді жетілдіруге ықпал ететін маңызды ақпаратты қолдау [5].

«Бағдарламалық өнімді дайындаудағы өлшемдер –

бағдарламалық инжинерия, бағдараламалық өнім немесе мәнмәтін үрдісінің ерікті аспектілерін сандық бағалау. Ол ұғымдарды жетілдіруге, бақылауға, болжамдауға және құрылатын өнімге, сонымен бірге пайдаланылатын жұмыс әдістеріне жақсартулар енгізуге көмектеседі» [7].

Барлық метрикаларды үш негізгі топқа бөлуге болады: үрдіс метрикалары; жоба метрикалары; өнім метрикалары.

Әрбір топтың ішінде метриканың келесідей түрлері болады: тікелей бақыланатын (өлшенетін), болжамды, есептелетін.

Қандай-да бір объет атрибутын тікелей бақылау өлшеу үдісінде басқа атрибуттарды немесе объектілерді пайдалануды талап етпейді. Тікелей бақылау немесе өлшеу тиісті объектіні бағалау кезінде қолданылады.

Болжау кезінде таңдалған атрибуттың математикалық моделі болжау процедурасымен бірдей қолданылады, олар белгісіз параметрлерді және нәтижелер түсіндірмесін анықтау үшін қолданылады.

Есептеу немесе жанама өлшеу басқа атрибуттардың немесе объектілердің анықталған математикалық модельдер көмегімен өлшеу үрдісіне тартуды білдіреді (кемінде екі метриканы пайдалану арқылы есептеуден тұрады).

5.1 кестеде БӨ, үрдістер, жобаға қатысты метрикалар типтерінің мысалы көрсетілген.

Метрикалар объективті және субъективті болады. Субъективті өлшеулер тұлғалық, субъективті тәсілдердің болуын, мысалы, қандай-да бір салмақтық коэффициентті пайдалануды білдіреді.

Өлшенетін атрибуттар сыртқы және ішкі болуы мүмкін. Ішкі атрибуттар объект терминінде өлшенуі мүмкін. Сыртқы атрибуттар сыртқы ортамен объектінің байланысы есебімен бағаланады. Сыртқы атрибуттар мысалдарына бағдарламалық өнімдегі (LOC) код жолдары санының көрсеткіші, әрекеттердің орындалу ұзақтығы, еңбек сыйымдылық шамасы, сәтсіз тесттік зерттеулер саны, ақшалай қаражаттар көлемі, модульділік және күрделілік деңгейі жатады. Сыртқы атрибуттар ретінде орындау уақытын (бағдарлама және компьютер), ұсыну пайдалылығын және ыңғайлылығын (бағдарлама және пайдаланушы талап етіледі), сенімділік, тиімділік, тестілеу, қайта пайдалану, операциялар расындағы өзара әрекеттестік пен көндігу қабілеттілігі қарастырылады.

Бағдарламалық жобалардағы өлшеулер әртүрлі деңгейдегі басшыларға көмек ретінде, және қарастырылатын деректерге тәуелді дайындаушыларға уақытылы және дұрыс шешімдер қабылдауға көмек ретінде қарастырылады. Одан басқа, олар үрдіске жетілдірулерді енгізуде жұмыс ұйымдастыру

үрдісін қадағалауды жеңілдетеді, енгізілген прогрессивті өзгерістерді бағалауға мүмкіндік береді. Бағдарламалық метрикалық көрсеткіштер БӨ, жоба немесе үрдістің анықталған атрибуттарын өлшеу үшін қолданылады.

Жоба жетекшісі келесі әдістерді қолдана алады: БӨ ақаулықтары мен қателерін талдау; БӨ қалпын бағалау; БӨ күрделілік деңгейін анықтау; дайындаудың негізгі бағыттарын қалыптастыру; ең жақсы әдістемелерді тәжірибелі растау; сапалы көрсеткіштерді, графиктерді, еңбек сыйымдылықты және басқа шығындар көлемін болжау; жобаны орындау кезінде үрдісті қадағалау; өнім сапасының қажетті деңгейіне жетудің оңтайлы мерзімін анықтау.

Жиналатын метрикалар шешім қабылдау үрдісін жақсарта және тездете алады, нәтижесінде олардың әрқайсы тікелей өлшеуді орындайды немесе өлшеу нәтижелерін пайдаланады, тек ұтады. Бұл кез-келген пайдалануы, немесе мүдделері жобаны сәтті аяқтаумен байланысты тұлғалар тобы (тапсырыс берушілер, демеушілер, шекті пайдаланушылар, ұйымның басты менеджерлері, жоба жетекшілері, пән саласының сарапшылары, авторлар, талдаушылар, дайындаушылар, БӨ сапасын қамтамасыз етуші инженерлер, бағдарламашылар және тестілеушілер).

Менеджерлер жобаны басқару әдістерін анықтауда, сонымен қатар, БӨ дайындамасы үрдісін жақсарту бойынша өзгерістерді бағалауда метрикалық параметрлерді қолданады, дайындаушылар командасы оларды келесі тапсырмаларды шешу үшін қолданады:

Жобалау, құрылымдау, тестілеу, енгізу фазаларында талаптарды талдау кезінде қол жетерлік мақсаттарды қалыптастыру;

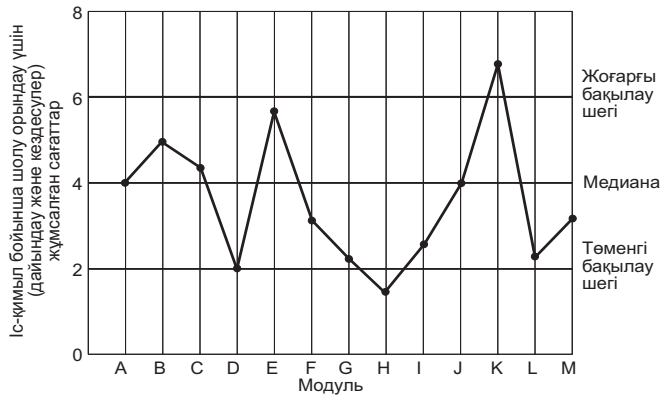
Осы мақсаттарға жетуде әлеуетті көрсету; мақсаттарға жетудегі алға жылжуды қадағалау; бақылаудан шығатын шекті шарттарды түзетуге арналған үрдістерді баптау (мысалы, егер шолулар маңызды болса, бірақ команда оған қосымша уақытын жұмсағысы келмесе, оларды 5.1 суретте көрсетілген бақылау графигі арқылы бағалауға болады).

Мүдделі тараптармен жинау, синтездеу, талдау, есепке қосу және метриканы игеру орындалады.

Үрдіс, жоба және өнім метрикасының мысалдары

Метрикалар тобы	Метрика мысалдары		
	Тікелей бақыланатын	болжамды	есептелген
Үрдіс метрикасы	Үрдіске сүйену; үрдісті қамту және оның тиімділігі; қызметкерді оқыту тиімділігі	CMM-SEI деңгейі	Бағалау дәлдігі (алдын-ала бағалаудың нақты мәнге қатынасы ретінде анықталады); өнімділік (дайындаушылармен бір айға жазылған код жолының (LOC) дайындаушылар санына қатынасымен анықталады); есеппен қамту; толтыруды тексеру немесе анықтау; оқыту тиімділігі; тәуекелдік ықтималдығы; метрикалық көрсеткішерді шолу; талаптарды талдау; метрикалық көрсеткіштерді қолдау.
Жоба метрикасы	БӨ дайындау уақыты; жоба құны; ресурстар қажеттілігі	Жобаны орындау ұзақтығы; жоба құну	Графикке сүйену (игерілген қаржының жоспарланған қаржы көлеміне қатынасымен анықталады, мысалы, орындалған жұмыстың нақты құнының бюджет құнының қатынасы).

Бағдарлама-лық өнім метрикасы	код жолының саны; тесттік мысалдар саны; жойылмаған ақаулықтар саны; тесттер саны; ақаулықтар саны ақаулықтар саны, олардың маңыздылығы; жүйе құраушыларының саны.	код жолының саны ; бағдарламалық өнім сапасы; бағдар-ламалық өнім ақаулықтарын-ың пайда болуы; қалған ақаулықтардың сипаттамалары және түсінігі; сенімділік; ақаулықтар саны, олардың маңыздылығы.	Тиімділік — жүйенің динамикалық беталысы; тиімділік— жүйе ресурстары; ақаулық жиіліктері (анықталған ақаулықтардың бағдарламалық өнім көлеміне қатынасымен анықталады: KLOC мәні, мұндағы 1 КШС= 1000 LOC); сенімділіктің қол жеткізілген деңгейін өлшеу; (мысалы, жасау уақытының мәні). отказ)



Сурет 5.1.Дайындаушылар командасымен қолданылатын қорытынды график мысалы

Аталған ұйым шеңберінде немесе нақты жоба ерекшеліктерін есепке ала отырып, кему реті бойынша басымдылықты сұрыптау жүргізіледі. Метрикалар басқа ұйымда қолдануға жарамауы мүмкін. Дегенмен, жұмыс үрдісінде растау алғаннан кейін және жоба деректерінің базасына енгізілгеннен кейін олардың маңыздылығы бірнеше есе артады. Олар келешек бағыттарды болжауда және ұйымдастыру немесе дайындау үрдісіне жетілдірулер енгізуде маңызды рөл атқаратын болады.

5.2. Метрикалар және CMM-SEI моделі

5.2.1. Екінші, қайталанатын, CMM-SEI моделінің деңгейі

Өлшеу нәтижелерін талдау және өлшеу SEI институтымен ұсынылған бағларламалық жасақтаманың жетілу моделінің құраушы бөлігі болып табылады (4.2 бөлімді қараңыз). CMM-SEI моделіндегі барлық үрдіс жетілудің кез-келген деңгейінде өлшеу және талдау құраушыларынан тұрады. Өлшеу нәтижелерін талдау және өлшеу үрдіс, жоба, өнім қалпын анықтау үшін қажет. Метриканы CMM-SEI моделінің деңгейлеріне сәйкес жинау және талдау 5.2 кестеде көрсетілген.

Негізгі үрдіс: БӨ талаптарын басқару. Өлшеу талаптарды басқару мақсатында орындалатын әрекеттерді анықтау үшін қолданылады. Метрика мысалдары:

Кесте 5.2

CMM-SEI моделінің әртүрлі жетілу деңгейіндегі метрикаларды талдау және жинау әрекеттері

Жетілу деңгейі	Әрекеттер
1	Деректерді жинау және талдау
2	Жекелеген жобаларда қолданылатын деректерді жоспарлау және басқару
3	Барлық анықталған үрдістерде жиналған деректерді қолдану. Деректерді әртүрлі жобаларда бірігіп жүйелі пайдалану.
4	Барлық ұйымдағы стандартталған деректерді жинау және талдау. Үрдіс параметрлерін сандық ұсыну және тұрақтандыру үшін деректерді пайдалану.
5	Үрдістегі жетілдірулерді бағалауға және бөлуге арналған деректерді пайдалану.

Қалыптасқан талаптардың әрқайсының жағдайы; қалыптасқан талаптарды өзгерту бойынша әрекеттер; қалыптасқан талаптардағы өзгерістер саны бойынша жиналған қорытынды, талаптардың жалпы санын есепке алғанда, олардың ішіндегі ұсынылған, ашық,бекітілген, базалық нұсқаға қосылған талаптар.

Негізгі үрдіс: бағдарламалық жобаны жоспарлау. Өлшеу жоспарлаумен байланысты түрлі әрекеттерді анықтау мақсатында орындалады. Метрикаға мысал ретінде білетіндеріміз мыналар.

Жобаны жоспарлауға сәйкес әртүрлі әрекеттердің негізін құраушы кезеңдердің аяқталу мерзімдері.

Жұмыс аяқталғаннан кейін анықталатын еңбек шығындары, бағдарламалық жобаны жоспарлаумен байланысты әрекеттер түрлері бойынша құралдарды бөлу.

5.2.2. Негізгі үрдіс: бағдарламалық жобаның орындалуын бақылау.

Өлшеу бағдарламалық жобаның орыдалуын бақылау үрдісінің жағдайын анықтау мақсатында орындалады. Метрика мысалдары: еңбек шығындары, бақылау жүргізудегі басқа ресурстардың шығындары, жобаны жоспарлаудағы өзгерістер саны, бағдарламалық жұмыс құны, есептік ресурстардың сыни көлемі және жұмыс графигі, дайындалатын БӨ көлемін бағалаудағы өзгерістер. **Үшінші, анықталған, CMM-SEI моделінің деңгейі**

Негізгі үрдіс: қызметкерлер квалификациясын арттыру. Өлшеу қызметкерлерді оқыту бағдарламасын жүзеге асыру әрекеттерінің әртүрлі формаларын анықтау мақсатында орындалады. Метрика түрлері:

- жоспарланатын қатысушылықпен салыстырғанда оқытудың әрбір курсының нақты қатысушылығы,
- оқытудың өткізілетін курстарына сәйкес, жобаның және ұйымның алға басуы,
- оқыту жүргізудегі жаңылысулар саны.

Одан басқа, өлшеу оқу бағдарламасының сапасын анықтау үшін орындалады. Метрика мысалдары былай бөлінеді.

- оқытудан кейін жүргізілетін тестілеу нәтижелері,
- студенттік сұраулар негізіндеге курстарды шолу,
- жоба жетекшілерімен кері байланысты қолдау.

Негізгі үрдіс: БӨ дайындау және жобалаудың индустриалды тәсілі.

Өлшеу БӨ сапасын және функционалды мүмкіндіктерді анықтау мақсатында орындалады. Метрика мысалдарын былай жіктеуге болады.

- БӨ-ге теңестірілген ақаулықтар маңыздылығы, түрлері, саны,
- талаптардың категорияларға сәйкес жіктелуін анықтау (мысалы, қауіпсіздік, жүйелік конфигурация, орындау және сенімділік), БӨ талаптары және жүйелік тесттік зерттеулер есебімен.

Одан басқа, өлшеу БӨ дайындау бойынша қызметтің әртүрлі формаларының жағдайын анықтау үшін орындалады. Метрика мысалы:

- жобаны орындаудағы әрбір талаптың жағдайы;
- уақыт шығыны және маңыздылық деңгейі көрсетілген мәселелер туралы есептер;
- әрбір қарастырылатын өзгеріс үшін ұсынылған өзгерістерді талдаудың еңбек шығындары және барлық өзгерістер бойынша кумулятивті бағалау;
- категориялар бойынша БӨ базалық нұсқасына енгізілген өзгерістер көлемі (мысалы, интерфейс, қауіпсіздік, жүйелі конфигурация, орындау және тәжірибелік);
- енгізілген өзгерістерді тестілеу және орындаудағы шығын көлемі, бастапқы бағалау және көлем/шығын арақатынасы.

5.2.3. Төртінші, басқарылатын, SEI CMM моделінің деңгейі

Негізгі үрдіс: үрдіс сапасын басқару. Аталған негізгі үрдіс мақсаттарды анықтау бойынша әрекеттерде, үрдіс өнімділігін

өлшеуді, осы өлшемдерді талдауды, баптауды, ұйғарынды шектеулер шеңберінде үрдістердің қызмет етуін қамтиды. Егер үрдісті орындауда ұйғарынды шектер шеңберінде оның тұрақтануы орындалса, онда жобамен анықталған үрдіс, онымен байланысты өлшеулер және өлшеудің ұйғарынды шектері бекітіледі және үрдісті орындаудың сандық бақылауын жүзеге асыруда қолданылады.

Негізгі үрдіс: өлшеу және үрдісті талдау. Өлшеу және талдау үрдістің ағымдағы жағдайын бақылау үшін және алынған деректер негізінде оны жақсарту үшін орындалады.

Метрика мысалдары мынадай болады.

- БӨ көлемі және құны туралы деректерді бағалау дәлдігі;
- әртүрлі жұмыстарды орындаудағы өнімділік;
- сапаны басқару жоспарына қатысты сандық көрсеткіштер;
- қызметкерлерді оқыту тиімділігі;
- тесттік жабу және тесттердің тиімділігі;
- бағдарлама сенімділігін қамтамасыз ету шаралары;
- БӨ талаптарында анықталған кемшіліктер саны және маңыздылығы;
- Бағдарламалық кодта анықталған ақаулықтар саны және маңыздылығы.

БӨ мүмкіндіктерін анықтаушы тенденциялар мысалдары:

бағдарламалық ақаулықтарды болжау және болжамдарды нақты деректермен салыстыру;

-егер өнімде қалған ақаулықтар туралы айтылса, онда ақаулықтарды бөлуді және сипаттамаларын болжау. Негіз ретінде тестілеудің сарапты бағалау деректері қолданылады.

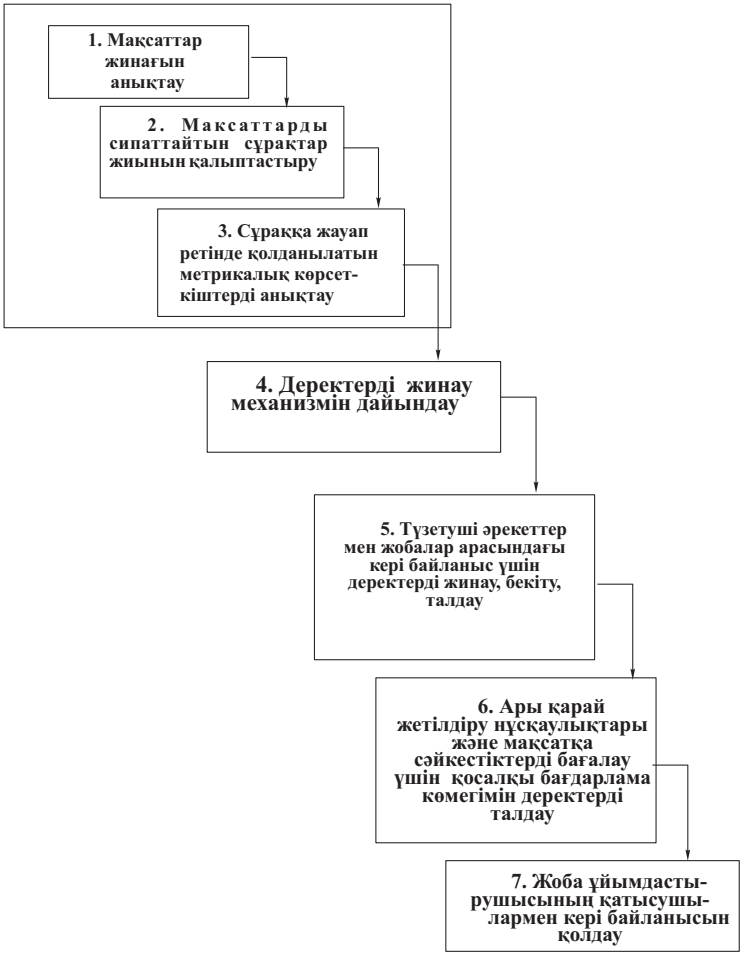
Метрикалар қолайлы, қарапайым, ұсынуға жеңіл, қымбат емес, түсінікті, тізбекті және жұмыс мерзімінде қолданылатын, деректерді жинау және өңдеу тарапынан ыңғайлы, барлық мүдделі пайдаланушылар үшін қол жетімді болуы керек.

Жиналатын деректер дұрыс (аталған көрсеткішті анықтау ережелеріне сәйкес жиналған), дәл, нақты және қайшылықсыз (өлшеу құралы немесе өлшеуді орындайтын маман ауысқанның өзінде бейнелетін шамаларда айтарлықтай айырмашылықтар жоқ) болуы қажет. Деректердің көп бөлігі нақты әрекетпен немесе уақыт мерзімімен ассоциацияланады. Олар аңдатпадан және барлық мүдделі тұлғаларды нақты орынмен және жинау уақытымен таныстыруға арналған уақыт және күн көрсетілімінен тұруы керек. Өлшеу процедурасы кез-келген орындаушы қажетті өлшеулерді қайта орындай алуы үшін анық сипатталуы керек.

5.3. Бейзили парадигмасы

5.3.1. Парадигманың жалпы сипаттамасы

Виктор Бейзили парадигмасы (Dr. Victor Basili) «мақсат, сұрақ, метрика» («Goal, Question, Metric», GQM) метриканы анықтауда кең қолданылатын және жақсы ұсынылатын, жобаны бақылауға және шешімдерді қалыптастыруға сәйкес болатын тәсіл болып табылады.



Сурет 5.2. Бейзили үрдісі

GQM пайдалану тәсілі мақсаттың метриканы таңдауға дейін қойылатынын болжайды. В.Бейзили жеті кезеңнен тұратын үрдіс негізіндегі әдістемені дайындады. Осыған байланысты БӨ дайындауда өлшемдерді жүйелі пайдалану мүмкіндігі пайда болды. 5.2 суретте парадигманың барлық кезеңдері көрсетілген. Олардың алғашқы үшеуі GQM парадигмасының шеңберінде жұмыс істеу мәнін анықтайды.

1 кезең. Мақсаттар жинағын анықтау. Бұл кезеңде бөлім жұмысына қатысты корпоративті мақсаттар немесе дайындамамен, сүйемелдеумен, өнімділікті және сандық көрсеткіштерді, үрдістерді арттырумен байланысты жобалық мақсаттар жинағын қалыптастыру орындалады (мысалы тапсырыс беруші өтініштерін қанағаттандыру, уақытылы жеткізу, сапаны жақсарту, қайта қолданылатын объектілерді құру, қайта пайдаланылатын тәжірибелі толықтырулар). Мақсаттарды анықтау «ақылдылар сарабы» әдісімен мәселені шешуде, сонымен бірге әрбір мүдделі тұлғадан талаптарды алу үрдісінде орындалады. Бұл мақсаттарды басымдылықтар есебімен қалыптастырып, БӨ жетілдіру салалары бойынша топтастыруға болады.

2 кезең. Мақсаттарды сипаттайтын сұрақтар жиынын қалып- тастыру. Әрбір мақсат жауабы мақсатқа жетуді анықтайтын сұрақты қалыптастыруға мүмкіндік береді. Аталған сұрақтар мақсаттарды сипаттайды, алға жылжуларды бағалайды, жету мезетін болжамдайды немесе мақсатқа жылжуды бағалауға арналған ынталандыру болып табылады.

3 кезең. Сұраққа жауап ретінде қажетті метрикалық көрсеткіштерді анықтау. Бұл кезеңде қойылған сұрақтарға адекватты жауаптар алуға не қажет екенін анықтайды.

5.3.2. 1 GQM кезеңі: мақсаттар жиынын анықтау

Мақсаттарды, сұрақтарды, метрикалық көрсеткіштерді анықтау

үшін мақсаттар, келешек үміттер, орталар жазылатын және мақсаттар құрылымын сипаттайтын шаблон дайындалды (кесте 5.3).

Мақсат — үрдісті ұсыну, бағалау, болжау, ынталандыру. (немесе өнім, модель, метрика және т.б). Мақсат үрдісті бағалауға, бақылауға, зерттеуге, игеруге, жетілдіруге мүмкіндік береді.

Ұсыну немесе сипат жиналған деректерді ұсынуға ыңғайлы объект моделін ұсынады.

Бағалау қолжетімді факторларға (дұрыс бағаланатын) негізделген тұрақты модельді дайындауға мүмкіндік беретін деректерге арналған шаблонның болуымен түсіндіріледі.

Кесте 5.3

Метрикалық көрсеткіштер, сұрақтар, мақсаттарды анықтауға арналған шаблон

Талдау (не талданады?)		Өлшенетін объект, үрдіс, өнім, модель, метрика және т.б
Мақсат	Қандай мақсатта (не үшін)?	Сипаттау, бағалау, болжау, ынталандыру, жетілдіру, ұсыну, бағалау, басқару, зерттеу, игеру немесе басқару
	Неге қатысты (фокус)?	Объект сапасына шоғырлану (өлшеу фокусталған) – шығындар көлемі, тиімділік, дұрыстылық, өнімді өлшеу, сенімділік, пайдаланушыларға қатысты достық қатынас.
Келешек	Келешек (аспекті қандай және оны кім орындайды?)	Объектіде өлшеуді жүргізетін пайдаланушылар: дайындаушылар, тапсырыс берушілер, жоба жетекшісі.
Орта	Мәнмәтін (сипаттар)	Өлшеу орындалатын орта: ресурс, үрдіс факторлары, адами факторлар, мәселелер, әдістер, құралдар, шектеулер және т.б.

Ынталандыру немесе *жетілдіру* объектіні немесе жағымды сапаны дұрыс ұсынуға мүмкіндік беретін нақты модельдің болуымен түсіндіріледі.

Мысалы: жақсарту мақсатында сүйемелдеу үрдісін бағалау.

Келешек (перспектива) — шығындар көлемін, тиімділікті, дұрыстылықты, ақаулықтардың, өзгерістердің болмауын, өнім өлшемдерін дайындаушы, менеджер, тапсырыс беруші тарапынан тексеру. Ақпараттың қолжетімділігі, оның «бөлшектену» деңгейі және дәлдігі сияқты аспектілерді әрдайым есепке алу керек.

Мысал: жоба жетекшісі тарапынан шығындар көлемін тексеру.

Орта— оқыту жүргізілетін мән-мәтінді анықтайды. Осыған орай, сипаттамалардың үлкен айырмашылықтары есебімен аталған жобаның жіктелуі, сонымен қатар, осы жобаға арналған сәйкес ортаны бөлу жеңілдетіледі, ұқсас сипаттамалар мен мақсаттардан тұратын жобалар класын анықтау қамтамасыз

етіледі, бұл оларды салыстыру үшін пайдалануға мүмкіндік береді. Ортаға үрдіс, қызметкер, мәселелік факторлар, әдістер, құралдар мен шектеулер жатады. Мәселелі факторларға адами факторлар, үрдіс, өнім, ресурс факторлары жатады.

Адами факторлар — пайдаланушылар саны, бағдарламалық инженеринг сараптамалары, топтарды ұйымдастыру мәселелері, бағдарламаны дайындаудың пәндік саласы, үрдісті орындау тәжірибесі және аспаптарды саралау.

Үрдіс факторлары — жобалауға арналған өміршеңдік модель циклын, әдістерді, әдістемелерді, құралдарды, бағдарламалау тілін таңдау.

Өнім факторлары — ішкі жеткізулер, жүйелер өлшемі және сапаның талапты деңгейі (мысалы, сенімділік, төзу қабілеті).

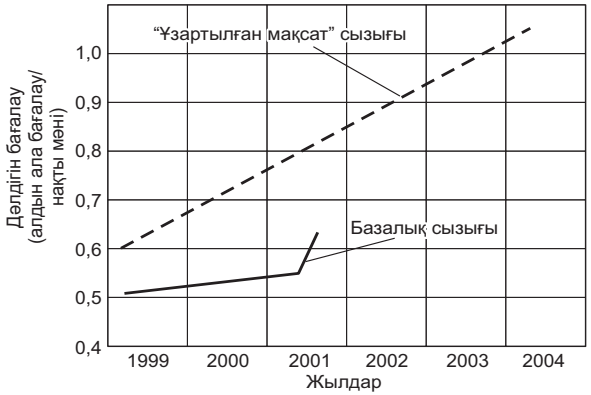
Ресурс факторлары мақсатты механизмдерде және жобалау механизмдерінде, күнтізбелік уақытқа, бюджетті құралдар көлеміне және қайта пайдалануға болатын бағдарламалық кодқа негізделеді.

Толық қалыптасқан мақсат мысалына келесі тұжырымда-ма жатады: «Жобаны дайындаушылар командасының тарапынан интерфейс күрделілігі есебімен және жобада қолданылатын объектілі-бағытталған дайындамада дайындалатын өнімді талдаңыз және бағалаңыз». Мақсаттар өлшенеді және бизнесте қабылданған модельдермен басқарылады.

Өлшенетін мақсаттарды анықтау үшін GQM парадигмасынан басқа басқа да механизмдер қызмет етеді, мысалы сапаны күшейту функциясы (QFD — Quality Function Deployment) және БӨ сапасын өлшеуде қолданылатын метрикаларды пайдалану әдістемесі (SQM — Software Quality Metrics).

GQM көрсетілгендей, мақсаттар сапаның түрлі модельдеріне сәйкес, әртүрлі шарттар үшін әрбір объект бойынша анықталады, олар әртүрлі көзқарас тарапынан және орта сипаттамасын анықтаумен пайдаланылады.

Стратегиялық мақсаттарды әдетте басқару үрдісінде қалыптастырады. Оларға жатады: өлшеулерді жүргізу кезінде негізгі бағыттарды қолдау, жылдам байланыстарды қамтамасыз ету, дайындау үрдісінде «іштей көзқарас» мүмкіндігін ұсыну; тарихи базада келешек жобаларды қалыптастыру. Стратегиялық мақсаттарды бағыттау бойынша алға жылжытуларды анықтауда қолданылатын метрика мысалына CMM-SEI моделін жетілдіру деңгейі жатады. *Тактикалық мақсаттарға* БӨ дайындаумен айналысатын ұйым және дайындаушылар командасы жатады: бағдарламалық инженерингті өткізу бойынша жігерлерді минимизациялау; шығын көлемін қысқарту; тапсырыс берушілердің өтініштерін максималды қанағаттандыру; ақаулықтардың пайда болу мүмкіндіктерін минимизациялау; сапалық көрсеткіштерді басқару және тестілеуді өткізу.



Сурет 5.3 «Ұзартылған мақсат» кескінінің мысалы

Ақаулықтарды азайту бойынша және сапаны басқару бойынша тактикалық мақсатқа жылжудағы алға жылжуды анықтауға арналған метрика мысалдары:

- қызметтік сұранымды бекітудің орташа уақыты;
- анықталған ақаулықтардың жиынтық көлемі;
- ақаулықтарды бекіту уақыты, бағалау және нақты уақыт

арақатынасы;

- ашық болып қалатын теңестірілген ақаулықтар;
- ақаулықтар көзін бөлу;
- ақаулықтардың пайда болуы арасындағы орташа уақыт.

Тактикалық, стратегиялық мақсаттарды графикте «ұзартылған мақсаттар» линиясы түрінде кескіндеуге болады, мұнда сапалы қолжетімді деңгей көрсетіледі. Осы мақсатқа бағытталу бойынша алға жылжу 5.3 суретте көрсетілген сызбада кескінделеді.

GQM алға жылжу мақсаттарын қалыптастырғаннан кейін мақсатқа жылжуды сипаттайтын алға жылжуды өлшеу тәсілдерін анықтау жағына ауысады.

5.3.3. Кезең 2 GQM: мақсатты сипаттайтын сұрақтар жиынын қалыптастыру

Мақсаттар абстрактылы деңгейде анықталады, ал сұрақтар орындалатын операциялар деңгейін жарықтандыру үшін қолданылады. Сұрақтарға жауап бере отырып, мақсатқа қол жеткізілгенін анықтауға болады. Мысалы, егер үрдіс мақсаты

келесідей қалыптасатын болса: «Осы мәселелерді түзетуге және талдауға қажетті циклдық уақытты қысқарта отырып, тапсырыс берушілер мәселелеріне жауап қайтару қабілеттілігін жетілдіру».

- Мәселелерді жоюға мүмкіндік беретін үрдіс құжаттарда көріне ме?
- Мәселелерді жою үрдісінде шолулар орындала ма және тестілеу жүргізіле ме?
- Мәселелерді жоюда қолданылатын бағалаулар дұрыс па?
- Барлық дайындаушылар таныс бақылау формасының өзгеруі бойынша механизм қолданыла ма?
- Мәселені анықтау үрдісінің нақты ұзақтығы қандай?
- Мәселені жою үрдісінің нақты ұзақтығы қандай?
- Анықтау және мәселені жою кезеңіндегі еңбек шығынының орташа көлемі қандай?
- Жойылған мәселелер тапсырыс берушімен анықталған мәселелердің жалпы санынан қанша пайызын құрайды?

БӨ ағымдағы нұсқасын жетілдіруге бағытталған мақсаттарды орындау жекелеген сұрақтардың пайда болуына ықпал етеді. Мысалы «Күрделі бағдарламалық жүйені жеңілдету» мақсаты келесі сұрақтардың пайда болуына себеп болады.

- Қолда бар бағдарламалық модульдердің саны қанша?
- Модульдерді байланыстыру деңгейі қандай?
- Модульді дайындау қанша уақыт алады?
- Әрбір модуль мүмкіндіктері бағаланды ма?
- Әрбір модульде анықталған қателер саны қанша?
- Әрбір модульдің өлшемі қандай?
- Модульдер өзара байланысқан ба?
- Модульдегі талаптарды жүзеге асыруды қадағалауға бола ма?

БӨ дайындауды басқару үрдісінде келесідей қағидалы сұрақтар туындауы мүмкін.

- Бұл әрекетті орындауға бола ма?
- Бұл үрдіс қанша уақыт алады?
- Шығындар көлемі қандай?
- Қанша жұмыскерді жұмысқа тарту қажет?
- Тәуекел деңгейі қандай?
- Болжамды мәмілелер қандай?
- Қанша қате пайда болуы мүмкін?
- Үрдісті жақсарту деңгейін өлшеуге бола ма?

Қойылған мақсаттарға жылжуды анықтауға мүмкіндік беретін сұрақтар спектрін анықтағаннан кейін, GQM үрдістің келесі кезеңінде осы сұрақтардың жауабына қолданылатын метрикалық көрсеткіштерді анықтайды.

5.3.4. Кезең 3 GQM: сұрақтарға жауап беруге қажетті метрикалық көрсеткіштерді анықтау

Аталған кезең сұрақтарға жауап беруге қажетті метрикаларды анықтау үшін, сонымен бірге үрдістің және өнімнің қойылған мақсаттарға сәйкестігін анықтауда қолданылады. Әдетте қажетті метрикаларды сұрақтардағы кілттік сөздер арқылы оңай алуға болады. Мысалы, «орташаландырылған түсірілген жігерлер» сөзі келесі сұрақта: «Тапсырыс беруші есептерінде белгіленген мәселелерді жоюға қажетті орташаландырылған түсірілген жігерлер қандай?» еңбек шығындарын бағалауға арналған метрикалық көрсеткішті көрсетеді, — жұмыс апталарында/айларында көрінетін еңбек шығындары. Одан басқа, метрика ретінде тапсырыс берушілер мәселелері туралы есептер санын пайдаланған жөн, олар апталық/айлық мерзімге жабылады.

Тағы мысалдар қарастырайық.

Мақсат келесідей қалыптасқан делік: «Ақаулықтар класы есебімен шекті өнімді сипаттау». Онда болжамды сұрақтардың біреуі: «Ақаулықтарды анықтау фазасы шегіндегі қателердің таралуы қандай»? Мұндай жағдайда келесі метрика қолданылады: талаптар қателері – олардың саны мен жіктелуі.

Басқа мысал. Мақсат: «Қойылған БӨ-ді ұсыну дәлдігі тарапынан, қайта пайдалану тиімділігі есебімен, сонымен бірге дайындаушылар командасы мүшелерінің тарапынан талдау».

Сұрақ 1: «Алдымен құрылмайтын модульдердің жалпы санынан қанша пайыз құрайды, мысалы, толығымен қолданылады немесе жекелеген қосалқы жүйелер түрінде?»

Метрика 1.1 (әрбір бағдарламалық модуль үшін): анықтау, модуль басынан бастап дайындалды ма? (ия, жоқ).

Метрика 1.2 (әрбір бағдарламалық модуль үшін): модульдің қатысы бар қосалқы жүйенің атауы.

Сұрақ 2: «Қайта және толығымен қолданылатын кодтар арасындағы пайыздық қатынас қандай (аяқталған жүйеде немесе қосалқы жүйелер бойынша)?».

Метрика 2.1 (әрбір бағдарламалық модуль үшін): анықтау, модуль қайта қолданылады ма? (ия, жоқ).

Метрика 2.2 (әрбір бағдарламалық модуль үшін): модуль құрамына кіретін қосалқы жүйе атауы.

Сұрақ 3 (код қайта қолданылатын әрбір бағдарламалық модуль үшін): «Түрлендіру кластарын қайта қолданудағы модульдерді бөлу қалай орындалады)?».

Метрика 3.1: кодтың өзгеру деңгей [өзгеріссіз, болжамды (20 % жолға өзгертілген); жойылған (20 % жолға өзгертілген)].

Метрика 3.2: бастапқы модульге арналған қайта пайдаланылатын атау және нұсқа.

Метрика 3.3: модуль құрамына кіретін қосалқы жүйе атауы.

Сұрақ 4: «Қайта қолданылатын модуль мен сенімділіктің қол

жетімді деңгейі арасындағы өзара байланыс қандай?».

Метрика 4.1 (барлық модульдер үшін): қайта пайдалану деңгейі.

Метрика 4.2 (бірнеше жаңылысулардан тұратын модульдер үшін): қайта пайдалану деңгейі.

Метрика 4.3 (жаңылысулардан тұрмайтын модульдер үшін): қайта пайдалану деңгейі.

Модульдер туралы деректерді жинағаннан кейін қайта пайдаланудың анықталған деңгейімен келесідей қорытындыларды жасауға болады:

- нәтижелері бекітілген модульдердің көп бөлігін қайта пайдалану,

модульдердің бастапқы үрдісіне қарағанда, қателер жиілігінің аз болуына алып келеді;

- қайта пайдаланылатын модульдердегі қателер санын оларды

пайдаланғанға дейін тексеру керек, себебі бұл модульдер қарқынды оқытылады және тестіленеді. Метрикалар сұрақтарға тиісті жауап алуды қамтамасыз ететін сандық ақпаратты қамтиды.

Бір ғана метрика көмегімен бірнеше сұрақтарға жауап табуға болады. Мысалы, «Қарастырылатын мәселелердің жасы» деген метрика келесі сұрақтарға жауап алуға мүмкіндік береді: «Мәселені шешуге ресурстар жеткілікті жиналды ма?»; «Мәселелер қаншалықты шешімсіз қалады?».

Екінші жағынан бір сұраққа бірнеше метрикалар қолданылуы мүмкін. Мысалы, «Қарастырылатын мәселелердің жасы» және «Шешілген мәселелер саны» метрикаларын келесі сұраққа пайдалануға болады? «Тапсырыс беруші өтініштеріне жауаптық реакцияны қалай анықтауға болады?».

Ұйымдағы мақсаттарды, сұрақтарды және метрикаларды анықтағаннан кейін немесе жобалық ортада нақты метрикалық деректерді жинауға арналған механизмді анықтау керек.

5.3.5. Кезең 4 GQM: деректерді жинау механизмін дайындау

Мақсатты әрдайым зейін фокусында ұстау керек. Мақсатты өлшеулерге жатпайтын деректерді жинау пайда әкелмейді. Жиналған деректер ұйымның тарихи базасына жиналады және жұмыста тұрақты қолданылады. Жинау механизмі жұмысын дұрыс ұйымдастыру үшін келесідей сұрақтарды қарастыру қажет.

Деректерді жинауды кім жүргізеді? Жинауды деректерге «жақын» адам жинайды. Мысалы, еңбек шығындары туралы деректер жинағы БӨ дайындауда дайындаушылардың өзімен орындалғаны дұрыс. Ақпараттандырылу деңгейінің жоғары

болуына байланысты, олар жүргізілген өлшемдер туралы есепті құратын қызметкерлер категориясына жатпайды, объективті позицияны алады.

Деректер жинағын қашан жинаған дұрыс? Барлығы қандай деректер жинауына және олардың қалай пайдаланылатынына байланысты болады. Дегенмен «барынша ертерек және барынша көбірек» орнату соншалықты жаман емес. Егер деректер жинағын «көңіл-күй бойынша» енгізсе, онда күн сайын, кемінде аптасына бір рет жасау қажет. Деректерді ай сайын жинау сирек орындалады, себебі бұл деректердің қай уақытта тиесілі екенін еске алу өте қиын.

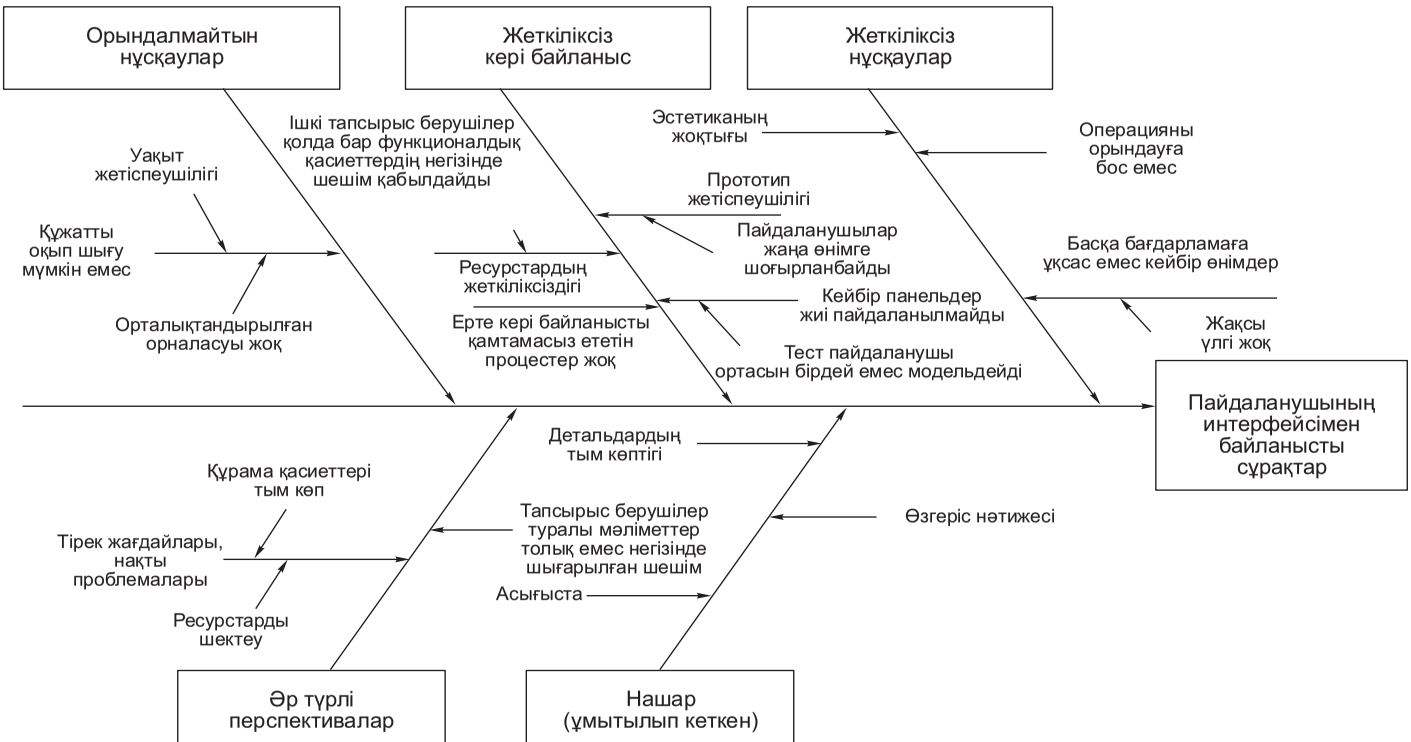
Деректерді жинауды нақты және тиімді ұйымдастыру қалай орындалады? Ең дұрысы автоматтандырылған құралдарды пайдалану. Деректерді жинау бойынша механизмдер жұмыста ыңғайлы болуы керек.

Метрикалармен кім тікелей жұмыс істейді? Барлығы мақсатты растауда сипатталған келешек үмітке немесе «көзқарасқа» тәуелді болады. Дегенмен жауапты кім алатыны соншалықты маңызды емес. Деректерді енгізген адам олардың қандай әдіспен түсіндірілетінін және онымен кім танысу керек екендігін білуі керек. Басқа жағынан алғанда, тек нақты деректермен жұмыс істеуді кепілдендіру қажет. Метрикалық деректерді жинау ыңғайлы және қарапайым болуы тиіс. Егер деректерді жинау және аналитикалық үрдіс мақсатты бағыттылықтан тұрады және бағдарламалық үрдіске автоматтандырылуы және біріктірілуі керек, онда жетістікке жетуге болады. Өлшеу үрдісінің деректерге бұрмалануды енгізбеуін қадалағау керек.

5.3.6. Кезең 5 GQM: түзетілетін әрекеттер мен жобалар арасындағы кері байланысты қолдауға арналған нақты уақыттағы деректерді жинау, растау, талдау

Жазбаларды қолмен жүргізу әдетте бұрмалануда, қателерден, қалдырып кетуден, кешігулерден пайда болады. Сондықтан деректерді автоматтандырылған жинау әдісін қолданған дұрыс: деректер процедурасын қиындатпау; қажет емес жазбаларды болдырмау; қызметкерді осы жерде қолданылатын процедураларға сәйкес жүргізуге оқыту; деректердің тарихи базасында жиналған барлық деректерді растау. Талдау үрдіі әртүрлі формалардан тұруы мүмкін, құралдардың көп бөлігін пайдаланумен және есептілік формасымен айрықшаланады. Метрикалық деректердің графиктік ұғымы әрдайым аналитикалық деректер түсінігін жеңілдетеді. Әдетте келесідей графиктік ұғымдар қолданылады:

қорытынды графиктер, гистограммалар, Ишикав диаграммалары(Ishikawa), Парето диаграммалары (Pareto) және шашыранды диаграммалар.

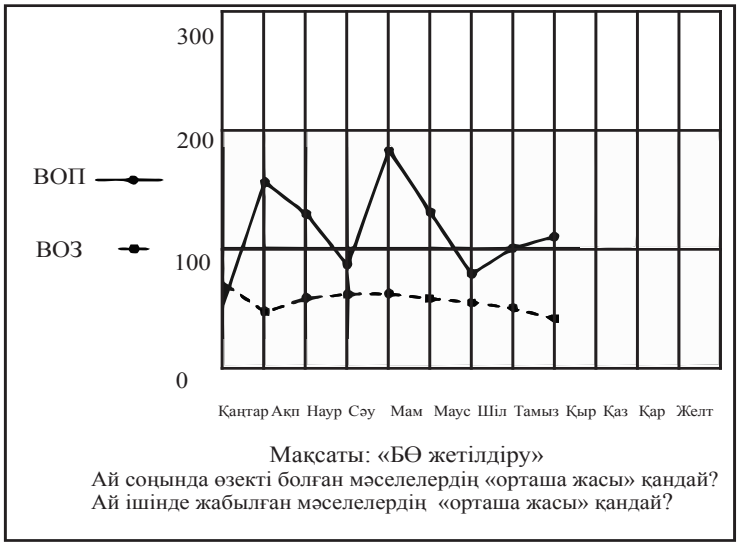

Орындалмайтын нұсқаулар
Жеткіліксіз кері байланыс
Жеткіліксіз нұсқаулар
Пайдаланушының интерфейсімен байланысты сұрақтар
Ішкі тапсырыс берушілер қолда бар функционалдық қасиеттердің негізінде шешім қабылдайды
Уақыт жетіспеушілігі
Құжатты оқып шығу мүмкін емес
Орталықтандырылған орналасуы жоқ
Ресурстардың жеткіліксіздігі
Ерте кері байланысты қамтамасыз ететін процестер жоқ
Эстетиканың жоқтығы
Прототип жетіспеушілігі
Пайдаланушылар жаңа өнімге шоғырланбайды
Тест пайдаланушы ортасын бірдей емес модельдейді
Операцияны орындауға бос емес
Басқа бағдарламаға ұқсас емес кейбір өнімдер
Кейбір панельдер жиі пайдаланылмайды
Жақсы үлгі жоқ
Құрама қасиеттері тым көп
Тірек жағдайлары, нақты проблемалары
Ресурстарды шектеу
Тапсырыс берушілер туралы мәліметтер толық емес негізінде шығарылған шешім
Детальдардың тым көптігі
Асығыста
Өзгеріс нәтижесі
Әр түрлі перспективалар
Нашар (ұмытылып кеткен)

Ишикав диаграммасының мысалы (себебі және салдары шырша ретінде көрсетілген) 5.4 суретте бейнеленген.

Бір орында (бір бетте) мақсаттарды, метрикаларды, аналитикалық графиктерді (сурет 5.5) біріктіру мүдделі тұлғаларға метрикаларды ұсынуға арналған әмбебап сипатты әдіс болып табылады. Мысалы, БӨ дайындамасымен айналысатын компания мақсатқа жетуге бағыттылған сұрақтардың келесі жинағын және үрдісті қадағалау метрикаларын пайдалана алады: «БӨ-ді жетілдіру»

Сұрақ 1: «Осы айда қанша ашық мәселелер пайда болды?».

Метрика 1: жаңа мәселелер саны (NOP — New Open Problems) – өнім шығарылғаннан кейінгі ай ішінде БӨ-нің ағымдағы нұсқасындағы ашық мәселелердің жалпы санына тең.



Сурет 5.5. Мақсаттар, сұрақтар, метрикалар, аналитикалық графиктердің бір бетінде орналасу мысалы:

ВОП — айлардың ашық мәселелерінің «орташа жасы»;

ВЗП — ВОП —ай ішінде жабылған мәселелердің «орташа жасы».

Сұрақ 2: «Ай соңындағы өзекті (ашық және жабық емес) мәселелер саны қанша?».

Метрика 2: Ашық мәселелердің жалпы саны (TOP — Total Open Problems,) — ай соңында жабылмаған БӨ нұсқасындағы жаңа мәселелердің жалпы санына тең

Сұрақ 3: «Ай соңында өзекті болып қалған мәселелердің орташа жасы қандай?».

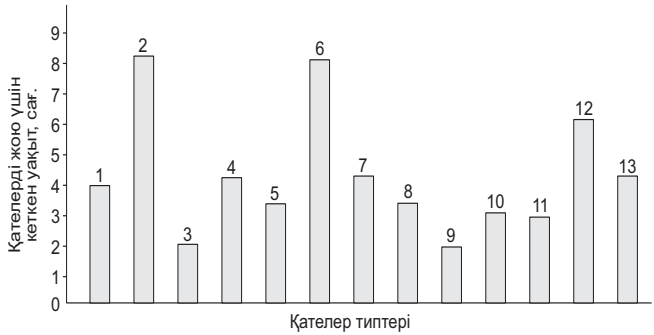
«Метрика 3: өзекті мәселелердің орташа жасы (AOP — Mean Age of Open Problems) — өнім шығарылымынан кейін, ай соңында жабылмаған, осы мәселелер санына бөлінген БӨ нұсқасындағы мәселелердің жалпы уақытына тең.

Сұрақ 4: «Ай ішінде жабылған мәселелердің орташа жасы қандай?».

Метрика 4: жабылған мәселелердің орташа жасы (ACP — Mean Age of Closed Problems) — өнім шығарылымынан кейін, ай соңында жабылған, осы мәселелер санына бөлінген БӨ нұсқасындағы мәселелердің жалпы уақытына тең.

5.3.7. Кезең 6 GQM: мақсаттарға сәйкестікті және ары қарай жетілдіру нұсқаулықтарын бағалауға арналған қосалқы бағдарламаны пайдаланып деректерді талдау

Өңделмеген метрикалық деректер әдетте түсінуге қиын. Мысалы, егер 150 жазбадан тұратын күні, уақыты, қате коды белгіленген қателер туралы тіркеу хабарламасы алынса, бұл деректерді түсіндіру қиын.



Сурет 5.6. Гистограмма көмегімен деректер алу мысалы:

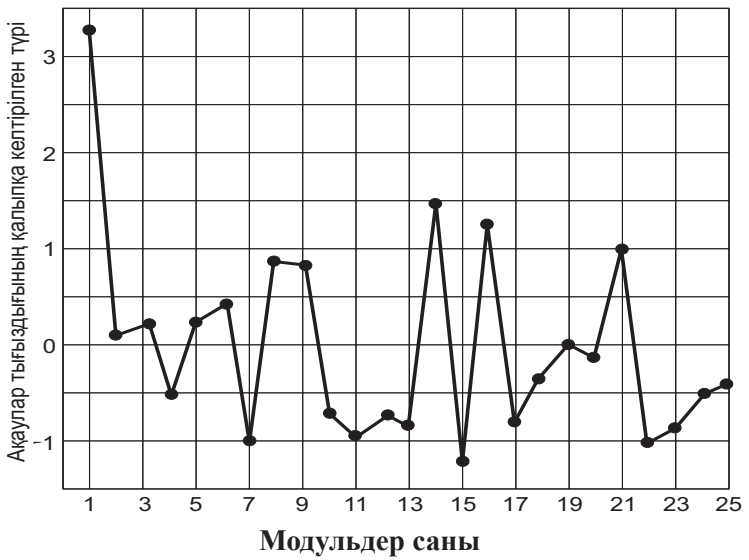
1 — логикалық қателер; *2* — есептік қателер; *3* — интерфейспен байланысты қателер; *4* — уақыт аралығын бөлумен байланысты қателер; *5* – операциялар арасындағы өзара әрекеттестікпен байланысты қателер; *6* — стандарттарға сәйкессіздік; *7* — бір мағыналы емес талаптар; *8* — қате шығыс деректері немесе олардың болмауы; *9* — қате кіріс деректері немесе олардың болмауы; *10* — деректерді дұрыс емес жүктеу; *11* — қате шақырылған ішкі бағдарламалр; *12* — деректерді өңдеумен байланысты қателер; *13* — құжаттамадағы қателер.

Дегенмен уақыттың белгілі мерзімінде әрбір типтегі қателердің пайда болу санын есептеу жүргізіледі, бірақ бұл деректер белгілі мағынада толтырылады. Оларды мақсаттың жету, жетпеуіне қатысты шешім қабылдауда қолдануға болады. 5.6 суретте гистограмма көмегімен әрбір типтегі қатені графиктік елестетуге болады.

5.3.8. Кезең 7 GQM: жоба ұйымдастырушыларының қатысушылармен кері байланысын қолдау

Кері байланысты қолдау —бұл өлшеуді орындау үрдісіндегі маңызды қадам. Жобаның барлық қызметкерлері және басқа пайдаланушылар, деректерді жинаған қызметкерлер өз жұмыстарының нәтижелерін тексеру мүмкіндігіне ие болуы керек.

Мысалы, егер дайындаушылар мен тестілеушілер тексерілген модульдерде қателерді бөлу жиілігі туралы ақпаратқа ие болса (сурет 5.7), онда олар белгілі бір қорытындылар жасай алады немесе кемінде сұрақтар қатарына сүйенеді. Бірінші модульде неге қателер саны көп? Бұл жұмыс түрінде жаңадан келген сарапшылар бұған себеп бола ма? Мүмкін бағларламашы бағдарламаның пән саласымен немесе тілімен жеткілікті таныс емес пе?

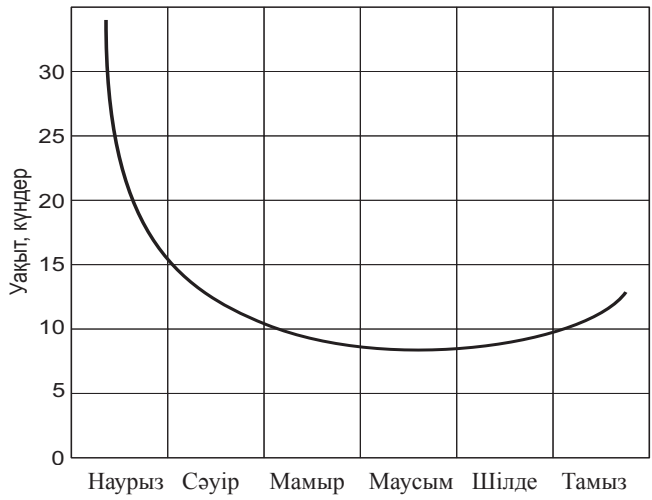


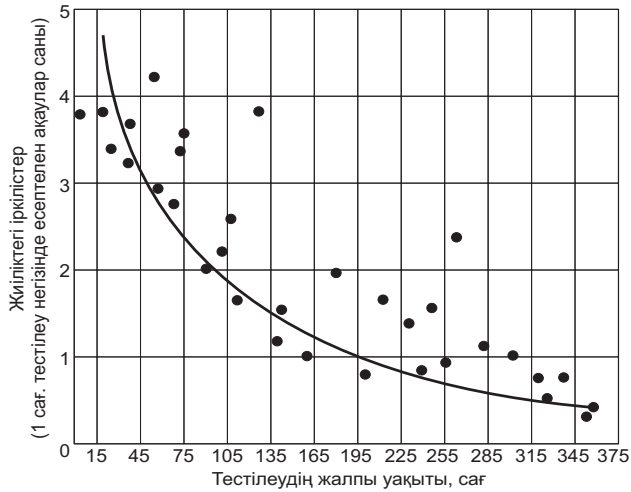
Сурет 5.7. Тексерілген модульдердегі қателерді тарату жиілігінің графигі

5.7 суретте көрсетілген талдауды үрдісті жетілдіру үшін пайдалануға болады. Егер дайындаушылар мен тестілеушілер, шолу деректерін және тесттік зерттеулерді жинаушы қызметкерлер алынған нәтижелермен таныспайды, олар үрдісті жетілдіру жолдарын талқылауда өз ойларын жеткізу мүмкіндігіне ие болмайды, сонымен қатар, нақты деректері бар жұмысқа арналған ынталандыруды да алмайды. Кері байланыстың әмбебап тәсіліне бір «бетте» мақсаттардың, сұрақтардың, метрикалар мен графиктік нәтижелердің 5.4 суреттегідей бейнелеуі жатады. Кері байланыс механизмдерінің графиктік кескіндерінің басқа мысалдары 5.8-5.12 суреттерде көрсетілген.

Қателерді тарату жиілігі туралы ұғым өте пайдалы, себебі көп қателерден тұратын модульді ары қарай талдау мақсатында бөлуге мүмкіндік береді. Көбінесе талдау үрдісі негізгі себепті анықтаудан басталады. Модульде қателер санының көп болуы оның күрделі болуынан ба? Анықталған кемшіліктер аталған бағдарламаға тиісті белгілі типтегі қателерге жатпайды ма (мысалы, жобаның бастапқы кезеңіндегі қателерге)? Егер бұл солай болса, онда бағдарламашы квалификацияны арттыру курстарынан өтуі керек пе? Егер осы модульдердің біреуін келешекте қайта пайдалану жоспарланса, ал ондағы қателерді бөлу жиілігі аз болса, бұл жоспарды қайта қарастыру қажет пе?

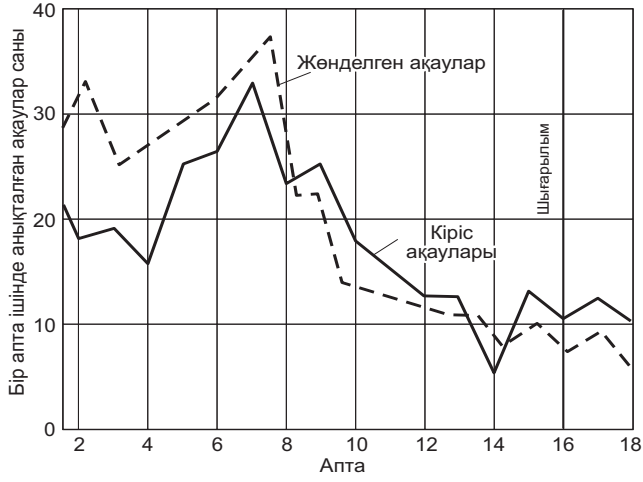
Егер негізгі кемшілік түзетілуі тиіс орташа уақыт белгілі болса және анықталған тарихи деректер негізінде күтілетін қателер санын анықтау мүмкіндігі болады,





Сурет 5.9. Уақыт озуымен кемшіліктер санының өзгеруі

онда осы кемшіліктерді түзетуге қажетте құралдар көлемі туралы қорытынды жасау негізделеді. Егер негізгі кемшіліктерді түзетуге бағытталған еңбек шығындарын алдымен азайып, одан кейін қайтадан көбейсе (5.8 суреттегідей), онда бір нәрсе дұрыс орындалмай жатыр.



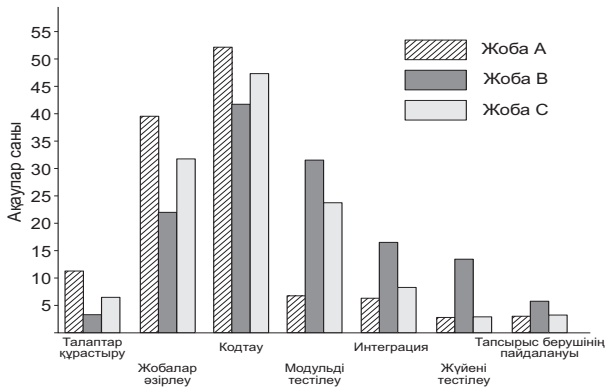
Сурет 5.10. Ақауларды анықтау және оларды уақыт өте жою

Сурет 5.11. Парето диаграммасы:

1 — есептеу қателері; *2* — стандарттарға сәйкессіздік; *3* — деректерді өңдеумен байланысты қателер; *4* — уақыт аралықтарын бөлумен байланысты қателер; *5* — құжаттамадағы қателер; *6* — әртүрлі талаптар; *7* — логикалық қателер; *8* — қате шығыс деректері немесе олардың болмауы; *9* — операциялар арасындағы өзара әрекеттестікпен байланысты қателер; *10* — деректерді қате жүктеу; *11* — қате шақырылған қосалқы бағарламалар; *12* — қате кіріс деректері немесе олардың болмауы; *13* — интерфейспен байланысты қателер.

Құрылым энтропиясының өсуіне байланысты күрделілік те артады және құраушыны әлдеқайда қарапайым қылу керек.

Тестілеу ортасында қателер анықталады және бекітіледі, регрессиялық тесттер асығыс орындалады. Регрессиялық қабықша орындалған кезде 5.9 суретте көрсетілгендей, бірнеше қателердің пайда болуы күтіледі. Егер жаңылысулар саны азаймай, керісінше көбейетін болса, онда дайындаушылардың мәселелерді шешумен емес, олардың өзімен айналысып жатқандығын білдіреді.



Сурет 5.12. Ақаулардың жинақталу кезеңін анықтауға мүмкіндік беретін диаграмма

Жоба басшысы тестілеуді тоқтатып, аталған тенденцияны талдай алады. Егер жаңылысулар саны қысқартылса, маңызды қателер қалса, онда компания басшылығы, жоба жетекшісі және тапсырыс беруші БӨ шығару бойынша келісімге келуі мүмкін. Тенденцияны талдаудың бұл әртүрлілігі тестілеу кезеңімен шектелмейді. 5.10 суретте көрсетілгендей, ақауларды қадағалау бірінші статистикалық шолулар пайда болғанға дейін басталуы мүмкін.

5.11 суретте көрсетілген Парето диаграммасы жобадағы қателер көзін жетекшімен анықтауға мүмкіндік береді. Егер 80 % мәселе 20 % кодта жасырынған деген жиі кездесетін түсінікке сүйенетін болсақ, онда қандай 20 % туралы айтылып жатқанын анықтау керек. Паретоның реттелген диаграммасы - осындай шешімді қабылдаудағы мінсіз көмекші. «Өміршеңдік маңызды» түсінік «тривиалды көпшіліктен» ажыратылғаннан кейін, ары қарай талдау егжей-тегжейді анықтауға мүмкіндік береді.

Егер БӨ дайындаудың өміршеңдік циклының қай фазасында ақаулықтар жиналғаны белгілі болса, онда соған көп көңіл бөлінеді. Сол жерде шолулар мен тексерулер жүргізіледі, мәселені шешуге тәжірибесі мен білімі жеткілікті компаниядағы басқа жоба өкілдері қосылады. Бір жобаның нәтижелері аномалия ретінде қарастырылуы мүмкін, бірақ егер бірнеше жобалар бір «күдікті» кезеңді көрсететін болса, мысалы 5.12 суретте көрсетілгендей кодтау кезеңінде, онда компания басшылығы дұрыс шешім қабылдауға мүмкіндік беретін анық түсініктер алады.

5.4. Негізгі метрикалық көрсеткіштер жинағы

5.4.1. Метрикалық көрсеткіштердің негізгі көздері

Бағдарламалық жобаны басқаруда деректердің көп таралған үш көзін белгілеуге болады: еңбек шығындары; шолулар; өзгертуге сұраным. Бұл үш базис бір-бірімен өзара әрекеттеседі және бірыңғай негіз құрады. Оларды метрикалық көрсеткіштердің тәжірибелік жүйесін қалыптастырудағы алғашқы қадам ретінде пайдалануға болады.

Басқару ең алдымен жоба шеңберінде еңбек шығындарын, сонымен қатар еңбек шығындарының сипаттамасын талап етеді; екіншіден шолу үрдісінде анықталған қателер мен ақаулардың сандары және типтері туралы мәліметті талап етеді; - сұраудың жүйенің бастапқы талаптарын өзгертуі туралы сұралатын

өзгерістер талап етіледі және жобаның алынған құраушысы ашылған кемшілік салдарынан өзгереді.

5.4.2. Еңбек шығындары

Еңбек шығындары туралы көбінесе БӨ дайындау үрдісінде жұмсалатын ресурстар айтылады. Аппараттық қамтамасыз ету, телекоммуникация, бағдарламалық құралдар және басқа бағалы мүлік БӨ дайындау бюджетіне сирек ықпал етеді. Бағдарламаларды дайындауда қымбат тұратын құрылғы талап етілмейді, қойма мекемелері және білдектерге қажеттілік жоқ. БӨ құру шығындары адами факторға тәуелді. Мұндай жағдайда қызметкердің немен айналысатыны (барлық құрылымдағы «ең қымбат» элемент), жоба дайындауға қатысты бағалы ақпараттың өңделуінің орындалуы, жоба бойынша жұмыстың қалған бөлігіне бюжеттің қалай бөлінетіні туралы ұғым құру талап етіледі. Дайындау үрдісін осы және келешек жобалар үшін жетілдіру, дайындаушылардың өніміділік деңгейін арттыру, келешек жобалардың құны туралы болжау құру қажет.

Еңбек шығындары туралы деректерді жинау келесі ұқсас жобаны бағалауда маңызды болып табылады; бұл деректер басшылыққа көптеген шешімдер қабылдауға мүмкіндік береді.

БӨ дайындауды шығындардың нақты көлемін бағалау үшін дайындау үрдісінде орындалатын әрбір тапсырма үшін шығындар көлемін қадалағау керек.

Еңбек шығындарын көрсететін деректерді жинау қағидаларымен танысу басшылыққа қандай фазалар аз еңбек шығындарын талап ететіні туралы, жекелеген тапсырмаларды шешуге жұмсалатын уақыт көлемі, жұмыс орындауға қажетті қосымша уақыт, міндетті тәртіпте орындалуы тиіс өндірістік емес қосымша жұмыс көлемі туралы түсінікті алуға көмектеседі.

Қызметкердің есебін қалыптастыру жүйесі және еңбек шығындары туралы мәліметтер автоматтандырылуы керек, ал қызметкер көп уақытын есепті мерзімді толтыруға жұмсамауы керек.

5.4.3. Шолулар

Шолу үрдісінде ақпаратпен байланысты жазбаны орындайды: қателер саны, әрбір қатенің түрі, аталған қатені тудырған фаза, осы қате пайда болған фаза туралы.

Жобаны орындауды бақылауда қолданылатын бағалы метрикалық көрсеткіштердің біреуіне әрбір өнімді шолуға жұмсалған уақыт жатады. Егер шолулар ерікті жүргізілетін болса, ал уақыт нақты қадағаланса, онда басқа жобалардан

команда мүшелерінің сараптамасын өткізуге болады. Онда тәжірибемен алмасу орындалады, себебі аталған командадағы сарапшылар өзара пайдалы бастауларда ұқсас жұмыстарды, өзара оқытуды, қолдауды орындай алады.

Әрине бұл шекті БӨ-нің сапасына әсер етеді. Жұмыстың осындай түрінде хронометраж жүргізу БӨ шолу үрдісі күрделі болған сайын, оларды қайта өңдеуге жұмсалатын уақыт та аз талап етіледі деген қорытындыға келу керек. Осындай шолулар құралдарды үнемдеуге және өнім сапасының жақсаруына алып келеді.

Қай кезеңде қиындықтар туғанын білу маңызды. Егер мәселелер дайындаудың өміршеңдік циклының соңғы кезеңдерінде пайда болса, бұл оларды жоюдың жоғары шығындарын талап етеді, себебі қайта жасау көлемі артады. БӨ шығарғаннан кейінгі мәселелерді әдетте тапсырыс беруші анықтайды. Шолулар қиындық түрлері туралы ақпаратты алуға мүмкіндік береді. Бұл жағдайда мәселелердің толық категорияларын шығару немесе олардың ықпалын азайту керек. Шолуды орындау үрдісінде тез және аз шығындармен түзетуге болатын мәселелер анықталады.

Мәселелерді қателер мен ақауларға бөлуге болады. *Қате* туылу кезеңінде анықталған мәселе ретінде жіктеледі. *Ақау* оның пайда болуы кезеңінде, дайындаудың өміршеңдік циклының соңғы кезеңдерінде анықталған мәселе ретінде саналады.

Қателерді жою арзан емес, бірақ ақауларды жою одан сайын қымбат. Өміршеңдік циклдың ерте кезеңдерінде, әсіресе талаптарды қалыптастыру кезеңінде, сонымен қатар жүйені тестілеу кезеңінде, өміршеңдік циклдың соңғы кезеңдерінде пайда болған ақауларды жою қымбат болып табылады.

Жобамен жұмыс істеуде тексеру, сараптамалық бағалау, өнім ақауларын құрылымдық толықсыз өлшеулер, яғни шолулар орындалады. Дайындаманың ерте кезеңдерінде ақауларды шолу жүргізу арқылы анықтау бағалы салым болып табылады және ұйым тарапынан салмақты сыйақыны талап етеді. Сонымен бірге, ақаулар статистикасы ақауды анықтау «себепшісімен» емес, шолумен ассоциациялануы керек екенін есте сақтау керек.

Әрбір қателік немесе ақау туралы ақпарат жазылады және бекітіледі: шолуды толтыру сәті; процедура қатысушылары туралы ақпарат; шолуға тиісті көрсетілім; қателер мен ақаулар, қателер мен ақаулар түрлері, пайда болу көзі, деңгейі, маңыздылығы туралы келесілген пікір. Одан басқа, БӨ қолжетімділігі немесе қайта шолуды өткізу қажеттілігі туралы жазба орындалады.

Шолу нәтижелері келесідей маңызды сұрақтарға жауап

беруге көмектеседі:

- Кім сапалы бағалауды өткізеді және ол қандай рөл атқарады?
- Қандай кезеңдер мәселелердің көп санымен ерекшеленеді?
- Қандай кезеңдерде маңызды мәселелер анықталған?
- Қателер мен ақаулар түрлері қандай?
- Қатенің қандай түрі жиі анықталады?

5.4.4. Өзгертуге сұраным

Өлшеуге арналған деректердің маңызды көзі бағдарламалық жүйенің өзгеруінде қалыптасады. БӨ жеткізгеннен кейін дайындаушылар қадағалау жүйесінің көмегімен жетіспеушіліктер жазбасын немесе өгертуге сұраным жазбасын орындайды және оларды жоюға күш салады. Стандарттар мен технологиялардың Ұлттық институтының материалдарына сәйкес (NIST — National Institute of Standards and Technology) өзгертуге сұранымның келесідей түрлерін бөледі:

түзетуші — кемшіліктерді түзетуге бағытталған (қателер мен ақауларды);

бейімделуші— БӨ-нің жүйедегеі өзгерістерге бейімделуіне, өзгертілетін ортаға бейімделуіне (сыртқы жағдайларға) сапалы өлшемдер мен жақсартуларды енгізусіз бағытталған;

ескертпелі — кемшіліктер пайда болғанға дейін ескертуге бағытталған;

жетілдірілген— ары қарай сүйемелдеуді жеңілдету мақсатында қолдауға бағытталған.

Мәселелер БӨ жеткізгеннен кейін сүйемелдеу кезеңінде ғана емес, сонымен бірге дайындау кезеңінде пайда болуы мүмкін. Олар шолуларды (статикалық тесттер) және тесттерді орындау кезінде (динамикалық тесттер) анықталуы мүмкін. Мәселелерді анықтаудың басқа да әдістері бар, мысалы БӨ дайындаушының өзімен игеруі немес пайдаланушының мәселермен танысуы.

Өзгерістер енгізу қажеттілігі әдетте жүйе пайдаланушысымен пайда болады, бірақ бастама бақылаушы тарапынан болуы да мүмкін.

Әдетте өзгерістер енгізу қажеттілігін негіздейтін ақпарат келесідей пунктерді қамтиды:

- өзгерістер қажеттілігі туралы айтатын тұлғаны көрсету;
- мәселені немесе қажеттіліктерді сипаттай (бағдарламалық өнімдердің аномалиясы бойынша IEEE стандарты — IEEE Standard on Software Anomalies — болжамды мәселелер категориясының толық көлемін қамтиды);
- мәселенің қайда және қашан пайда болғандығын көрсету

немесе қате туралы хабарлама;

- жүйенің немесе қосалқы жүйенің атауы (егер белгілі болса);
- аталған сұранымның сынамалығы туралы көрсетілім (алдын-ала анықталған маңыздылық деңгейлеріне негізделеді);
- аталған өзгерістерді түзету туралы, оның өнімді жетілдіру үрдісіндегі орны, бейімделгіштігі немесе осыған байланысты сапалы көрсеткіштердің артуы туралы көрсетілім;
- аталған өзгерістің ықпалы туралы хабарлама.

Өзгерістер туралы сұранымға жауап беру келесі ақпаратпен сүйемелденеді:

- жүйедегі қандай орын өзгертіледі және мәселе қай жерде?
- мәселе пайда болған дайындау кезеңі;
- тесттердің қажетті жинағы;
- өзгерістер енгізуге қажетті еңбек шығындарының бағаланған саны (сағаттар саны);
- өзгерістер енгізу қай кезде (күнтізбелік уақыт көрсетілімімен) күтіледі?

Өзгерістерді орындағаннан кейін деректердің қорытынды жинағына мәселені анықтау және жою, жүргізілетін жетілдіру бойынша ақпарат жатады. Сонымн бірге көрсетіледі:

- өзгерістерді енгізу мерзімі;
- түсірілген күштердің нақты көлемі;
- өзгеріс қай жерде жүргізіледі (жүйе шегінде)?
- тестілеу нәтижелері.

Әдетте БӨ өзгерісі үрдісінде келесідей мақсаттар қойылады:

- БӨ өзгерту бойынша сұраным жауаптарына арналған цикл уақытын қысқарту;
- Бағдарлама сапасын арттыруға қажетті өзгерістер санын азайту;

Аталған мақсаттарға жетудегі үрдістер деңгейін анықтау мақсатында келесідей сұрақтар жиі қойылады:

- сұралатын өзгерістер мәні неде?
- өзгерістер қателерді жою үшін қолданылады ма әлде жетілдіруді енгізу үшін бе?
- мәселенің құрамдас құраушылары қандай?
- талаптар қаншалықты жиі өзгертіледі?
- мәселе қаншалықты шешілмей қалды?
- есепте қандай маңызды мәселелер бар
- бұл кезеңде өзгерту қажеттілі туады ма?

Өзгерістерді енгізу кезінде оларды жүзеге асыру бойынша қажеттілік туындайды. Жоспарла утиісті өзгерістерді анықтаудан, өлшеу үшін БӨ модулін таңдаудан, конфигурацияны

бақылау есебімен БӨ нұсқасын таңдаудан, қандай атрибуттарды жұмысты орындағаннан кейін өлшеу керек екендігін анықтаудан тұрады. Жоспарлау сонымен бірге жиналған деректерді өңдеуге арналған процедураларды анықтаумен және өлшеу нәтижелерінің есебімен, барлық процедура құжаттамасын жүргізумен түсіндіріледі, ал қажет болған жағдайда мүдделі тұлғалар шеңберінен пайдаланушыларды оқыту орындалады.

Бақылау сұрақтары:

1. Өлшеу мақсаты қандай: а) үрдістің; б) бағдарламалық өнімнің; в) жобаның?
2. «Метрика» түсінігіне анықтама беріңіз.
3. Метриканың қандай топтары мен типтерін білеңіз?
4. Метрика мысалдарын келтіріңіз: а) бағдарламалық өнімге; б) үрдіске; в) жобаға.
5. Атрибуттарды өлшеу мақсаты қандай: а)бағдарламалық өнімнің; б)жобаның; в) үрдістің?
6. Қандай негізгі тапсырмалар дайындаушыларға алынған метрикаларды шешуге мүмкіндік береді?
7. Жиналған метрикалар қай жерде сақталады?
8. Метрикаларды жинау және талдау бойынша қандай негізгі әрекеттер CMM-SEI моделінің бес деңгейінің әрқайсына сәйкес келеді?
9. Қандай өлшемдер орындалады: а)бағдарамалық талаптарды басқаруда; б)жобаны жоспарлауда; в)жобаның орындалуын қадағалауда және бақылауда; г)оқыту бағдарламасын жүзеге асыруда; д)бағдарламалық инженерингте; е)үрдісті сандық басқаруда?
10. Бейзили парадигмасының мәні неде?
11. Бейзили парадигмасының негізіндегі әдістеме қандай негізгі кезеңдерден тұрады?
12. Бейзили әдістемесінің әрбір кезеңіне қысқаша сипаттама беріңіз.
13. Қандай әдіспен орындалады: а)мақсаттар жинағын анықтау; б)мақсаттарды сипаттайтын сұрақтар жинағын анықтау; в)қойылған сұрақтарға жауап алуға қажетті метрикалық көрсеткіштерді анықтау; г)деректерді жинау механизмін дайындау; д)түзетілетін әрекеттер мен жобалар арасындағы кері байланысқа қажетті деректерлі талдау, растау, жинау; е)мақсаттарға сәйкестікті бағалауға арналған деректерді талдау; ж)жобаны басқаруға арналған кері байланысты қолдау.
14. Негізгі метрикалық көрсеткіштерді атаңыз.
15. «Қате» және «ақау» түсінігіне анықтама беріңіз және олардың арасындағы айырмашылықты түсіндіріңіз.
16. Өзгерту сұранымдарының негізгі түрлерін көрсетіңіз.

6 БӨЛІМ

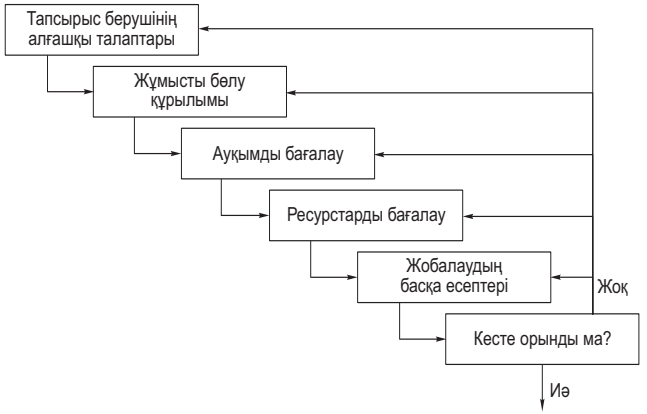
БАҒДАРЛАМАЛЫҚ ӨНІМДЕРДІ ҚҰРУ ЖҰМЫСТАРЫН ЖОСПАРЛАУ

6.1. Бағдарламалық өнімдерді құру жұмыстарын бөлу құрылымы

Жұмысты жоспарлау тапсырыс берушіден алғашқы талаптарды алудан басталады, ал жоспарлау негізіне жобаны сәтті орындауға және аяқтауға қажетті барлық тапсырмаларды бөлу және олардың арасындағы байланыстарды анықтау жатады. Оның нәтижесіне БӨ құру бойынша *жұмыстарды бөлу құрылымы* жатады.

Әрбір белгіленетін тапсырмалар мен құрылым элементінің еңбек сыйымдылығы және көлемі бағаланады, қажетті ресурстар мен өміршеңдік циклды орындаудың уақытша графигі анықталады. Жоспарлау үрдісі циклдық үрдіс ретінде анықталады: оның циклы 6.1 суретте көрсетілген.

БӨ-ді дайындау графигі орындау нақтылығы тарапынан бағаланады, қандай-да бір көрсеткіштер бойынша нақты емес графикті алу жағдайында жоспарлау циклы қайталанады. Жоспарлау кезеңіндегі барлық белгіленген тапсырмаларды орындауды қайталау әрдайым міндетті емес.



Сурет 6.1. Бағдарламалық өнімді құру бойынша жұмыстарды жоспарлау циклы

Жұмыстарды бөлу құрылымы тапсырмалар иерархиясынан

тұрады.

Тапсырмалар иерархиясындағы тексеруді әрбір тапсырма көлеміне және күрделілікке баға беруді жүргізуге жеткілікті деңгейге дейін орындау қажет. Тапсырмалардың әрқайсын қысқа мерзім уақытында жеке орындаушы орындауы үшін құрылымның төмен деңгейіндегі тапсырмалар шағын және қарапайым болуы керек.

Құрылымдауды құрылымды диаграмманы құрумен аяқтау керек, ол БӨ ары қарай жобалаудың ортақ тұжырымдамасын бейнелейді.

6.2. Бағдарламалық өнімнің күрделілігін және көлемін бағалау

БӨ көлемінің бірлігіне бағдарламалық код жолының саны қабылданады (LOC), ал өнімділік бірлігіне бір айда бір адаммен жүргізілетін (LOC/адам-мес) тиімді бағдарламалық код жолының саны (яғни БӨ-дегі бағдарламалық кол жолының саны) қабылданады.

Бағдарламалық кодты құрылымдаумен байланысты емес жекелеген жұмыстарды адам/сағат бірлігінде өлшеу керек.

Жұмыстарды бөлу құрылымының әрбір элементінің көлемі мен күрделілігі сараптық бағалау көмегімен анықталады және LOC санымен және адам/сағат бірлігімен өрнектеледі. Әрбір бағалауды алу үшін оның көрсеткішін орташаландырып, кемінде үш тәуелсіз сарапшыны пайдалану керек. Құрылымдық элементтің Д = 0,75...1,25 күрделілігі күрделіліктің салмақтық коэффициентімен есепке алынады. Құрылымдық элемент көлемін алу үшін оның сараптық бағалауын $Д_с$. күрделілік коэффициентіне көбейту керек.

6.3. Бағдарламалық жобаны орындауға қажетті техникалық, техникалық емес және қаржылық ресурстарды бағалау

Жекелеген құрылымдық элементтердің көлемі бойынша БӨ құру бойынша жұмыстың жалпы көлемі есептеледі (LOC және адам/сағат). Код көлеміне тәуелді БӨ шағын, аралық, орташа және үлкен деп бөлуге болады. 6.1 кестені пайдалана отырып, бағдарламалық кодты құруға арналған орындаушылардың қажетті саны анықталады.

Жоба орындаушыларының жалпы санын алу үшін бағдарламашылар санына адам/сағат шығыны бойынша анықталған адам саны қосылады.

Бағдарламалық өнім көрсеткіштері

Бағдарламалық өнім көлемі	Адамдардың еңбек сыйымдылығы, айлар	Өнімділік LOC/адам.-айлар	Құру мерзімі, айлар	Қажетті штат, адам
Шағын (2 KLOC)	5,0	400	4,6	1,1
Аралық (8 KLOC)	21,3	376	8,0	2,7
Орташа (32 KLOC)	91,0	352	14,0	6,5
Үлкен (128 KLOC)	392,0	327	24,0	16,0

Әрбір белгіленген құрылымдық элемент бойынша жұмыстарды бөлу аспаптық құралдарды (аппаратты және бағдарламалық) және болжамды қаржылық шығындарды талап ететін орындаушылар квалификациясын анықтайды. Одан кейін қажет болған жағдайда жеке ресурстарды пайдалану кезектілігін, әртүрлі құрылымлық элементтерді бөлу механизмдерін, дайындау мерзімі бойынша шектеуді анықтайды.

6.4. Бағдарламалық өнімді құрудағы болжамды тәуекелдерді бағалау

Дайындау үрдісінде пайда болатын тәуекелдер ресурстық, қаржылық, ұйымдық (әкімшілік) қамтамасыз етумен байланысты және үлкен көлеммен, БӨ күрделілігімен байланысты болып бөлінеді.

Ресурстық тәуекелдер ресурстардың бағалануын талдау кезінде және олардыңң пайдаланылуын жоспарлау кезінде анықталады. Мұндай тәуекелдер тиісті квалификациядағы қызметкердің жетіспеуімен, аппаратты қамтамасыз ету өнімділігінің жетіспеуімен, бағдарламалық аспаптық құралдардың сәйкессіздігінен немесе кейбір қосымша қажеттіліктерге қаржыландырудың жетіспеуінен туындайды.

Қаржылық тәуекелдер ресурсты тәуекелдермен және үлкен көлеммен, БӨ күрделілігімен тығыз байланысқан, ресурстарды қате жоспарлау жоба бюджетінің артуына алып келеді. Одан басқа, қаржылық ресурстарға БӨ нарығы жағдаятының өзгеруі

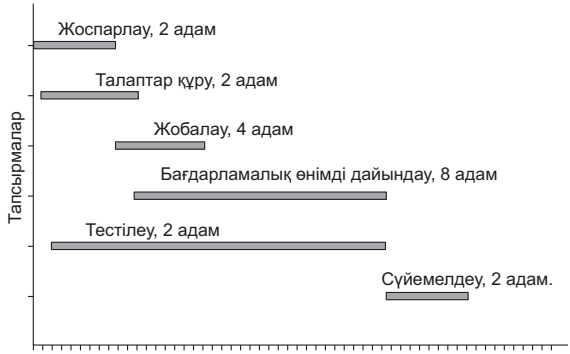
және тапсырыс берушінің қаржылық жағдайы әсер етеді.

Ұйымдық немесе әкімшілік тәуекелдер жобаны дайындауды дұрыс ұйымдастырмаумен, жоспарлаудағы қателермен, міндеттерді бөлумен, орындаушылардың жауапкершілігінің жеткіліксіз болуымен түсіндіріледі.

Соңғы түрдегі тәуекелдер БӨ күрделілігін және көлемдерін бағалау дәлсіздігінен туындайды. Азайтылған алдын-ала бағалаулар ресурстардың қажетті көлемін қате анықтауға және жоба жұмысы мерзімінің тоқтауына алып келуі мүмкін. Болжамды тәуекелдерді анықтағаннан кейін, олардың пайда болу ықтималдылығының сарапты бағалауы жүргізіледі және оларды жою тәсілдері жоспарланады.

6.5. Бағдарламалық өнімді орындаудың уақытша графигін құру

Жобаны орындаудың уақытша графигін құру үшін ресурстар мен көлемдердің бұрын алынған бағалауларын, жұмыстың жоспарланған көлемдерін, құралдық ресурстарды талдау және өміршеңдік цикл фазалары бойынша персоналды бөлу қажет. Мұндай бөлу осындай жоспарлаудың тарихи тәжірибесі негізінде орындалады. Егер тәжірибе болмаса, онда 6.2 кестені пайдалануға болады. Жобаның уақытша графигін құру дайындама кезеңдерінің GANTT-диаграммасын құрудан басталады, оның мысалы 6.2 суретте көрсетілеген. Шағын жобаларды жоспарлауда оны қолмен орындаған жеңіл болып келеді. Генри Гант(Henry Gantt) авторының атымен аталған диаграммада кезеңдердің кезектілігі, олардың уақыт бойынша тізбектілігі, жобаны орындаудың шекті мерзімі жақсы көрсетілген.



Сурет 6.2. GANTT-диаграммасының мысалы

Бағдарламалық өнімді дайындаудың негізгі кезеңдері бойынша уақытша шығындарды және еңбек шығындарын бөлу

Кезең	Жоба көлемі, %			
	шағын (2 KLOC)	аралық (8 KLOC)	орташа (32 KLOC)	үлкен (128 KLOC)
Еңбек шығындары				
Жоспарлау, талаптарды құру, жоғары-деңгейлі жоспарлау;	16	16	16	16
Егжей-тегжейлі жобалау	26	25	24	23
Дайындау	42	40	38	36
Тестілеу, сүйемелдеу	16	19	22	25
Уақытша шығындар				
Жоспарлау, талаптарды құру, жоғары-деңгейлі жоспарлау;	19	19	19	19
Егжей-тегжейлі жобалау Дайындау	63	59	55	51
Тестілеу, сүйемелдеу	18	22	26	30

Әрбір кезең үшін оны қанша адам орындайтыны, оның ұзақтығы, басталу және аяқталу мерзімі көрсетіледі. Көптеген үлкен жобалар үшін диаграмманы құру қиын. Мұндай жағдайда Microsoft ProjectTM сияқты автоматтандырылған құралдарды пайдалану ұсынылады, ол ресурстар мен жұмыстар кезеңін бөлу құрылымын тереңірек қарауға, жеке жұмыстар арасындағы өзара байланысты есепке алуға, қызметкерді рационалды бөлуге, аса жүктелуді және тұрып қалуды болдырмауға мүмкіндік береді. Үлкен жобалар үшін жобаны орындаудың жалпыланған графигін (қолмен құрылады) ұсыну нұсқаланады, бұл графикте 6.2 суреттегідей негізгі фазалар белгіленеді. Егер жұмыстарды бөлудің толық құрылымы кестемен ұсынылса, жоба жоспарында GANTT- диаграмманың болмауына жол беріледі.

6.6. Жиналатын метрикалар, пайдаланылатын әдістер, стандарттар мен шаблондар

Жоспарлау кезеңінде жобаның жоспарын құруға жұмсалған ресурстарды бағалауды орындау қажет (жоспардың бекітілген нұсқасын алуға қажетті уақыт, жоспарлауға қатысатын адам саны, жоспардың компьютерлік нұсқасын және оның парақтың көшірмесін құру уақыты, жоспарды бекіту және растау процедурасының ұзақтығы).

Барлық алынған деректерді жобалық топтың тарихи деректер базасында сақтау керек (ТДБ). Одан басқа ТБД-на жұмыстардың жоспарланған көлемдерін және БӨ өміршеңдік циклының кезеңдері бойынша ресурстарды бөлуді, осы кезеңдер ұзақтығын, жоба жетекшісінің пікірі бойынша барлық басқа деректерді енгізу қажет, бұл жоспарлау үрдісін ары қарай жақсартуға және оның өнімділігін арттыруға көмектесуі мүмкін.

Пайдаланылатын құралдар: құжаттарды дайындау жүйесі (мысалы, MS Word); электронды кестелер (мысалы, Excel); жоспарлауды автоматтандыру жүйесі (мысалы, Project).

Пайдаланылатын әдістер мен стандарттар: ұйым үрдісі; ұйымның метрикалық бағдарламасы.

Қолданылатын шаблондар: жоба жоспары; шолу бойынша есеп; жоба мәртебесі туралы есеп.

Бақылау сұрақтары

1. Бағдарламалық өнімді дайындаудың өміршеңдік циклындағы жоспарлау кезеңінің мақсаты қандай?
2. Жоспарлау циклы дегеніміз не?
3. Жұмысты құрылымдау мақсатын және қолданылу бағытын түсіндіріңіз.
4. Бағдарламалық өнім көлемі мен күрделілігі қалай өлшенеді?
5. Жұмысты орындауға қажетті ресурстарды бағалау қалай орындалады?
6. Тәуекелдің қандай түрлерін білесіз және олар қалай бағаланады?
7. Жұмыстың орындалуының уақытша графигін түсіндіріңіз.
8. Жоспарлау кезеңінде қандай метрикалар жиналады?
9. Жоспарлау кезеңін орындауда қандай құралдар, әдістер, стандарттар, шаблондар қолданылады?

7 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМ ТАЛАПТАРЫН БАСҚАРУ

7.1. Талаптарды басқару туралы жалпы деректер

БӨ жобалаудағы бірінші әрекеттердің бірі – оған қойылатын талаптарды жинау және реттеу. Бастапқы жиналатын талаптар тапсырыс берушінің бастапқы талаптарынан, тапсырыс берушілер мен пайдаланушылардың сұхбатынан және жиналыс протоколынан, әртүрлі құжаттардың түпнұсқалары мен көшірмелерінен, БӨ есептерінен және басқа материалдар массасынан тұрады. Жинағаннан кейін оларды реттеп, қайшылықтардан тазалайды. Одан кейін БӨ құраушыларына қойылатын талаптарды дайындайды – техникалық, бағдарламалық, деректер базаларында. Құрылымдық емес жиі пайдаланылатын талаптар мен ынталардың үлкен көлемімен жұмыс істеуге тура келеді, олар ниет туралы әртүрлі келісімдер бойынша таратылған және кездесулер протоколдарына, келісімшарттарға, зерттеудің алғаш жазылған материалдарына тіркелген.

Осылайша, осы талаптарды орындауды бақылау және тіркеу бойынша ұйымдасқан жігерсіз тәуекел жоғары болады және оларды есептеу мүмкін емес. Мәселенің шешімі белгілі: жиналатын талаптардың есебін жүргізу және олардың өңдеуін бақылау, бағалау (өткізуден бас тарту). Мұндай жұмыс *талаптарды басқару жұмысы* деп аталады.

Талаптарды басқаруға (requirements management) жатады:

БӨ талаптарын құжаттауға, ұйымдастыруға, анықтауға арналған жүйелі тәсіл;

БӨ талаптарын өлшеуге қатысты тапсырыс берушілер мен дайындаушылар арасындағы келісімді қалыптастыратын үрдіс.

Талаптарды басқарудың мақсаттары келесідей:

тапсырыс беруші және пайдаланушымен БӨ не істеуі керек екендігі туралы келісімге қол жеткізу;

БӨ талаптарын дайындаушылар тарапынан түсінуді жақсарту;

БӨ шектерін орнату, яғни, комьютер аппаратурасының, операциялық жүйенің, БӨ мүмкіндіктерінің техникалық

талаптарын анықтау;

жоспарлау базисін анықтау.

Талаптарды басқару маңыздылығы төтенше жағдайларда орындалатын барлық жобаларда барлық талаптарды үш категорияға бөле отырып, басымдылықтар орнатумен түсіндіріледі: «орындау керек», «орындаған дұрыс» және «орындауға болады». Жобадағы басымдылықтарды бұлай орнатуда стратегия келесідей болады: ең алдымен орындалуы тиіс талаптарға шоғырлану; одан кейін уақыт қалса, орындалғаны абзал болып саналатын талаптарға шоғырлану; және мүмкіндік болған жағдайда, орындауға болатын талаптарға шоғырлану.

Егер мұндай стратегияға жоба басталған сәттен сүйенбесе, онда жағымсыз дағдарыс жағдаймен қақтығысуға болады.

Талаптарды басқару техникалық қолдауы үлкен қаржылық шығындарды талап етпейтін жұмыстарға жатады, бірақ бұл жұмыстар құрылатын бағдарламалық өнім сапасын жақсарта алады.

Талап — БӨ қанағаттандыруы тиіс шарт немесе сипат. Функционалды және функционалды емес талаптар болады.

Функционалды талаптар БӨ орындауы тиіс әрекеттерді шектеулерсіз анықтайды. Басқа сөзбен айтқанда олар ақпаратты өңдеу үрдісінде БӨ тәртібін анықтайды.

Функционалды емес талаптар БӨ тәртібін анықтамайды, бірақ оның атрибуттарын немесе жүйелі шеңбер атрибуттарын сипаттайды. Функционалды емес талаптардың келесідей түрлерін бөліп қарастыруға болады:

Пайдалану талаптары — пайдалану интерфейсін, құжаттаманы және оқу курсын анықтайды;

Өнімділік талаптары — ресурстарды пайдалану тиімділігін, өткізу қабілеттілігін және реакция уақытын орната отырып, функционалды талаптарға шектеулер қояды;

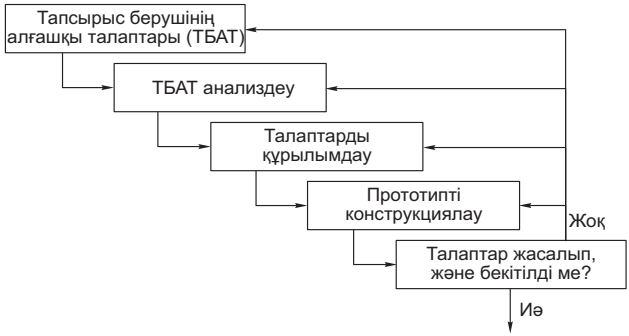
Өткізу талаптары — операциялық орта, бағдарламалау тілі, анықталған стандарттарды пайдалануды ұсынады;

Сенімділік талаптары — шекті жиілікті және БӨ жұмысындағы кемшіліктердің ықпалын, сонымен бірге кемшіліктерден кейін БӨ қалпына келтіру мүмкіндіктерін ескертеді;

Интерфейс талаптары — жүйе өзара әрекеттесетін сыртқы болмыстарды және осы өзара әрекеттестіктер реттемесін анықтайды (яғни, пайдаланушылар мен сыртқы құралдар).

7.2. Талаптарды қалыптастыру циклы

Талаптарды басқару кезеңін келешек БӨ құрылымын ресми анықтау кезеңі деп атауға болады. Осы кезең күрделі болып табылады, себебі БӨ жұмыс үрдісінде БӨ өзі туралы ұғым тапсырыс берушіде де, орындаушыда да өзгереді. Сондықтан, жобаның ерте кезеңінде БӨ қандай сипаттамаларға ие болуы керек екендігін анықтау қиын.



Сурет 7.1. Бағдарламалық өнімге талаптар қою циклы

Талаптарды құру, жоспарлау секілді циклдық үрдіс болып табылады (сурет 7.1), әрбір циклдағы (немесе қосалқы циклда) негізгі әрекетке БӨ сипатындағы семанитикалық қателерді анықтау және жою жатады. Талаптарды құру үрдісі нақты анықталған қорытындыдан тұрмайды. Ол аяқталуы үшін ақылға қонымды жеткіліктік қағидасы басшылық етеді, құрылған талаптардың семантикалық қатесі жоқ деген шешім қабылдайды және БӨ орындауы тиіс жұмысты сипаттайды.

7.3. Тапсырыс берушінің алғашқы талаптарын талдау және құрылымдау

Әдетте ТБТ бағдарламалық өнімді егжей-тегжейлі сипаттамайды. Әдеттегідей, жекелеген функциялардың сипаттамасы түсіндірілмеген және мүлдем айтылмайды.

ТБТ талдауы қадамында тапсырыс берушімен серіктестікте міндетті түрде (жеке кездесулер, телефонмен сөйлесу, электронды пошта) болжамды БӨ құру мүмкіндігін анықтау қажет, белгіленген мерзімдерде сипаттамада қай бөлімдердің жоқ екендігін анықтау керек. Одан басқа, жетіспейтін ақпаратты алу тәсілдерін және мерзімін, тестілеу тәсілдерін және дайын

бағдарламалық өнімді қабылдау шарттарын анықтау қажет.

Талаптарды құрылымдау бойынша жұмысты болашақ БӨ-ді толық сипаттаудан бастайды және жекелеген функциялар талқыланбайды (жұмыстың бұл бөлігі ТБТ талдауы қадамында орындалады), яғни, БӨ талаптарының жалпыланған құрылымын құрудан басталады.

Сонымен қатар, БӨ жалпы тағайындалуы, БӨ өткізілуі тиіс аппараттық құралдардың ерекшеліктері, жалпы ерекшеліктер, БӨ жұмыс істеуі керек кешен ерекшеліктері көрсетіледі, БӨ негізгі құраушыларын бөледі.

Оларды бөлуде олардың арасындағы байланыстардың аз болуына тырысу керек, яғни, ақпараттық тәуелсіздікті қамтамасыз ету қажет.

БӨ талаптарының жалпыланған құрылымы (жалпыланған сипаттама) жаһанды байланыстар, негізгі қасиеттер және БӨ құрылымдық жекелеген элементтерінің өзара байланыстары туралы ұғым ұсынуы қажет.

Ары қарай талаптардың жалпыланған құрылымы тексеріледі: негізгі құраушылардың өзара байланыстарының сипаты анықталады және әрбір құрылымдық элементтің егжей-тегжейлі сипаты орындалады (БӨ талаптарының егжей-тегжейлі құрылымын құрайды). Әрбір құрылымдық элемент үшін оның функцияларының тексерілуі орындалады және барлық құрылымдық элементтердің әрбір функциясының сипаты құрылады. Одан кейін әрбір құрылымдық элемент ішіндегі функциялардың өзара байланысы және әртүрлі құрылымдық элементтердің функцияларының сипатталады және анықталады.

Барлық талаптарды олардың бекітілген мерзімде немесе БӨ кезекті нұсқаларында кейінгі қалдырып орындау мүмкіндіктері тарапынан талдайды және өткізу күрделілігі бойынша орындаушылардың басымдылықтары мен тапсырыс берушінің басымдылықтары пайдаланылады.

БӨ талаптарының егжей-тегжейлі құрылымы құрылымдық элементтердің өзара байланыстары және олардың функциялары туралы толық түсінік беруі керек.

БӨ талаптарын құрылымдау бойынша соңғы әрекетке әрбір құрылымдық элементтің жекелеген функцияларының сипатын құру жатады.

Функциялардың әрқайсы үшін кіріс деректері және олардың орналасу орындары көрсетілуі, деректермен жүргізілетін әрекеттер егжей-тегжейлі сипатталуы, функцияның шығыс өнімі, орналасу орны, тестілеу әдістері анықталуы керек.

Әрбір талапты тексеру тереңдігін жоба жетекшісі анықтайды. Ол жекелеген функциялар барлық егжей-тегжейден

тұратындай және дайындаушы мен тапсырыс берушіге БӨ-нің неден тұратыны және не істей алатыны туралы анық түсінік беретіндей болуы қажет.

Сипаттаманы техникалық әдебиетте қабылданған терминологияда құру ұсынылады.

7.4. Прототипті құрылымдау

Прототипті құрылымдауды БӨ-ді жобалауға ұқсас орындайды. Прототип қасиеттерін игеру тапсырыс беруші талаптарын қорытынды бекітуге дейін анықтау және тексеру мақсатында жүзезе асырылады.

Бағдарламалық қамтамасыз ету тапсырысшыларына және шекті пайдаланушыларға дайындалатын БӨ талаптарын нақты қалыптастыру әдетте қиын болып келеді. БӨ-нің басқа бағларламалық пакеттермен өзара әрекеттесетінін және пайдаланушылармен орындалатын қандай операцияларды автоматтандыру қажет екендігін болжау қиын. Талаптардың егжей-тегжейлі талдауы БӨ не істеу керек екенін түсінуге мүмкіндік береді. Дегенмен талаптарды растамай тұрып, оны тексеру мүмкін емес. Бұл жағдайда жүйе прототипі көмектесе алады.

Прототип БӨ-нің бастапқы нұсқасы болып табылады, ол жүйеде қалыптасқан тұжырымдаманы көру үшін, талаптар нұсқасын тексеру үшін, дайындау кезінде пайда болатын мәселелерді табу үшін БӨ-ді қолдану кезінде және оны шешудің болжамды нұсқаларында қолданылады. БӨ прототипін тез дайындау өте маңызды, пайдаланушылар сол арқылы олармен тәжірибе жасауды бастай алады.

БӨ талаптарын құрумен жұмыс істеуде прототип бірнеше артықшылықтарды қамтамасыз етеді. Пайдаланушылар БӨ-нің қалай жұмыс істейтінін тексеруге мүмкіндік беретін прототиппен жұмыс істей алады. Олар БӨ әлсіз және күшті жақтарын анықтай алады, нәтижесінде жаңа талаптар қалыптасады.

Прототип ерте қабылданған талаптардағы қателер мен қалып қойғандарды анықтауға мүмкіндік береді. Мысалы, талаптарда анықталған БӨ-нің кейбір функцияларын пайдаланушылар алдымен пайдалы және қажетті деп санауы мүмкін, дегенмен бұл функцияларды басқа функциялармен пайдалану үрдісінде олар туралы пікірін өзгертуі де мүмкін. Нәтижесінде БӨ талаптары өзгертіледі және БӨ функцияларын пайдаланушылар басқаша түсінетін болады.

Прототиптеуді жоспарлау кезеңінде тәуекелдерді талдау үшін қолдануға болады. БӨ дайындаудағы негізгі қауіптілікке

талаптардағы қателер мен қалып қойғандар жатады. Талаптардағы қателерді жою шығындары дайындау үрдісінің соңғы кезеңінде жоғары болуы мүмкін. Прототиптеу жүйені дайындаудың жалпы құнын азайтады. Осы себептер бойынша ол талаптарды дайындау үрдісінде жиі қолданылады.

7.5. Тапсырыс беруші талаптары бойынша сипаттама құру

Талаптарды басқару кезеңі кесте ретінде кескінделетін талаптар сипаттамасын құрумен аяқталады.

Сипаттамаларды құру кезінде әртүрлі түсіндірмеден тұратын сөздер мен сөздер тіркесін пайдаланбаған жөн.

Талаптарды орындаушылармен және тапсырыс берушімен бекіту талаптар сипаттамасының барлық пункттері бойынша олардың арасындағы келісімге қол жеткізу сәтін анықтайды. Тапсырыс беруші прототипті немесе БӨ талаптарын тексеруді кескіндейтін басқа мысалды ұсынуды талап етеді, ол көптеген тексеру мысалдарын анықтай алады немесе БӨ жекелеген талаптарының параметрлерін бірнеше рет өзгерте алады. Бұл жағдайда тапсырыс берушінің мұндай ынталарын талаптар сиптаттамасының қосымшасында дайындау қажет.

7.6. Жиналатын метрикалар, пайдаланылатын әдістер, стандарттар мен шаблондар

Тапсырыс беруші талаптарының сипаттамасын құру кезеңінде талаптар сипаттамасын құруға жұмсалған ресурстарды бағалау қажет (талаптар сипаттамаларының бекітілген нұсқасын алуға қажетті уақыт, талаптар құруға қатысатын адам саны, талаптар сипаттамасының компьютерлік нұсқасын құруға қажетті уақыт және оның қағаздық көшірмелері, талаптар сипаттамасын растау және бекіту процедурасының ұзақтығы). Осы бағалаулар бойынша талаптар сипаттамасының құрылу өнімділігін анықтауға болады.

Барлық алынған деректерді жобалық топтағы деректердің тарихи базасында сақтау қажет. Одан басқа, ол жерге жоба басшысының қалауынша барлық басқа деректерді енгізу керек. Олар талаптар сипаттамасын құру үрдісін жақсартуға және оның өнімділігін арттыруға көмектеседі.

Пайдаланылатын құрал: құжаттарды дайындау жүйесі (мысалы, MS Word).

Пайдаланылатын әдістер мен стандарттар: ұйымдастыру

үрдісі; ұйымдастырудың метрикалық бағдарламасы.

Пайдаланылатын шаблондар: талап сипаттамалары; шолу есебі; жоба мәртебесі туралы есеп.

Бақылау сұрақтары:

1. Бағдарламалық өнімді дайындаудың өміршеңдік циклындағы бағдарламалық өнім талаптарын басқару кезеңінің мақсаты қандай?
2. Талаптарды басқару бойынша жұмыс қандай әрекеттерден тұрады?
3. Талаптарды басқару бойынша жұмыс мақсаттары қандай?
4. Басымдылықтарды анықтауда барлық талаптарды қандай категорияларға бөлуге болады?
5. Бағдарламалық өнім талаптарын қандай кезектілікте жүзеге асыру қажет?
6. Қандай талаптар функционалды және функционалды емес талаптарға жатады?
7. Бағдарламалық өнім талаптарын қалыптастыру циклы қандай әрекеттерден тұрады?
8. Тапсырыс берушінің алғашқы талаптарын талдау мақсаттары қандай?
9. Тапсырыс беруші талаптарын құрылымдауда қандай әрекеттер орындалады?
10. Бағдарламалық өнім талаптарын қалыптастырудағы прототип рөлі қандай?
11. Прототип қандай артықшылықтарды қамтамасыз етеді?
12. Неге талаптар сипаттамасындағы талаптар бірмәнді болулары керек?
13. Талаптарды басқару кезеңінде қандай метрикалар жиналады?
14. Қандай әдістер, стандарттар мен шаблондар талаптарды басқару кезеңінде қолданылады?

8 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ЖОБАЛАУ

8.1. Жобалаудың жалпы сипаттамасы мен құраушылары

Жобалау кезеңі өндірілетін бағдарламалық жүйе моделін жасау мен талдау үшін арналған. Мұндай модель іске асырудың кезекті кезеңіне қажетті бағдарламалық жүйе құрылымын, модульдер,интерфейс пен мәліметтердің ұйымдастырылуын анықтайды. Белгіленген кезең әр қайсысы үшін қасиеттер жинағы мен басқа құраушылармен байланысы анықталатын жобалау құраушыларының жиынтығы түрінде ұсынылады.

Жобалау құраушысы ретінде тапсырыс берушінің БӨ-на талаптарының бөлшектеп байланыстыру нәтижесінде алынған жобалау элементі саналады.Жобалау құраушысы болып жүйелер, жүйе бөлігі, модульдер, мәліметтер элементтері мен т.б. есептеледі. Олардың барлығы құраушы атрибуттары деп аталатын жалпы сипаттамаға ие. Жобалау құраушысы үшін келесі атрибуттар анықталуы мүмкін:

- құраушы атауы — оның бірегей аты;
- құраушы типі - оның болмысы (жүйе бөлігі, процедура, процесс, мәліметтер элементі мен т.б.)
- функция — құраушымен жасалатын әрекеттер;
- тәуелділіктер — басқа құрауыштармен өзара байланысты сипаттау;
- интерфейстер — басқа құрауыштармен өзара әрекет тәсілдерін сипаттау;
- ресурстар — қажетті аппараттық,кітапханалық немесе басқа қолдаудың сипаттамасы;
- өңдеу —функцияны орындау алгоритмінің сипаттамасы;
- мәліметтер — мәліметтердің ішкі құрылымдарының сипаттамасы.

Қажетті құраушылар, олардың атаулары мен мақсаттарының тізімі БӨ моделін жасаудың таңдалған әдісіне байланысты анықталады, және бірінші кезекте, бағдарламалаудың таңдалған әдісіне (мысалы, құрылымдық бағдарламалау немесе объектілі-нысаналы) байланысты. Осы әдістердің әрқайсысы БӨ моделін жасаудың өзіндік әдістеріне ие.

Жобалау нәтижелері атрибуттың анықталған жиынтығы бойынша жобалау құраушысының сипаттамасы түрінде ұсынылады.

8.2. Бағдарламалық өнімді жасау эволюциясы

Алғашқы электронды-есептеу машиналарының пайда болуымен бағдарламалық қамтамасыз етуді жасау үлкен жол жүрді. Қандай да бір бағдарламаны жазу мүмкіндігімен масаттану нақ бағдарламалық қамтамасыз етуді жасау технологиясы есептеу техникасында прогрессті анықтайды деген оймен алмастырылды.

Алғашқы кезеңдерде есептеуіш техникасының дамуы техникалық мәселелерді шешуге бағытталды. Мәселе мәні бірінші кезекте аппаратура, яғни есептеуіш машинада болды. Мұндай машиналар үшін бағдарламаларды екілік кодта жасайды деген ой болды. Бағдарламалау энтузиасттардың еншісінде еді.

Есептеуіш машиналарының мықты және берік бола бастауына байланысты бағдарламалық қамтамасыз ету мәні көптеген ғалымдар мен практиктермен ұғыныла бастады. Маңызды жаңалық 1950-жылдардың аяғында болды, ол кезде жоғары деңгейлі бағдарламалау тілдері пайда болды- Fortran, Algol және т.б. Олардың пайда болуы осындай тілдерді қолданбай бағдарламаны жазу қиын мәселеге айналғандығымен шартталады. Ағымдағы мәселелерді шешіп, бұл тілдер жаңаларын жасады. Бағдарламалауды тездетіп және жеңілдетіп, олар ЭЕМ көмегімен шешілетін мәселелер шеңберін айтарлықтай ұлғайтты. Бұрын шешілгендерге қарағанда қиынырақ мәселелердің пайда болуы қолдағы құралдар тағы да мәселені оңтайлы шешу үшін жеткіліксіз болуына алып келді.

Мұндай даму компьютерлерді пайдалану тарихының барлық кезеңіне, тіпті қазіргі кезге дейін, сай келеді. Есептеуіш техника әрдайым өз мүмкіндіктері шегінде қолданылады. Аппараттық немесе бағдарламалық қамтамасыз етудегі әрбір жаңа жетістік ЭЕМ-ді қолдану саласын кеңейту талпыныстарына алып келеді және сөйтіп шешілу үшін жаңа мүмкіндіктерді талап ететін жаңа міндеттер қояды.

Қалыптасқан жағдайда бағдарламашы талпыныстары бірінші кезекте бағдарламалаудың жаңа тілдерін жасауға бағытталады. Экономикалық ақпаратты өңдеу мәселелерін шешуге мүмкіндік беретін Cobol тілі пайда болды. Ол ондағы бағдарлама ағылшын тіліндегі мәтінге ұқсас етіліп жасалды. Бағдарламау тілдері ие бола алатын барлық мүмкіндікті біріктіретін PL/I тілі пайда болды.

Fortran, Algol, PL/I тілдері бағдарламалаудың процедуралық стилін қолдайды. Бағдарлама ол өзі атқаратын әрекет терминінде жасалады. Бағдарламаның негізгі бірлігі процедура болып табылады. Процедуралар басқа процедуралар шақырады, сөйтіп

олардың барлығы мәселені шешуге әкелетін арнайы алгоритм бойынша жұмыс істейді. Алгоритм – қажетті нәтижеге қол жеткізу үшін әрекеттердің нақты реттілігі.

Жоғарыда аталған тілдердің пайда болуы есептеуіш техника көмегімен шешілетін мәселелер шеңберінің нәтижесі болды. Алғашқы кезекте олар есептеуіш мәселелер еді. ЭЕМ-нің негізгі тұтынушылары қажетті нәтижеге тез қол жеткізу мүмкіндігі қызықтыратын ғалымдар мен инженерлер болды.

Жасанды интеллект пен мәтіндерді өңдеу мәселелерін шешу үшін ЭЕМ-ді қолдану функционалды тілдердің, оның ішінде Lisp тілінің пайда болуына әкелді. Бұл тілдер жақсы зерттелген математикалық негіз – лямбда-есептеуге ие. Әрекеттер көбіне итерация түрінде көрінетін (бағдарлама фрагментінің бірнеше рет қайталануы) Algol типті тілдерге қарағанда, Lisp тілінде есептеулер рекурсия көмегімен жасалады – өзін-өзі функциясының шақырылуы, оның мәліметтерінің негізгі құрылымы тізім болып табылады.

1960-жылдары көптеген теориктер мен практиктер бағдарламалаудың жаңа, қазіргі заманғы тілдердін жасау бағдарламаны жасаудың барлық мәселесін шеше алмайтынын ұғынды. Бағдарламаны тестілеу, бағдарламалық қамтамасыз етуді жасау процесін ұйымдастыру салаларында қарқынды зерттеулер басталды.

Бағдарламалауды эмпирикалық процесстен жақсы реттелгенге келтіру,оған «инженерлік» сипат беру талпыныстары бағдарламалау әдістерінің жасалуына алып келді. Бұл өте маңызды қадам болды. Бағдарламаны жасау әдістерін құру, бір жағынан, жалпы өнеркәсіптік бағдарламалау үшін негіз жасады, екінші жағынан, бағдарламалық қамтамасыз етудің ағымдық жағдайын жалпылай және саралай отырып, бағдарламаудың жаңа тілдерін, операциялық жүйе, бағдарламалау орталарын жасауға мықты импульс берді.

Бағдарламалаудың ең кең таралған әдісі ретінде бағдарламалауға құрылымдық келу саналды. Оны жасаушы - Э. Дейкстра (E. Dijkstra). Бағдарламалауға шынайы құрылымдылық келу – бағдарламалаудың алғашқы аяқталған тәсілі. Оны аяқталған деудің себебі ретінде құрылымдық бағларламалау оның міндетінен бастап бағдарламада орындалуына дейін жолды ұсынатындығы саналады. Құрылымдық бағдарламалау бағдарламалаудың дамуына зол үлес қосты. Тәжірибелік бағдарламалауда кең таралған бұл әдіс әлі күнге дейін міндеттің арнайы классы үшін өз маңызын жоғалтпады.

Құрылымдық әдіс екі негізгі принципке негізделеді:

■ бағдарламалаудың процедуралық стилін қолдану;

■ жоғарыдан төмен қарай тізбектес декомпозиция (орналасу).

Мәселе кейбір тізбекті әрекеттерді орындау арқылы шешіледі. Алғашында ол кіріс-шығыс терминінде (бағдарлама кірісіне бастапқы мәліметтер беріледі,ал бағдарлама орындалғаннан кейін шығыста жауап беріледі) жасалады.Кейіннен барлық мәселенің жеңілірек әрекеттерге тізбекті жіктелуі басталады. Мысалы, егер бізге адрестің дұрыстығын тексеру бағдарламасын жазу керек болса, онда біз оны алғашында келесідей жазамыз:

1. Адресті оқу.
2. Адресті базада бар адрестермен тексеру.
3. Тексеріс нәтижесі оң болса, «Иә» деп басу, ал теріс болса, «Жоқ» деп басу керек.

Мұндай жазу бағдарламада кейде жоғары дәрежелі тілде,мысалы Pascal немесе С-де көрінуі мүмкін.

Кез-келген кезеңде бағдарламаны тексеруге болатындығы өте маңызды, тек бітеуіштер – төмен деңгейлі процедуралардың кірісі мен шығысын имитациялайтын процедуралар – жазу жеткілікті. Жоғарыда көрсетілген бағдарламада терминалдан енгізудің орнына кез-келген белгіленген адресті келтіретін адресті оқу процедурасын және ешнәрсе істемейтін, тек дұрыс аяқталукодын қайтаратын мәліметтер базасымен тексеру процедурасын қолдануға болады.

Бағдарлама бітеуіштермен құрастырылады және жұмыс істей алады. Басқаша сөзбен, бітеуіштер жоғарғы деңгей логикасын келесі деңгейдің іске асуына дейін тексеруге мүмкіндік береді. Бітеуіш әдісін тізбекті қолдану арқылы бағдарламаны жасаудың әрбір қадамында уақыт өте келе детальдармен толықтырылатын жұмыс істейтін қаңқаға ие болуға болады.

Құрылымдық бағдарламалау 1960-жылдардың аяғында пайда болған (Pascal, Algol- 68, Fortran және т.б.) бағдарламалау тілдерінен қолдау тапты. Сол уақытта есептеуіш техника көмегімен мәселелерді шешуде бағдарламалық қамтамасыз ету мағынасы ұғынылды. Бұл тілдер әртүрлі салынған процедуралар, параметрлерді жіберудің түрлі әдістерн қолдады.

Құрылымдық бағдарламалау үлкен жобалардың жасалуында бағдарламалардың модульдық құрылуының мәнін анық анықтауына қарамастан, бағдарламалау тілдері модульдікті әлі әлсіз қолдады. Бағдарламалар құрылымдығының жалғыз әдісі ретінде оның ішкі бағдарлама немесе функциядан жасалуы саналады. Функцияны шақырудың дұрыстығын бақылау, оның ішінде шынайы аргументтердің саны мен типінің күтілетін формалды параметрлерге сай келуін бақылау орындау (функция прототипі мағынасы кейін пайда болды) сатысында ғана жүзеге асырылады.

Объекті-бағдарлы бағдарламалау бағдарламалаудың үш маңызды мәселесін ұғыну нәтижесінде пайда болып, кең таралды.

Бірінші мәселе бағдарламалаудың тілдері мен әдістерінің дамуы бағдарламалық жүйелердің жасалуын жеңілдетіп, жылдамдатса да, одан да жылдам дамитын бағдарлама талаптарына жете алмауында еді. Дамытуды лезде жылдамдатудың жалғыз шынайы тәсілі ретінде құрастырылған бағдарламалық қамтамасыз етуді бірнеше рет қолданудың тәсілі саналды. Басқаша сөзбен, әр жолы жүйені нөлден бастау қажет етілмеді, дұрыстау мен тестілеу циклінен өтіп, тиімді жұмыс атқарған бұрын жасалған модульдерді қолдану қажет етілді.

Процедуралық бағдарламалау бағдарламаны жасау үшін блок ретінде қызмет ететін функцияны жасау тәсілін ұсынды. Алайда бұл әдістің икемділігі де, қолдану масштабтары да жалпы бағдарламалауды айтарлықтай жылдамдатуға мүмкіндік туғызбады.

Біріншісіне байланысты екінші мәселе құрастырылған жүйені сүйемелдеу мен модификациялауды жеңілдету қажеттілігі болып саналады. Жүйеге талаптардың тұрақты өзгеру фактісі құрастырушылардың білместігі немесе жұмыстың айтарлықтай анық орындалмауы ретінде емес, жүйенің дамуының қалыпты шарты ретінде есептелді. Жүйені сүйемелдеу мен модификациялауға құрастыруға кететін күштен аз күш жұмсалмайды. Біріншіден, жергілікті модификациялар барлық жүйенің жұмысқабілеттілігін бұзбайтындай ету мен, екіншіден, жүйе тәртібінің өзгеруін оңай жасау үшін бағдарламалық жүйелердің жасалу әдісін радикалды өзгерту қажет болды.

Үшінші мәселе,шешілуді талап ететін ең елелеулі мәселе – жүйені жобалауды жеңілдету. Барлық мәселелер құрылымдық бағдарламалау талап ететін алгоритмдік көрініске ермейді, оның ішінде алгоритмдік декомпозияға.Бағдарлама құрылымын шешілетін мәселеге жақындату,олардың арасында семантикалық алшақтықты қысқарту керек болды. Семантикалық алшақтық жөнінде мәселе тілі мен оны шешу әдісі негізінде жататын ұғымдар әртүрлі болған кезде айтылады. Сондықтан шешімді жазу қажеттілігімен бірге бір ұғымды екіншісіне ауыстыру да қажет етіледі. Оны бір табиғи тілден екіншісіне ауыстырумен салыстырыңыз. Орыс тілінде ондай ұғымдар болмағандықтан, «брокер», «оффшор» немесе «инвестор» деген сияқты сөздер пайда болады. Өкінішке орай, бағдарламалауда кірме сөздер жоқ.

Сөйтіп, жобалауды жеңілдету, дайын модельдерді бірнеше рет қолдану арқылы құрастыруды жылдамдату мен модификация

жеңілдігі – оның жақтаушыларымен насихатталатын объектілі-бағдарлы бағдарламалаудың үш негізгі құндылығы осы.

Бағдарламалауға объектілі-бағдарлы келу бірінші кезекте Smalltalk, C++, Java және т.б. сияқты бағдарламалау тілдерімен байланыстырылады. Бұл ұстам едәуір негіздерге ие. Тілдер объектілі-бағдарлы бағдарламалаудың басты құралы болғандықтан, оларды жасау барысында қазіргі таңда объектілі-бағдарлы әдістің негізін құрайтын көптеген идеялар пайда болды.

Жаңа идеялардың жиналуы 1960-жылдың аяғынан бастап, шашамен 10 жыл болды. 1970-жылдардың аяғына таман абстракцияның жаңа деңгейіне өту академиялық ортаның фактісі болды. Жаңа идеялар өнеркәсіпке ену үшін тағы 10 жыл керек еді. Объектілі модельді жасау жолындағы алғашқы қадам ретінде мәліметтердің абстрактілі типінің пайда болуы саналады.

Абстракция деңгейі бойынша жүйенің құрылымдану қажеттілігін Э.Дейкстра көрсетті. Инкапсуляция (ақпаратты жасыру) идеясы Д.Парнаспен айтылды. Кейінірек типтер мен подтиптер теориясында С.Хоормен толықтырылған мәліметтердің абстрактілі типтерінің механизмі жасалды.

Бағдарламалау тіліндегі мәліметтердің абстрактілі типін алғашқы толық іске асырған Modula, CLU, Euclid және т.б. тілдерге сүйенетін Simula тілі. Бағдарламалаудың алғашқы «шынайы» объектілі-бағдарлы тілі Паоло-Альтода «Ксерокс» компаниясының лабораториясында жасалған Smalltalk тілі. (Көп адамдар әлі күнге оны бағдарламалаудың жалғыз шынайы объектілі тілі деп санайды). Кейін бағдарламалаудың қазіргі жағдайын анықтайтын басқа да объектілі-бағдарлы тілдер пайда болды (пайда болуын жалғастыруда).Олардың ішіндегі кең таралғандары - C++, CLOS, Eiffel, Java.

Алайда тек бағдарламалаудың тілдерімен объективті-бағдарлық тәсіл бітпейді. Тілдер бар болғаны қолдануға болатын не болмайтын инструментарийді береді. C++ тілінде процедуралық стильде жазуға болады. Объектілі-бағдарлық бағдарламалау жүйені жобалау, жасау мен дамытудың біріккен тәсілін ұсынады.

Объектілі-бағдарлық әдістің пайда болуы бағдарламалық қамтамасыз етудің жасалу әдістерінің дамуы, сонымен қатар ғылымның басқа да салалары негізінде болды. Бағдарламалау саласындағы американдық сарапшы Гради Буч (Grady Booch) бағдарламалау тілінің дамуынан басқа, жүйені жобалаудың объектілі-бағдарлық тәсілінің пайда болуына әсер еткен келесі жетістіктерді атап өтті:

1. Есептеуіш техниканың дамуы, оның ішінде операциялық жүйенің негізгі концепциясының аппараттық қолдауы мен функционалды-бағдарлы жүйенің жасалуы. Есептеуіш машинаны жобалауда объект түсінігі нефоннеймандық сәулет бойынша зерттеуден басталды.Дәстүрлі процессорлардың төмендеңгейлі сәулеті мен операциялық жүйенің жоғарыдеңгейлі түсінігі арасындағы семантикалық айырманы азайту талпынысы Барроус, Intel i432, IBM/38 және т.б. жүйелерді жасауда қолданылды. Олармен тығыз байланысты операциялық жүйенің объектілі-бағдарлы талдамасы Э.Дейкстра басшылығымен THE жобасынан басталып, Intel i432 және т.б. үшін iMAX жүйесінде жалғасын тапты. Салыстырмалы жаңа Cairo Microsoft пен Taligent Pink жобалары (зерттеу деңгейінде тоқтаған) – олар да объективті-бағдарлық операциялық жүйе жобалары..

2. Бағдарламалау әдіснамасындағы, оның ішіндегі жүйенің модульдық жасалуы мен ақпарат инкапсуляциясындағы жетістіктер.

3. Объектілі бағдарламалауға объектілер арасында қатынас құру идеясын енгізген мәліметтер базасымен басқару жүйесінің теориясын құру мен модельдеуді жасау. Объектілер арасында қатынас арқылы мәліметтерді модельдеу алғаш рет П. Ченмен (P. Chen) ER – модельдеу тәсілінде ұсынылды. Бұл тәсілде мәліметтер моделі объектілер (болмыстар), олардың атрибуттары мен арасындағы қатынастар түрінде жасалады.

4. Абстракция механизмдерін жақсы ұғынуға мүмкіндік туғызатын жасанды интеллект саласындағы зерттеулер. Образдарды тану жүйесінде шынайы объектілерді ұсыну үшін М.Минскиймен ұсынылған фреймдер теориясы тек қана жасанды интеллект жүйесі емес, сонымен қатар бағдарламалау тіліндегі абстракция механизмдерінің дамуы үшін мықты импульс берді.

5. Философия мен тану теориясының дамуы. Жүйенің объектілі-бағдарлы жасалауы – модельденетін шынайы әлемге анық көзқарас. Осы аспектіде философия мен тану теориясы объектілі модельге қатты ықпал етті. Ежелгі гректер әлемді объект немесе процесс түрінде қарастырған. Декарт адам үшін қоршаған әлемді объектілі-бағдарлық қарастыру табиғи деп атап өткен. М.Минский сана жеке ойлана алмайтын агенттердің өзара әрекеті ретінде шығады деп есептеді.

Образдар немесе шаблондар концепциясын шығарған сәулет пен құрылыс теориясы соңғы жылдары объектілі-бағдарлы анализ(ОБА) бен объектілі-бағдарлы жобалау (ОБЖ) салаласында белсенді қолданылады.

Бағдарламалау тәсілі саласында қарқынды зерттеу нәтижесі

ретінде бағдарламаны жасау автоматизациясы құралының немесе CASE - құралының (Computer Aided Software Engineering) көптігі саналады. Кез-келген жоғарыдеңгейлі тілде міндетті жазудан кейін бұл жүйелер автоматты түрде (немесе тым болмағанда адамның минималды қатысуымен) қойылған міндетті сәйкесінше шешетін дұрыс жұмыс істейтін дайын бағдарламаны өндіреді деп болжанды.

CASE-құралдар қажет нәтижеге қол жеткізе алмады: қиындық бойынша міндет сипаттамасы нәтижелі бағдарламамен салыстырылатындай болды, немесе жеңіл міндеттермен салыстырғанда тар шектелген шеңбер шешілді, немесе өндірілетін бағдарлама сапасы тым төмен болды.

Негізгі мақсатқа қол жеткізу талпынысының сәтсіздігіне қарамастан, CASE-құралын жасау тәжірибесі бағдарламалау тәсілдерін дамыту үшін өте көп нәрсе берді. Жүйенің кішкене бөлігі тура (мысалы,lex пен уасс компиляторларының генераторлары) қолданылса, идея бөлігі бағдарламаның тез жасалуының қазіргі құралдарын жасауда (RAD — Rapid Application Development), JBuilder, Delphi, PowerBuilder және т.б. сияқтылар қолданылды.

Толықтыққа мүлде ұмтылмайтын бағдарламалау әдіснамасының дамуының келтірілген тарихи экскурсы бағдарламалау әрдайым қиын мәселелерді шешу мүмкіндігі үшін, қиын бағдарламалық жүйелер жасау мен оны тезірек жүзеге асыру үшін «күресті» деп мысал келтірді. Уақыт өте келе сол немесе басқа жетістіктер панацеямен (ол құрылымдық бағдараламалау, не CASE-құралы немесе тағы басқа болсын) жарияланды. Алайда кең жарнамаланған құрал барлық мәселелерді шешуге қабілетті емес екендігі мәлім болатын.

8.3. Құрылымдық бағдарламалау

Алғашқы ЭЕМ пайда болуы есептеуіш техниканың дамуында жаңа саты жасады. Әрқайсысын түсінікті ЭЕМ нұсқаулыққа ауыстыруға болатын және кез-келген есептеуіш мәселе шешіле алатын қарапайым әрекет тізбектігін жасау жеткілікті болатын идея пайда болды. Бұл тәсіл бағдарламаны жасау процессінде басқалары алдында ұзақ уақыт бойы үстем болатын тіршілікке икемді болды. Бағдарламалаудың арнайы тілдері пайда болды, олар сәйкес бағдарламалық кодқа жеке есептеуіш операцияларды жасауға мүмкіндік туғызды.

Бағдарлама жасау әдіснамасы мәліметтерінің негізі бағдарламалық код құрылымының процедуралық немесе алгоритмдік ұйымдастыру болды. Оның есептеуіш міндеттерді

шешуде шынайылығы соншалықты жоғары дәрежеде болды, тіпті ешкім бұл тәсілдің мақсаттылығына күмән келтірмеді. Бұл әдіснамада бастауыш ретінде «алгоритм» түсінігі болды, оның мәні қойылған мақсатқа қол жеткізу немесе мәселені шешуге бағытталған әрекеттердің нақты анықталған тізбектігін жасауды ұйғаруда.

Осы көзқараспен, математиканың бүкіл тарихы өз дәуірі үшін өзекті мәселелерді шешу алгоритмдерін жасаумен тығыз байланысты. Сонымен қатар, «алгоритм»түсінігінің өзі сәйкес теорияның пәні болды – олардың ортақ қасиеттерін зерттеумен айналысатын алгоритмдер теориясы. Уақыт өте келе бұл теорияның мазмұны абстрактілі болғандығы соншалықты, оның нәтижелерін тек мамандар түсінетін болды. Бұл дәстүрдің алымы ретінде бір уақыт ішінде бағдарламалау тілдері алгоритмдік деп аталды, ал бағдарламаны құжаттандырудың бірінші графиктің құралы «алгоритмнің блок-сызбасы» деген атқа ие болды. Графиктік белгілеудің сәйкес сызбасы алгоритмдер, бағдарламалар, мәліметтер мен жүйелер сызбасында шартты белгілерді қолдануды регламенттейтін МемСТ 19.701 — 90-де бекітілген.

Алайда тәжірибе қажеттілігі нақты функцияның есептелуі мен жеке міндеттердің шешілуін бекітуді әрдайым талап етпеді. Бағдарламалау тілінде жаңа түсінік пайда болып, бекітілді – компьютерлерде міндетті шешуде қолданылатын «алгоритмнің» жалпы түсінігін нақтылаған «процедура». Алгоритм сияқты процедура жеке міндетті шешуге бағытталған әрекет немесе операциятның аяқталған тізбектілігін ұсынады. Бағдарламалау тілдерінде «процедура» деп аталатын арнайы синтаксистік конструкция пайда болды.

Уақыт өте келе үлкен бағдарламаларды жасау ауыр мәселеге айналды және оның бірнеше майда фрагменттерге бөлінуін талап етті. Мұндай бөлінудің негізі болып процедуралық декомпозиция саналды, бұл жерде бағдарламаның жеке бөліктері немесе модульдері міндеттердің кейбір жиынтықтарын шешу үшін процедура жиынтығын құрады.Процедуралық бағдарламалаудың басты ерекшелігі – бағдарлама әрдайым уақыт бойынша бастау алады, немесе бастапқы процедураға (бастапқы блок) және соңғы процедураға (соңғы блок) ие. Бұл ретте барлық бағдарлама графиктік примитив немесе блоктың бағыттылған тізбектігі түрінде визуалды ұсынылуы мүмкін.

Мұндай бағдарламаның маңызды қасиеті келесі процедураның әрекетінің басталуы үшін алдыңғы процедураның барлық әрекеттерін аяқтау қажеттілігінде жатыр.Бір процедура шегінде де бұл әрекеттердің орындалу тәртібінің өзгеруі бағдарламалау

тілдеріне міндетті шешудің аралық нәтижелеріне байланысты есептеуіш процесті тамырлауды жүзеге асыру үшін if-then-else пен go to сияқты арнайы шарттық операторларды қосу талап етілді.

Шартты оператор мен сөзсіз өту операторының пайда болуы мен қарқынды пайдалану бағдарламалау бойынша мамандар арасында қатты талқылануға салынды. Оператор бағдарламасында go to сөзсіз өтуді бақылаусыз қолдану кодты түсінуді қиындатуы мүмкін. Сәйкес бағдарламаларды bowl of spaghetti деп аталатын спагеттимен салыстырды,олай атау себебі бағдарламаның бір фрагменттен екіншісіне бірнеше рет өтуі, немесе оданда жаман бағдарламаның соңғы операторынан бастапқы операторға қайта өтуінде еді.

Жағдай соншалықты драмалы болды, тіпті әдебиетте go to операторын бағдарламалау тілдерінен алып тастау ұсыныстары болды. Дәл сол сәттен бастап go to операторынсыз бағдарламалау жақсы стиль деп саналды.

Қарастырылған идеялар *құрылымдық бағдарламалау әдіснамасы* деген атаққа ие болған бағдарламалық кодтарды жазу мен бағдарламаны жасау процесіне кейбір көзқарастардың пайда болуына себепкер болды. Әдіснама негізі ретінде бағдарламалық жүйенің процедуралық декомпозициясы мен орындалатын процедура жинағы түріндегі жеке модульдердің ұйымдастырылуы саналады. Аталмыш әдіснама аясында бағдарламаның бәсеңдейтін жобалануы немесе «жоғарыдан төмен» бағдарламалану дами бастады. Құрылымдық бағдарламалау идеясының танымалдығы 1970-жылдардың аяғы мен 1980-жылдардың басына келді.

Бағдарламалық кодтың құрылымдануының қосымша құралы ретінде салынған циклдер мен шартты операторларды бөлуге тиісті әрбір тармақ басында шегініс қолдану ұсынылды. Мұның барлығы бағдарламаның өзін ұғыну немесе оқылымды болуына септігін тигізуге арналған. Аталмыш ереже уақыт өте келе бағдарламаны жасаудың қазіргі инструментарийларында іске асты.

8.4. Объектілі-бағдарлы жобалау

Объектілі-бағдарлы тәсілдің негізгі ұғымдары ретінде объект пен класс саналады.

Объект сезілетін шындық (tangible entity) ретінде анықталады, яғни анық анықталатын тәртіпке ие пән немесе құбылыс. Объект жағдай, тәртіп пен даралықпен сипатталады; ұқсас объектілердің құрылымы мен тәртібі олар үшін жалпы

классты анықтайды. «Класс экземпляры» мен «объект» терминдері эквивалентті болып саналады. Объект жағдайы осы қасиеттердің әрқайсысының ағымдағы (динамикалық) мағынасы мен аталмыш объекттің мүмкін болатын (статикалық) қасиеті тізімімен сипатталады. Объект тәртібі оның басқа объектілерге әсер етуі мен керісінше осы объектілердің жағдайының өзгеруі мен хабарды жіберу көзқарасымен сипатталады.Басқаша сөзбен, объект тәртібі оның әрекетімен толық анықталады. Бірегейлік – объектіні басқа объектілерден бөлетін объект қасиеті.

Бір объекттің басқа объектке сәйкес реакция шақыру мақсатымен әсер етуі операция деп аталады. Әдетте, аталмыш объектпен жасалатын объектілі және объектілі-бағдарлы тілдерінде операциялар тәсіл деп аталып, классты анықтаудың құрамдас бөлігі болады.

Класс — құрылым мен тәртіп бірлігімен байланыстырылған көптеген объектілер. Кез-келген объект класс данасы болып табылады. Класстар мен объектілерді анықтау – объектілі-бағдарлық жобалаудың қиын мәселелерінің бірі.

Объектілі-бағдарлық тәсілдің маңызды түсініктерінің келесі тобын мұрагерлік пен полиморфизм құрайды. Полиморфизмді класстың бір типке қарағанда одан көп типке қосылу қабілеті ретінде есептеуге болады. Мұрагерлік мәліметтер мен тәсілдерді қосу немесе қайта анықтау мүмкіндігімен бұрыннан бар класстар негізінде жаңа класстарды жасауды білдіреді.

Объектілі-бағдарлық жүйе басынан оның эволюциясы есебімен жасалып келеді. Мұрагерлік пен полиморфизм туынды класстарды – базалық класстардың ұрпақтары - жасау арқылы жаңа функционалдықты анықтау мүмкіндігін қамтамасыз етеді. Ұрпақтар бастапқы көріністің өзгеруінсіз ата-аналар класстарының сипаттамаларын иеленеді және қажет болған жағдайда мәліметтер мен тәсілдердің өзіндік құрылымын енгізеді.Ерекшеліктер немесе нақтылаулар берілетін туынды класстарды анықтау спецификация мен бағдарламалық коды жасау мен қолдануда уақыт пен күшті үнемдейді.

Құрылымдық және объектілі-бағдарлы тәсіл арасындағы принципиалды айырмашылық жүйе декомпозициясы тәсілінде жасалады. Объектілі-бағдарлы тәсіл объектілі декомпозицияны қолданады, бұл ретте жүйенің статикалық құрылымы объектілер мен байланыстар арасындағы терминде сипатталады, ал жүйе тәртібі объектілер арасында хабар алмасу терминінде бейнеленеді. Жүйенің әрбір объектісі шынайы әлем объектісін модельдейтін өзінің жеке тәртібіне ие.

«Объект» ұғымы алғаш рет шамамен 30 жыл бұрын фон

Нейманның дәстүрлі сәулетінен жылжу мен бағдарламалық абстракцияның жоғары деңгейі мен абстракциялаудың төмен деңгейі арасындағы тосқауылдан өту талпынысымен қолданылды. Объектілі-бағдарлы сәулетпен объектілі-бағдарлы операциялық жүйелер тығыз байланысты. Объектілі тәсілге ең көп салым Simula, Smalltalk, C++, Object Pascal бағдарламалаудың объектілі және объектілі-бағдарлы тілдерімен жасалды. Объектілі тәсілге тәуелсіз дамып келе жатқан мәліметтер базасын модельдеу тәсілдері де өз ықпалын тигізді, оның ішінде болмыс-байланыс тәсілі.

Объектілі-бағдарлы тәсілдің концептуалды негізі ретінде объектілі модель саналады. Оның негізгі элементтері - абстракциялау abstraction), инкапсуляция (encapsulation), модульдік (modularity), иерархия (hierarchy).

Негізгілермен қатар тағы үш қосымша элемент - типизация (typing), параллелизм (concurrency), тұрақтылық (persistence), олар негізгілерге қарағанда міндетті болып саналмайды.

Абстракциялау – басқа объектілер түрлерінен ерекшелейтін, және сөйтіп кезекті қарау мен анализ үшін концептуалды шегін нақты анықтайтын кейбір объектінің сипаттамасының бөлінісі. Абстракциялау объектінің сыртқы ерекшеліктеріне аса мән береді және оның жүзеге асырудан детальдардың тәртібінің ерекшеліктерін бөлуге мүмкіндік береді. Берілген пәндік сала үшін абстракцияның дұрыс жиынтығын таңдау объектілі-бағдарлы жобалаудың басты міндетін ұсынады.

Инкапсуляция — объектінің жасалуы мен тәртібін анықтайтын жеке элементтерді бір-бірінен бөлу процесі. Инкапсуляция объектінің ішкі реализациясынан сыртқы тәртібін көрсететін объектінің интерфейсін оқшаулау үшін қызмет етеді. Объектілі тәсіл класстың өзінің тәсілдері бақылай алатын жеке ресурстары сыртқы ортадан жасырылған деп санайды.

Абстракциялау мен инкапсуляция өзара әрекеттесін операция болып табылады: абстракциялау объектінің сыртқы ерекшеліктеріне назар аударса, инкапсуляция объект-қолданушыларға объектінің ішкі құрылысын көруге мүмкіндік бермейді.

Модульдік —бір бірімен іштей байланысқан, бірақ әлсіз біріктірілген модульдер қатарына декомпозия мүмкіндігімен шартталған жүйе қабілеті. Инкапсуляция мен модульдық асбстракциялар арасында тосқауыл жасайды.

Иерархия —асбтракцияның сараланған немесе реттелген жүйесі, оның орналасуы деңгей бойынша. Қиын жүйеге қатысты иерархиялық құрылымның негізгі түрлері - класстар құрылымы (номенклатура бойынша иерархия) мен объектілер құрылымы

(құрам бойынша иерархия).Класстар иерархиясы мысалы - оңай және көп мұрагерлік (бір класс бір немесе бірнеше басқа класска қатысты құрылымдық немесе функционалды бөлікті қолданады), объект иерархиясы мысалы – агрегация.

Типтеу— объектілер классына қойылатын және әртүрлі класстардың өзара ауысына бөгет жасайтын (немесе оның мүмкіндігін қатты тартатын) шек. Типтеу объектінің бір классының орнына екіншісін қолданудан қорғануға,немесе тым болмағанда, осындай қолданысты басқаруға мүмкіндік береді.

Параллелизм — объектілердің пассивті жағдайда болуы мен арасында активті және пассивті объектілерді анықтау қасиеті.

Тұрақтылық — объектінің уақытта (осы объектіні жасаған процесске тәуелсіз) және/немесе кеңістікте (объектінің жасалған адрестік кеңістігінен жылжуы) өмір сүру қабілеті.

Объектілі-бағдарлы тәсілдің маңызды қасиеті - талаптардың жасалуы сатысынан бастап іске асу сатысына дейін ұйым қызметі моделі мен жобаланатын жүйенің ұйғарымы.Модельдердің ұйғарымдылығы талаптары абстракциялау, модульдық, жасаудың барлық сатысындағы полиморфизмға байланысты жасалады. Ерте кезеңнің модельдері іске асыру модельдерімен салыстыруға келеді. Объектілі модель бойынша ақпараттық жүйенің объектілері мен класстарына модельденетін пәндік саланың (ұйым) шынайы болмыстарының көрінісі бақыланады.

Объектілі-бағдарлы анализ бен жобалаудың (ОБАЖ) тәсілдерінің көбісі өзіне модельдеу тілі мен модельдеу процесін сипаттауды қосады. Модельдеу тілі – жобаларды сипаттау үшін тәсілмен қолданылатын нотация (графикалық). Графикалық нотация модельдерде қолданылатын графикалық объектілердің жинағынан тұрады; ол модельдеу тілінің синтаксисі болып табылады. Мысалы, класстар диаграммасының нотациясы класс, ассоциация мен көптілік ретінде мұндай элементтер мен түсініктер қалай ұсынылатындығын анықтайды. Процесс – жобаны жасауда орындауға тиісті қадамдардың сипаты.

UML Modeling Language) модельдеудің жүйеленген тілі – 1980 жылдардың аяғы мен 1990м-жылдардың басында пайда болған ОБАЖ тәсілдері ұрпақтарының мұрагері. UML жасау 1994 жылдың аяғында Гради Буч и Джеймс Рамбо А.Джекобсон көмегімен және Rational Software компаниясының қамқорлығымен Booch пен OMT (Object Modeling Technique) әдістерін біріктіру бойынша жұмысқа кіріскенде басталды. 1995 жылдың аяғында олар Method, 0.8 версиясы деп аталған біріккен тәсілдің алғашқы спецификациясын жасады. Сол жылы,1995-те, оларға OOSE (Object-Oriented Software Engineering) тәсілінің жасаушысы Ивар Якобсон қосылды.

Сөйтіп, UML Е. Буча, Д. Рамбо, А. Джекобсон мен И. Якобсон тәсілдерінің тура бірігуі мен унификациясы болып табылады, алайда оларды жаңа мүмкіндіктермен толықтырады. UML жасауда басты болып келесі мақсаттар саналды:

- қолданушыларға қолдануға дайын мағыналы модельдерді жасауға және олармен алмасуға мүмкіндік туғызатын визуалды модельдеу тілін ұсыну;
- нақты пәндік салада жүйелердің нақты көрінісі үшін кеңею мен специализация механизмдерін қарастыру;
- бағдарламалаудың нақты тілдері мен жасау процесінен тәуелсіздікті қамтамасыз ету;
- модельдеудің осы тілін түсінуге формалды негіз жасау (тіл бір уақытта түсіну үшін нақты және қолжетімді, артық формализмсіз болу керек);
- объектілі-бағдарлы аспаптық құрал нарығының өсуін ынталандыру;
- жақсы практикалық тәжірибені біріктіру.

Қазіргі таңда UML тілі модельдеудің стантартты тілі (OMG (Object Management Group) – объектілі-бағдарлы тәсіл мен технология саласында стандарттау бойынша ұйым) ретінде қабылданды және БӨ жасау индустриясында кең қолдауға ие болды. UML тілі барлық ірі компаниялармен – БӨ жасаушылар (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase және т.б.) – қаруланауға алынды. Сонымен қатар, CASE құралының барлық әлемдік өндірушілері, Rational Software (Rational Rose) басқа, өз өнімдерінде UML қолдайды (Paradigm Plus 3.6, System Architect, Microsoft Visual Modeler for Visual Basic, Delphi, PowerBuilder және т.б.). UML толық сипаттамасын http:// www.omg.org,http:// www.rational.com и http://uml.shl.com сайттарынан табуға болады.

UML жасаушылары оны бағдарламалық, ұйымдастырушылық-экономикалық, техникалық және баса жүйелерді анықтау, ұсыну, жобалау мен құжаттандырудың тілі ретінде ұсынады. UML тілі нақтырақ 14 бөлімде жазылған.

8.5. Жиналатын метрикалар, қолданылатын тәсілдер, стандарттар мен шаблондар

БӨ бобалау сатысында жоспарлы мерзім мен көлемнің нақтымен айырмашылығы, өткізілген шолу саны, табылған қателер мен дефекттер, сонымен қатар жобалаудың орташа еңбек шығыны мен өнімділігін бағалауды өткізу қажет.

Барлық алынған мәліметтерді жобалық топтың МБИ –де сақтау керек.

Қолданылатын аспап: құжаттарды даярлау жүйесі (мысалы, MS Word).

Қолданылатын әдістер мен стандарттар: ұйымдастыру процесі; ұйымның метрикалық бағдарламасы.

Қолданылатын шаблондар: жоғары деңгейлі жобалау нәтижелерін сипаттау; шолу бойынша есеп; жоба статусы жөнінде есеп.

Бақылау сұрақтар

1. Бағдарламалық өнімді жасаудың өміршеңдік циклінде жобалау сатысының мақсаты неде?
2. Жобалау құраушысы не болып табылады және ол үшін қандай атрибуттар анықталады?
3. «Алгоритм» ұғымына анықтама беріңіз.
4. Бағдарламалаудың құрылымдық тәсілі негізінде қандай негізгі принциптер жатыр?
5. «Бағдарламалық бітеу» ұғымына анықтама беріңіз. Оның мақсаты неде?
6. Бағдарламалаудың объектілі-бағдарлы тәсілінің кең таралуына бағдарламалаудың қандай мәселелері септігін тигізді?
7. Бағдарламалаудың объектілі-бағдарлы тәсілінің пайда болуына технологияның қандай жетістіктері себепкер болды?
8. CASE-құралдар немесе CASE-технологиялар не үшін керек?
9. «Объект» ұғымына анықтама беріңіз.
10. Абстракциялау дегеніміз не?
11. Инкапсуляция дегеніміз не?
12. «Класс» ұғымына анықтама беріңіз
13. UML тілінің мақсаты неде?
14. Жобалау сатысында қандай метрикалар жинайды?
15. Жобалау сатысында қандай тәсілдер, стандарттар мен шаблондар қолданады?

9 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ЖАСАУ САТЫСЫ

9.1.Кодтау

БӨ жасау сатысында төмендегі негізгі әрекеттер іске асады. Қодтау, тестілеу, БӨ анықтамалық жүйенің жасалуы, қолданушы құжаттамасын жасау, БӨ версиясы мен инсталляциясын жасау.

Кодтау дайын бағдарламалық өнімде жоғарыдеңгейлі және төмендеңгейлі жобалау нәтижелерін қайта жасау процесін көрсетеді. Басқаша сөзбен, кодтауда кез-келген тіл бола алатын бағдарламалаудың таңдалған тілі тәсілімен ЭЭ құралған моделінің сипаттамасы жасалады.

```
;
;1  2  3Description:                               *
;**
;*  File  Listing  template  for  CPU32  has  been *
developed ;* with MASM v.5.0.                      *
• 9r                                               *
;                                                  *
;*  Creation  Date:  21/04/95  From:  New          *
• 9r                                               *
;                                                  *
;*  Author:  IBS  Russia                           *
• 9r                                               *
;                                                  *
;*  Update  By  Modification:                      *
• 9r                                               *
;                                                  *
;*  06/05/95  AR  Add  changes  according  to  R.Soja *
and  ;*  B.Calvert  comments  /  letter  from      9r
06/05/95  /  for  ;*  MASM  5.0
• 9r
____________________________

;*                                                 *
;*         **** This Revision: 1.11 ****           *
;*                                                 *
;*  LastModified:  06/05/95 By: Alexandr  Rudakov  *
2
9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9r9
r
;
```

• Тілді таңдау тапсырыс беруші қалауымен, немесе шешілетін мәселе және жасаушының жеке тәжірибесі есебімен жүзеге асырылады.

Кодтау барысында таңдалған тіл стандартына сай болу керек, мысалы, С тілі үшін - ANSI C, ал C++ үшін - ISO/IEC 14882 «Standart for the C++ Programming Language».

Компанияда бағдарламау тілінің жалпы қабылданған стандартынан басқа бағдарламаны жазу ережелеріне қосымша талаптар жасалуы мүкін. Ол негізінен бағдарлама мәтінін ресімдеу ережесіне қатысты болады.

Компания стандарты мен ережесін қолдану жасаушы, жасау уақыты, аты мен бағыты, сонымен қатар конфигурацияны басқару үшін қажетті мәліметтер жөнінде ақпаратқа ие дұрыс жұмыс істейтін, оңай оқылатын, басқа жасаушыларға түсінікті бағдарлама жасауға мүмкіндік береді.

9.2. Тестілеу

Кодтау сатысында бағдарламашы бағдарламаны жазып, оны өзі тестілейді. Мұндай тестілеу модульдық деп аталады. БӨ тестілеуге қатысты барлық сұрақтар 10 бөлімде айтылған, ал бұл жерде БӨ жасау барысында қолданылатын тестілеу технологиясы жөнінде көрсетілген. Бұл технология *«әйнек жәшік» (glass box)* тестілеуі деп аталады; кейде оны «қара жәшік» (black box) классикалық ұғымына қарама-қарсы *«ақ жәшік» (white box)* тестілеуі деп те атайды.

«Қара жәшік» тестілеуінде бағдарлама ішкі құрылымы беймәлім объект ретінде қарастырылады. Тестілеуші мәліметтерді енгізеді және нәтижесін саралайды, бірақ ол бағдарламаның қалай жұмыс істейтінін білмейді. Тесттерді таңдай отырып, маман стандартты емес нәтижелерге әкелетін өз көзқарасына сай кіріс мәліметтер мен шарттарды іздейді. Ол үшін бірінші кезекте тестіленетін бағдарламаның қателері шығуы мүмкін кіріс мәліметтерінің әрбір классының өкілдері қызығушылық туғызады.

«Әйнек жәшікті» тестілеуде жағдай мүлдем басқаша. Тестілеуші (бұл жағдайда бағдарламашы) өзі толық қол жеткізуге ие бастапқы кодты білуге негізделіп, тесттер жасайды. Нәтижесінде ол келесі артықшылықтар иеленеді:

1. Тестілеу бағыты. Бағдарламашы бағдарламаны бөлік бойынша тестілей алады, тестіленетін модульді шақыратын және оған бағдарламашыны қызықтыратын мәліметтерді беретін арнайы тестілеу бағдарламашаларын жасай алады. Жеке модульді «әйнек жәшік» ретінде тестілеу оңай.

2. Кодты толық қамту.Бағдарламашы әрдайым әрбір тестте қандай фрагменттер жұмыс істейтітін анықтай алады. Ол кодтың қандай тармақтары әлі тестіленбегенін көреді де, сөйтіп оларға тестілену үшін жағдай таңдап бере алады. Өткізілген тесттермен бағдарламалық кодтың қамту деңгейін бақылау төменіректе көрсетілген.

3. Пәрмен ағынын басқару мүмкіндігі.Бағдарламашы келесінің бағдарламасында қандай функция атқарылу керек және оның ағымдағы жағдайы қандай екендігін әрдайым біледі. Бағдарлама өзі ойлағандай жұмыс істеп тұрғандығын тексеру үшін бағдарламашы оның орындалу жүрісі жөнінде ақпаратты көрсететін ретке келтіруші пәрмен қоса алады немесе ол үшін реттеуші деп аталатын арнайы бағдарламалық құрал қолдана алады. Реттеуші өте көп пайдалы нәрселер істей алады: бағдарлама пәрмендерін қарайды және оның орындалу тізбегін ауыстырады,оның ауыспалылары мен жадтағы адрестер жөнінде мәліметтерді көрсетеді.

4. Мәліметтердің тұтастығын бақылау мүмкіндігі. Мәліметтердің әрбір элементін бағдарламаның қандай бөлігі өзгерту керектігі бағдарламашыға мәлім. Мәліметтер жағдайын бақылай отырып (сондай реттеуші көмегімен), ол қажетті емес модульдермен мәліметтердің өзгеруі, олардың дұрыс емес интерпретациясы немесе сәтсіз ұйымдастырылу сияқты қателерді таба алады. Бағдарламашы өз бетімен тестілеуді автоматтандыра алады.

5. Ішкі шектес нүктелерді көру. Бастапқы кодта көзге көрінбейтіндей жасырылған бағдарламаның шектес нүктелері көрінеді.Мысалы, кейбір әрекеттерді жасау үшін әртүрлі алгоритмдер қолданылуы мүмкін, және кодқа қарамай, бағдарламашы олардың қайсысын таңдағанын білу мүмкін емес. Тағы бір типтік мысал ретінде кіріс мәліметтерін уақытша сақтау үшін қолданылатын буфердің толып кетуі мәселесі болуы мүмкін. Бағдарламашы мәліметтердің қандай санында буфер толып кетуі мүмкін екендігін бірден айтып, мыңдаған тесттер өткізудің қажеті жоқ.

6. Таңдалған алгоритммен анықталатын тестілеу мүмкіндігі. Өте қиын есептеуіш алгоритмдерді қолданатын мәліметтерді өңдеуді тестілеу үшін арнайы технологиялар қажет болуы мүмкін. Классикалық мысал ретінде матрицаның қайта жасалуы мен мәліметтердің іріктелуін келтіруге болады. Тестілеушіге бағдарламашыға қарағанда, қандай алгоритмдер қолданылу керектігін білген маңызды, сондықтан арнайы әдебиетке жүгінуге тура келеді.

«Әйнек жәшікті» тестілеу – бағдарламалау процесінің бөлігі. Бағдараламашылар бұл жұмысты тұрақты түрде жасайды,

олар әрбір модульді оның жазылуынан кейін тестілейді, кейін оның жүйеге бірігуінен кейін тағы тестілейді.

Модульдік тексеруді жасау барысында не құрылымдық, не функционалды тестілеу немесе екеуінің де технологиясын қолдануға болады.

Құрылымдық тестілеу «әйнек жәшікті» тестілеудің бір түрі болып табылады. Оның басты идеясы тестіленетін бағдарламалық жолды дұрыс таңдау болып табылады. Оған қарама-қарсы *функционалды* тестілеу «қара жәшікті» тестілеу категориясына жатқызылады. Бағдарламаның әрбір функциясы оның кіріс мәліметтері мен шығыс анализін енгізу арқылы тестіленеді. Бұл ретте бағдарламаның ішкі құрылымы өте сирек есептеледі.

Құрылымдық тестілеу мықты теориялық негізге ие болғанмен, көптеген тестілеушілер функционалды тестілеуді қалайды. Құрылымдық тестілеу математикалық модельдеуге жақсы беріледі, бірақ ол оның тиімдірек екендігін білдірмейді. Технологияның әрқайсысы басқасын қолдану жағдайында өткізілетін қателерді табуға мүкіндік береді. Осы көзқараста оларды бірдей тиімді деуге болады.

Тестілеу объектісі ретінде тек қана бағдарламаның толық жолы ғана болмайды (ол старттан аяғына дейін орындайтын пәрмендердің тізбектілігі), сонымен қатар оның жеке бөліктері де саналады. Бағдарламаның орындалудың барлық мүмкін болатын жолдарын тестілеу мүмкін емес. Сондықтан тестілеу бойынша мамандар барлық мүмкін болатын жолдардың ішінен тестілеу міндетті топтарды бөледі. Таңдау үшін олар тесттердің шынайы санын (көп болса да) анықтайтын қамту белгісі (coverage criteria) деп аталатын арнайы белгімен қолданады. Мұндай белгілерді кейде қамтудың логикалық белгісі немесе толықтық белгісі деп атайды.

Тестілеушілер жиі қолданатын үш белгіні қарастырайық:тармақ қамтуы, бұтақтану қамтуы мен шарт қамтуы. Тестілеу осы белгілерге сай жасалса, ол жайлы жолдарды тестілеу дейді.

Тармақ қамтуы белгісі – үшеуінің ішінде ең әлсізі. Ол кодтың әрбір тармағы тым болмаса бір рет жасалуын талап етеді. Көптеген бағдарламашылар оны қолдану және осы талапты қолдануларына мұрша болмаса да, бағдарламаны тестілеу үшін ол жеткіліксіз. Егер тармақ шешім қабылдау операторына ие болса, мәнді шешудің барлық басқарулары тексерілу керек. Мысал ретінде келесі код фрагментін қарастырайық:

```
IF (A < B and C = 5)
THEN ӘЛДЕНЕ жасау
SET D = 5
```

Осы кодты тексеру үшін келесі төрт нұсқаны анализдеу керек.

а) A < B и C = 5 (ӘЛДЕНЕ жасалады, кейін D-ға 5 мағынасы беріледі);

б) A < B и C ф 5 (ӘЛДЕНЕ жасалмайды, D-ға 5 мағынасы беріледі);

в) A > B и C = 5 (ӘЛДЕНЕ жасалмайды, D-ға 5 мағынасы беріледі);

г) A > B и C ф 5 (ӘЛДЕНЕ жасалмайды, D-ға 5 мағынасы беріледі).

Кодтың барлық үш тармағын орындау үшін а) нұсқасын тексеру жеткілікті.

Тестілеудің негіз салушы тәсілінде – бұтақтану қамтуы белгісі бойынша – бағдарламашы а) нұсқасын және қалған үш нұсқаның біреуін тексереді. Бұл әдістің мағынасы IF операторының жасалған және жасалмаған шарттарында бағдарлама әрекеттері тексерілунде жатыр. Сөйтіп, бағдарлама кодтың барлық тармақтарынан ғана емес, сонымен қатар мүмкін болатын барлық бұтақтанудан өтеді.

Кейде бұтақтану қамтуын кодтың толық қамтуы деп атайды. Бірақ бұл термин дұрыс емес: бұтақтану қамтуы тестілеудің толықтығына дәмелене алмайды, өйткені жақсы жағдайда ол бағдарламада бар қателердің жартысын ғана таба алады.

Шарт қамтуы белгісі одан да қатал болып саналады. Бұл белгі бойынша әрбір логикалық шарттың құрамдас бөліктерін тексеру міндетті. Келтірілген мысалда ол барлық төрт нұсқаны тексеруді білдіреді.

Бағдарлама жолдарын тексеру аяқталған болып саналады, егер қамтудың таңдалған белгісі толығымен жасалса. Бұл процесті автоматтау үшін бағдарламалық код пен жолдарды тестілеуге келетін есептеуіш сандарды анализдейтін, сонымен қатар қаншауы өткізілгенін санайтын бағдарламалар жасалды. Мұндай бағдарламаларды қамтуды тексерудің құралы (execution coverage monitors) деп атайды.

Әдетте, жолдарды тестілеу барысында әртүрлі мәліметтерде бір жолды тексеру қабылданбайды.Келтірілген мысалдың б),в) және г) нұсқалары маңызды болғанымен, қамтуды тексеру құралының көбісі оны тексеруді уақытты бос жоғалту деп санайды. Үшеуі де бір оператордан басталады және бір тізбектегі пәрмендерді орындауға әкеледі, сондықтан оларды тексеру бір жолды үш рет тексерумен тең болады.

Қамту белгілері пайдалы болғанымен, қателерді тиімді табу үшін жолдарды тестілеу ғана жеткіліксіз. Сөйтіп, И.Гуденаф (Goodenough) пен К. Герхарт (Gerhart) код тармағының өтуі ондағы қатенің бар екендігін білдірмейтіндігі жөнінде классикалық мысал келтірді. Бағдарламаның кедесі тармағын

қарастырайық: SET A = B/C

Ол айтарлықтай табысты орындалады, егер С ф 0 болса. Егер С = 0 болса, онда бағдарлама не қате жөнінде айтады, не өз жұмысын тоқтатады, не «қатып қалады». Осы екі нұсқа арасындағы айырмашылық жолда емес, мәліметтерде.

Де Милло (De Millo) бағдарламалық жолды қателерге сезімтал дейді, егер оны өту барысында қателер пайда болса. Егер осы жолды өтуде қателер міндетті түрде пайда болса, онда ол жол қате табаын жол деп аталады. Бағдарламаның келтірілген тармағы арқылы өтетін кез-келген жол қателерге сезімтал, ал қате айнамалы С-дің нөлдік мәнінде ғана пайда болады. Осындай қателерді табу үшін «қара жәшік» тестілеу технологиясы бағдарламалық кодқа қарағанда жақсы келеді.

Бағдарламалық жолдарды тестілеу технологиясын формалды сипаттауды Реппс (Rapps) пен С. Вейакерден (Weyuker) табуға болады,мәселені егжей-тегжейлі зерттеуді - Р. Бейзерден (Beizer), ал қамту белгісін жете талқылауды - Д. Майерсте (Myers).

Кез-келген жүйе бөлік бойынша жасалады – процестер немесе модульдер жинағы ретінде. Оны осылай тестілеуге болады, яғни бірінші жеке бөліктерді тексереді, кейін олардың өзара әрекетін. Мұндай тестілеу көтерілуші (bottom-up)деп аталады.

Бағдарламаның жеке элементтері дұрыс екендігін біліп, маман олардың біріккен жұмысын тестілеуге кіріседі. Бұл жерден олар бірге жұмыс істеуден бас тартатындығы байқалуы мүмкін. Мысалы, егер бағдарламашы байқаусыздан шақырылатын функция параметрлерінің орнын алмастырса, онда шақыруды жасауда қате шығады. Және ол тек екі функцияның біріккен жұмысын – шақыратын және шақырылатын- тексеру барысында ғана шығады.

Бағдарламалық модульдердің біріккен жұмысын тестілеуді интеграциялық деп атайды.Мұндай тестілеуде бірінші модульді жұптарға біріктіреді, кейін үлкен блоктарға, және ол барлық модуль біріккен жүйеге бірікпегенше жалғасады. Барлық жүйенің интеграциялық тестілеуін тестілеуші жасайды, бірақ бір міндеттің интеграциялық тестілеуін – жасаушы бағдарламашы.

Көтерілуші тестілеу – қателерді ауыздықтаудың керемет тәсілі.Егер қате жалғыз модульді тестілеу барысында шықса, онда ол соның ішінде екендігі анық, және оның қайнаркөзін табу үшін бүкіл жүйенің кодын анализдеудің керегі жоқ. Ал егер қате екі алдын-ала тестіленген модульдің біріккен жұмысында шықса, онда мәселе олардың интерфейстерінде жатыр. Көтерілуші тестілеудің тағы бір артықшылығы – оны жасайтын бағдарламашы өте таң шеңберлі салаға назарын аударады (жалғыз модульде, жұп модуль арасында мәліметтер жіберу және т.б.). Соның арқасында тестілеу мұқият жасалады

және үлкен ықтималдықпен қателерді табады.

Көтерілуші тестілеудің басты кемшілігі – тестіленетін модульды шақыратын арнайы код-қабықшаның жазылу қажеттілігі. Егер ол өз кезегінде басқа модульді шақырса, онда ол үшін бітеуіш жазу керек. Бітеуіш – сол мәліметтерді қайтаратын шақырылатын функцияның ұқсастырушысы, ол басқа ешнәрсе істемейді.

Қабықша мен бітеуіштерді жазу жұмысты бәсеңдетеді, ал соңғы өнім үшін олар мүлдем қажетсіз, бірақ бір рет жасалған бұл элементтер бағдарламаның әрбір өзгерісінде қайталанып қолданыла береді. Бітеуіш пен қабықшаның жақсы жиынтығы – ол тестілеудің өте тиімді құралы.

Көтерілуші тестілеуге қарсы тұтастық тестілеу жүйенің толық біріктірілуіне дейін оның жеке модульдері мұқият тестілеуден өтпейді деп санайды. Мұндай стратегияның артықшылығы – қосымша кодтың жазылу қажеттілігінің болмауы. Көптеген бағдарламашылар уақытты үнемдеу үшін тестілеудің осы тәсілін таңдайды, себебі олар бір рет тесттердің ауқымды жинағын жасап, және соған негізделіп бүкіл жүйені бір рет тексеріп шыққан жөн деп санайды. Бірақ мұндай көрініс келесі себептер бойынша қате.

1. Қате көзін табу өте қиын. Бұл басты мәселе. Себебі ешқандай модуль жақсы тексерілмейді, олардың көбісінде қателер бар.Сондықтан мәселе қай модульде қате шыққанында емес, процеске қатысқан барлық модульдердің қай қатесі алынған нәтижеге алып келгендігінде. Бірнеше модуль қателері жиналғанда қате көзін ауыздықтау қиынға соғады.

Сонымен қатар, бір модульдің қатесі басқа модульді тестілеуді блоктауы мүмкін. Функцияны қалау тестілеуге болады, егер оның шақырушы модулі жұмыс істемесе? Бұл функцияға бағдарлама-қабықшаны жазбаса, модульдің ретке келуін күтуге тура келеді, ал ол өте көп уақыт алуы мүмкін.

2. Қатенің жөнделуін ұйымдастыру қиын.Егер бағдарламаны бірнеше бағдарламашы жазып (үлкен жүйелерде солай болады) және бұл ретте қате қай модульде екендігі беймәлім болса, онда ол қатені кім іздеп, оны жөндейді? Бірінші бағдарламашы екіншісіне көрсетеді, ол оның кодының қатысы жоқ екендігін біліп, жауапкершілікті біріншісіне артады, нәтижесінде жасау жылдамдығы күрт төмендейді.

3. Тестілеу процесін автоматтау қиын. Бір қарағанда тұтастық тестілеудің артықшылығы ретінде көрінетін қабықша мен бітеуіш жасау қажеттілігінің болмауы негізінде кемшілік болып саналады. Жасау барысында бағдарлама күнделікті өзгереді және оны қайта-қайта тестілеуге тура келеді, ал қабықшалар мен бітеуіштер бұл біробразды еңбекті автоматтауға жәрдем

береді.

Бағдарламашылардың біреуі тұтастық тестілеуді таңдаса, онда осы тәсілдің артықшылықтарын тек өзі ғана көре алады деп есептеуге болады. Кейбір бағдарламашылар тестілеу тиімділігі жөнінде мүлдем «бастарын ауыртпайды».

Олар үшін бастысы – тез арада басшылыққа жұмыстың біткендігі жөнінде баянат беру, тіпті расында ешнәрсе жұмыс істемесе де. Егер осыдан кейін жобада қиындықтар туса, онда олар Мэрфи заңын, тестілеушілерді, жолболмаушылықты, басқаша айтқанда өздерінен басқалардың бәрін айыптайды, себебі олардың ойынша олар өздерінің жұмысын тиімді және уақытында орындады.

Тестілеуді ұйымдастырудың тағы бір принципі бар, ол бағдарлама көтерілуші тәсіліндегідей, толығымен емес, бөліктер бойынша тестіленсе, тек қозғалыс бағыты өзгереді – бірінші модульдер иерархиясының ең үстінгі деңгейі тестіленеді, ал содан тестілеуші ақырындап төмен түседі. Мұндай тестілеу бәсеңдеуші (top-down) деп аталады. Көтерілуші де, бәсеңдеуші тестілеулерді инкрементальды деп атайды.

Бәсеңдеуші тестілеуде қабықшаларды жазу қажеттілігі жойылады, бірақ бітеуіштер қаладыу Тестілеу барысында қабықшалар кезекпен шынайы модульдерге алмастырылады.

Екі инкрементальды тестілеудің қайсысы тиімді екендігі жөнінде мамандардың ойлары әртүрлі. Р. Иордан (Yourdon) бәсеңдеуші тестілеуші жақсы деп дәлелдейді, ал Д. Майерс (Myers) екі әдістің де өз артықшылықтары мен кемшіліктері болғанымен, толығымен көтерілуші тестілеу жақсы дейді. Д. Данның (Dunn) ойы бойынша, бұл тәсілдер шамамен баламалы.

Тәжірибеде тестілеу стратегиясын таңдау мәселесі әдетте оңай шешіледі: әрбір модуль мүмкіндігі бойынша жазылудан кейін тестіленеді, нәтижесінде бағдарламаның бір бөлігін тестілеу тізбектігі көтерілуші, ал қалғандары бәсеңдеуші болуы мүмкін.

Статикалық тестілеуде бағдарламалық код жасалмайды – ол тек логикалық анализ жолымен тестіленеді.

Статикалық анализ үшін көптеген инструменталды құралдар бар. Синтаксистік қате немесе жарамайтын операцияны байқап, компилятор сәйкес хабар береді. Пайдалы хабарлар қатарын (айнымалы және басқа объектілердің қайталанатын аттары, жарияланбаған айнымалылар мен функцияларға сілтемелер жөнінде) құрастырушы да береді. Бағдарламаның статикалық анализі адамдармен де жасалына алады. Олар бастапқы кодты оқып, оны талқылауы мүмкін, және, әдетте, көптеген қателер табады.

9.3. Бағдарламалық өнімнің ақпараттық жүйесін жасау. Пайдаланушы құжаттамасын құрау

Жобаның бір қызметкерін құжаттаманың техникалық редакторы етіп тағайындаған жөн. Бұл қызметкер басқа да жұмыс жасай алады, бірақ оның басты міндеті құжаттаманы анализдеу, тіпті оны басқа қызметкерлер жасағанның өзінде де.

БӨ жасауда бірнеше адам атсалысуы мүмкін, бірақ олардың ешқайсысы оның сапасы үшін толық жауапкершілік иемденбейді. Нәтижесінде БӨ оны бірнеше адам жасағаннан жеңбейді, қайта жасаушылардың әрқайсысы жауапкершілікті басқасына артқызып, жұмыс бөлігінің оның әріптесі жасайды деп күткеннен жеңіледі.Бұл мәселені редакторды тағайындау арқылы шешуге болады, себебі ол техникалық құжаттың сапасы мен дәлдігі үшін толық жауапты болады.

БӨ ақпараттық жүйесі пайдаланушыны басқару үшін жасалған материалға негізделіп жасалады. Оны жасап, құрастыратын осы жұмыс үшін жауапты адам. Ол адам ретінде техникалық редактор, не техникалық редактормен бірге жасаушылардың бірі саналуы мүмкін.

Жақсы құжатталған БӨ-да келесі артықшылықтар болады.

1. Қолдану жеңілдігі. Егер БӨ жақсы құжатталса,онда оны қолдану да оңай болады. Пайдаланушылар оны тез зерттеп,қателер аз болады, нәтижесінде өз жұмыстарын жылдам, әрі тиімді жасайды.

2. Техникалық қолдаудың құнының аздығы. Пайдаланушы оған қажетті әрекеттерді қалай жасау керек екендігін түсінбесе, ол техникалық қолдау қызметіне БӨ жасаушысына хабарласады. Мұндай қызметті қамтамасыз ету өте қымбатқа түседі. Жақсы басшылық өздері пайдаланушыларға мәселені шешуге және және техникалық қолдау тобына хабарласуды азайтуға жәрдемдеседі.

3. Жоғары сенімділік. Түсініксіз немесе ұқыпсыз құжаттандыру БӨ сенімділігін төмендетеді, пайдаланушылар көп қате жібергендіктен, оларға қатенің себебін табу мен оның салдарын шешу қиынға соғады.

4. Сүйемелдеу жеңілдігі. Пайдаланушылармен жасалатын қателерді анализдеуге өте көп ақша мен уақыт кетеді. БӨ жаңа шығарылымдарымен енгізілетін өзгерістер көбінесе ескі функцияның интерфейсін ауыстыруында болады. Олар пайдаланушылар БӨ-ны қалай қолдану керектігін ұғынсын және техникалық қолдау қызметіне хабарласуларын тоқтатсын деген мақсатпен енгізіледі. Жақсы басшылық айтарлықтай деңгейде бұл мәселені шешеді, ал нашар басшылық оны керісінше одан

да қиын етеді.

5. Жеңілдетілген орнату. БӨ өнімін сатып алғаннан кейін пайдаланушы оны өз компьютеріне орнату керек. Бұл процесс толықтай автоматтандырылған болса да, пайдаланушыға бірқатар сұрақтарға жауап беріп, БӨ жиынтығы мен құрастырушыларының орналасуы мен оның функциясына қатысты шешім қабылдауға тура келеді. Ал пайдаланушының БӨ орнату бойынша тәжірибесі болмауы мүмкін және бағдарламаның кейбір сұрақтары оны бұрышқа тақауы мүмкін. Компания үшін ол тағы техникалық қолдауға шығынмен байланыстырылады. Сондықтан БӨ орнату бойынша нақты және түсінікті нұсқаулықтар оның құжаттамасының ең маңызды құрамдас бөлігі болып саналады.

Орнату бойынша нұсқаулықтан басқа бағдарламалық қамтамасыз етуде жүйеден БӨ жою бойынша да нұсқаулық болу керек. Құжатаммада сонымен қатар баптау параметрлерін қалай өзгерту керек, БӨ құрастырушысын қосу немесе жою мен бұрынғысының үстінен оның жаңа нұсқасын енгізу бойынша түсініктемелер болуы керек.

6. Коммерциялық табыс. Құжаттама сапасы БӨ коммерциялық табысын анықтайтын факторлардың бірі болып табылады. Жақсы құжаттамасы бар дилерлерге БӨ-нісатып алып алушыларға ұсынған және оның мүмкіндіктері жөнінде айтқан оңай. Кәсіби баспасөзде басылатын бағдарламалық қамтамасыз етудің көп образында құжаттамаға аса көңіл бөлінеді.

7. Ақпараттың анықтығы. Құжаттамада пайдаланушыны жаңылыстыратын және оларды бос уақыт пен күш жұмсауға мәжбүрлейтін қате ақпарат болмау керек.

9.4. Бағдарламалық өнімнің нұсқасы мен орнатуын жасау

БӨ орнатуын жасау. Бұл әрекет пайдаланушы компьютерлеріне БӨ орнату процесін автоматтауға мүмкіндік береді. Және бұл ретте оларға орнатудың әртүрлі сценарийін таңдау мен оның кезекті жұмысының дұрыстығын қамтамасыз ететін мүмкіндік береді. Орнату барысында БӨ өзі жұмыс істеуге тиісті бағдарламалық ортаға «енеді». Сонымен қатар, БӨ жасаушыларында санкцияланбаған «пираттық» орнатуларға (мысалы, БӨ сериялық нөмірін тексеру арқылы) тыйым салуға мүмкіндік пайда болады. БӨ орнату процесі пайдаланушы басшылығында немесе жеке брошюрада міндетті түрде жазылу керек.

БӨ орнату процесінде әртүрлі мәселелер туындауы мүмкін. Олардың ішінде келесілер жиі кездеседі.

1. БӨ орнатылатын орта ол жобаланған ортаға сай келмейді. Мысалы, БӨ операциялық жүйенің тек белгілі нұсқасын ұсынатын функциялар қолдана алады. Алайда операциялық жүйенің нұсқасы орнатылатын БӨ ағымдағы ортасын анықтайтын операциялық жүйенің өзі ондай функцияға ие болмауы мүмкін. Бұл жағдайда орнатылудан кейін БӨ мүлдем жұмыс істемеуі мүмкін немесе оның кейбір функциялар жүзеге аспай қалады.Бұл мәселені шешу үшін пайдаланушы басшылығында не операциялық жүйенің нұсқалары, не БӨ дұрыс жұмыс істейтін операциялық жүйенің өзі аталып өтілу керек.

2. Жаңа БӨ өзі орнатылған ұйымда ескісімен қатар бола алады, жаңа БӨ керекті жолмен жұмыс істеуді бастамағанша. Ескімен бірге жаңа БӨ орнату айтарлықтай қиындықтарға әкелуі мүмкін, әсіресе, егер жаңа және ескі БӨ толықтай тәуелсіз болмай, кейбір орталық құрауыштарға ие болса. Жаңа БӨ ескісін жоймайынша орнатылмайтын жағдайлар болады. Бұл ретте жаңа БӨ сынауын ескі БӨ жасамаған кезде ғана жасауға болады. Егер мұндай жағдай болса,онда оны жою бойынша нақты ұсыныстармен пайдаланушы басшылығында жазылу керек.

БӨ нұсқасын жасау мен жеткізуді басқару. Мұндай басқару БӨ барлық нұсқасын сәйкестендіру мен жеткізу мен оны байқау үшін қажет. БӨ нұсқасы мен жеткізуді басқаруға жауап беретін арнайы бөлінген жоба қызметкері (конфигурацияны басқаруға жауапты) БӨ ескілерін іздеу мен жаңа нұсқасын жасау немесе БӨ жеткізу процедураларын жасайды, сонымен қатар, өзгерістер өз бетімен жасалмауын бақылайды. Нұсқаларда өзгерістер жөнінде ақпарат болған жағдайда ғана конфигурацияны басқару бойынша жауаптымен енгізіліп, нұсқалардың келісімділігін кепілдеуге болады.

БӨ *нұсқасы* деп сол БӨ-нің басқа данасынан ерекшеленетін БӨ данасын атайды. Жаңа нұсқалар функционалды мүмкіндіктер, тиімділік немесе ескі нұсқада болған қателердің жоқтығымен ерекшеленеді. Жаңа нұсқалар бірдей функционалды мүмкіндіктерге ие, алайда олар аппараттық немесе бағдарламалық қамтамасыз етудің әртүрлі конфигурациясымен жасалады. Егер нұсқалар арасында айырмашылықтар болмашы болса, онда олар бір нұсқаның *түрлері* деп аталады.

Шығыс нұсқасы (release), немесе *БӨ жеткізілімі* – тапсырыс берушіге жеткізілетін нұсқа. Әрбір шығыс нұсқасында не жаңа функционалды мүмкіндіктер міндетті түрде болады, не ол жаңа платформаға сай жасалады. Нұсқа саны әдетте жеткізу санынан артады, себебі нұсқалар ішкі қолданыс үшін жасалады және тапсырыс берушіге жеткізілмейді.

Қазіргі таңда нұсқаларды басқаруды қолдау үшін көптеген

әртүрлі CASE-құралдар жасалды, олардың көмегімен әрбір нұсқаны сақтауды басқару мен БӨ құрауыштарына рұқсатты бақылау жүзеге асырылады. Құрауыштар БӨ-ден оларға өзгеріс енгізу үшін алынады. БӨ-ге өзгертілген құрауыштарды енгізгеннен кейін жаңа нұсқа шығады, ол үшін нұсқаны басқару жүйесі көмегімен жаңа атау жасалады.

Нұсқаны сәйкестендіру. Кез-келген үлкен БӨ жүздеген құрауыштардан тұрады,олардың әрқайсысы бірнеше нұсқаға ие. Нұсқаны басқару процедурасы құрауыштың әрбір нұсқасын нақты сәйкестендіру керек. Нұсқаны сәйкестендірудің үш негізгі тәсілі бар.

1. Нұсқа нумерациясы.Әрбір құрауыш нұсқаның бірегей және анық нөміріне ие. Сәйкестендірудің бұл тәсілі кең пайдаланылады.

2. Атрибуттар мағынасына негізделген сәйкестендіру. Әрбір құрауыш әртүрлі нұсқалар үшін бірегей болмайтын атпен және құрауыштың әрбір нұсқасы үшін әртүрлі атрибуттар мәнінің жиынтығымен сәйкестендіріледі. Басқаша сөзбен, құрауыш нұсқасы аттар комбинациясы мен атрибуттар мәнінің жиынтығымен сәйкестендіріледі.

3. Өзгерістерге негізделген сәйкестендіру. БӨ-нің әрбір нұсқасы өткен тәсілдегідей аталады, және оған қосымша ретінде БӨ-нің аталмыш нұсқасында жүзеге асырылатын өзгерістерге сұранысқа сілтемелер беріледі.Сөйтіп, БӨ нұсқасы құрауыштарды іске асырылған аттар мен сол өзгерістермен сәйкестендіріледі.

Нұсқа нумерациясы.Құрауыш атауы немесе БӨ-ге нұсқа нумерациясының ең оңай сызбасы бойынша нұсқа нөмірі қосылады. Мысалы, MASM TPU 4.3 белгісі келесіні білдіреді: 4.3 нұсқасы TPU ассемблерінікі. Бірінші нұсқа әдетте 1.0 нөмірін иемденеді, оның қалған нұсқалары – 1.1,1.2 және т.б. Жаңа мөлшерлеме қандай сатыда жасалады - 2.0; бұл нұсқаның түрлер нумерациясы қайтадан басталады: 2.1,2.2 және т.б. Нумерацияның мұндай линиялық сызбасы нұсқаны жасау тізбектігі жөнінде тәсілге негізделеді. Нұсқаны сәйкестендірудің мұндай тәсілі нұсқаны бақылаудың көптеген бағдарламалық тәсілдерін қолдайды.

Нұсқаны сәйкестендірудің аталмыш тәсілі жеткілікті оңай, алайда олардың арасында айырмашылық пен өзгеріске сұраныс арасындағы байланыс пен нұсқаларды бақылау мақсатымен нұсқаларды сәйкестендіру үшін қажетті ақпаратты талап етеді. Сондықтан БӨ-нің жеке нұсқасын немесе оның құрауышын іздеу айтарлықтай қиын болуы мүмкін, әсіресе конфигурацияның мәліметтер базасы мен нұсқаны сақтау жүйесі арасында интеграцияның болмауы жағдайында.

Атрибуттар мәніне негізделген сәйкестендіру.

Нұсқаны анық атау тәсілінің негізгі мәселесі бұл жағдайда нұсқаны сәйкестендіру үшін қолдануға болатын белгілер көрінбейтіндігінде жатыр. Мұндай белгі ретінде келесілер саналады: тапсырыс беруші аты; бағдарламалау тілі; жасау жағдайы; аппараттық платформа; жасау күні.

Егер әрбір нұсқа атрибуттардың біріккен жиынтығымен анықталса, онда қолда бар нұсқалардың кез-келгеніне негізделген жаңа нұсқаларды қосу қиын болмайды, себебі олар атрибуттардың мәндерінің біріккен жиынтығымен сәйкестендіріледі. Бұл ретте жаңа нұсқаның көптеген атрибуттарының мәні бастапқы нұсқа атрибутының мәніне сәйкес келеді; сөйтіп, нұсқалар арасында өзара қатынасты бақылауға болады. Нұсқаны іздеу атрибуттар мәніне негізделіп жасалады. Бұл ретте «ең соңғы нұсқа», «анықталған дата арасында жасалған нұсқа» сияқты сұраныстар болуы мүмкін.

Атрибуттар мәніне негізделген сәйкестендіру нұсқаны басқару жүйесімен тікелей қолданыла алады. Алайда нұсқа атының бөлігін ғана қолдану кең таралған, бұл ретте конфигурацияның мәліметтер базасы атрибуттар мәні, БӨ нұсқалары мен құрауыштар арасында байланысты қолдайды.

*Өзгерістерге негізделген сәйкестендіру.*Атрибуттар мәніне негізделген сәйкестендіру нұсқаны іздеуде оның атрибуттарын білу талап етілгенде нумерацияның ең оңай тәсілімен нұсқаны іздеу мәселесін жояды. Бірақ бұл жағдайда нұсқалар мен өзгерістер арасындағы өзара байланысты тіркеу үшін өзгерісті басқарудың жеке жүйесін қолдану қажет.

Өзгерістерге негізделген сәйкестендіру құрауыштарына қарағанда БӨ-ге қолданылады; жеке құрауыштардың нұсқалары конфигурацияны басқау жүйесі пайдаланушыларынан жасырылған.БӨ-дегі әрбір өзгеріс БӨ-нің аталмыш өзгеруін іске асыратын жеке компоненттердегі өзгерістерде көрсетілетін массивті өзгеріспен сипатталады. Өзгеріс массиві барлық қажетті өзгерістер іске асқан БӨ нұсқасын жасау үшін тізбекті түрде қолданылуы мүмкін. Бұл жағдайда нұсқаны анық белгілеу қажет етілмейді. Конфигурацияны басқаруға жауапты өзгерісті басқару жүйесі арқылы нұсқаны басқару жүйесімен жұмыс істейді.

БӨ-нің бірнеше өзгеріс массивтерін қолдану келісіліп жасалады,себебі өзгерістің жеке массивтері сәйкес болмай қалуы мүмкін және олардың кезекті қолданылуы жұмысқа қабілетсіз БӨ пайда болуына алып келуі ықтимал. Сонымен қатар, өзгерістер массиві қақтығыса алады, егер олар бір құрауышта әртүрлі өзгерістер қарастырса. Бұл мәселелерді шешу үшін өзгеріс негізінде сәйкестендіруді қолдайтын нұсқаларды басқару құралы қолданылады,ол БӨ нұсқаларының келісілген тізбектігінің нақты ережелерін бекітуге жол береді.

Ол, өз кезегінде, өзгеріс массивтерін құрамдастыру әдістерін шектейді.

9.5. Жиналатын метрикалар, қолданылатын тәсілдер, стандарттар мен шаблондар

БӨ жасау сатысында жоспарлы мерзім мен көлемнің шынайымен салыстырғандағы айырмашылық, өткізілген шолулар саны, табылған қателер мен дефекттер, сонымен қатар орташа еңбек шығыны мен жасау өнімділігін бағалау керек.

Барлық алынған мәліметтерді жобалық топтың ИМБ сақтау керек.

Қолданылатын құрал: құжаттарды әзірлеу жүйесі (мысалы, MS Word).

Қолданылатын тәсілдер мен стандарттар: ұйымдастыру процесі; кодтау стандарты.

Қолданылатын шаблондар: пайдаланушы құжаттамасы; шолу бойынша есеп; жоба статусы жөнінде есеп.

Бақылау сұрақтары

1. Бағдарламалық өнімді жасау сатысының мақсаты неде?
2. Кодтау процесінің мәні неде?
3. «Қара жәшікті» тестілеу нені білдіреді?
4. Құрылымдық тестілеудің функционалдыдан айырмашылығы неде?
5. Қамту белгілерін атаңыз және оларға анықтама беріңіз.
6. Көтерілуші тестілеудің біртұтас тестілеу стратегиясынан айырмашылығы неде?
7. Біртұтас тестілеу стратегиясының қандай кемшіліктері бар?
8. Төмендеуші тестілеу принципін түсіндіріңіз.
9. Статикалық тестілеу қалай жасалады?
10. Бағдарламалық өнім мен пайдаланушыны басқарудың анықтамалық жүйесін жасау процесі қалай ұйымдастырылған?
11. Бағдарламалық өнімді орнатуды жасаудың мақсаты неде?
12. Орнату барысында қандай мәселелер туындауы мүмкін және олармен қалай күресу керек?
13. Бағдарламалық өнім нұсқаларын басқару не үшін керек?
14. Бағдарламалық өнім нұсқасын басқарумен кім айналысады?
15. Бағдарламалық өнімнің нұсқасы, нұсқа белгісі, шығыс нұсқасы деп не аталады?
16. Бағдарламалық өнім нұсқасын сәйкестендіру тәсілдерін атаңыз.

Бағдарламалық өнімді жасау сатысында қандай тәсілдер, стандарттар мен шаблондар қолданылады?

10 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ТЕСТІЛЕУ

10.1. Тестілеудің жалпы сипаттамасы мен оның циклі

Тестілеу бағдарламалық код пен құжаттаманы тексеру бойынша қызметті білдіреді. Ол алдын-ала жоспарлану керек және арнайы тағайындалған тәуелсіз тестілеушімен жүйелі түрде өткізілуі тиіс. Тестілеуші жұмысы талаптарды сипаттауды бекіткенге дейін басталады. Ол БӨ-нің толықтыққа және тестілеу мүмкіндігіне талаптарын тексереді, тестілеу әдістерін анықтайды.

Талаптарды жоспарлау мен оның сипаттамасын жасау сатысының басталуымен бір уақытта тестілеуші тестілеу стратегиясын жасайды. Талаптар сипаттамасы бекітілгеннен кейін ол тестілеу жоспарын жасап, оны тәптіштейді. Сол кезде тестілеуші интеграциялық және жүйелік тестілеуді өткізу үшін мәтіндер жиынтығын жасайды. Тестілеу өткізілу туралы барлық нәтижелер ұсынылатын тестілеу жөнінде есеппен аяқталады.

Әрбір бағдарламалық бұйым үшін оның дұрыстығын тексеретін тесттер жинағы бол керек. Бағдарламалық бұйымды толықтай тексеруге жол беретін тестілеудің бірнеше деңгейі болады.

10.1 сур. Тестілеу циклі

Әрбір деңгей өзіндік мақсат пен құрауышқа ие. Тестілеудің бес деңгейін бөлуге болады: модульдік; интеграциялық; жүйелік; шығыс; қабылдаушы.

Алғашқы төрт деңгейді тестілеу ұйым ішінде өткізіледі, ал қабылдаушы тестілеу тапсырыс беруші өкілдерімен бірігіп жасалады. Бірінші деңгейді тестілеуді жасаушы өзі жасау сатысында өткізеді, ал қалған деңгейлерге тестілеушінің өзі жауапты.

Тестілеу циклі - тестілеудің сәтті аяқталуына дейін интеграциялық, жүйелік немесе қабылдаушы тестілеу үшін тестілеушіге БӨ базалық нұсқасын беру сәтінен бастап тестілеушімен жасалатын әрекеттер жиынтығы (10.1 сур.).Тестілеу циклінің әрбір өткелінде келесілер жасалады:

- тестілеуге жатқызылатын БӨ базалық нұсқасы;
- тестілеу барысы жөнінде есеп;
- тестілеу метрикасы (жобаның мәліметтер базасына енгізіледі).

10.2 Тестілеу түрлері

Модульдік тестілеу. Тестілеудің бұл түрі бағдарлама немесе бағдарламалық жүйе құрамына кіретін жеке бағдарламалық процедура мен ішкі бағдарламаны тексеру процесінен тұрады. Модульдік тестілеу тікелей жасаушымен өткізіледі және әрбір модульдегі барлық ішкі құрылым мен мәліметтер ағынын тексеруге жол береді. Тестілеудің бұл түрі жасау сатысының бөлігі болып табылады. Модульдік тестілеуде тестіленетін әрбір модульдің қамтуы 70...75% аз болмайтындай етіліп өткізілетіндей жасаушымен анықталатын тесттер жиынтығы жасалады. Модульдік тестілеу элементтері: синтаксистік тексеру – бағдарламалық кодта синтаксистік қатені табу үшін кейбір инструменталды құралды қолдану арқылы тексеру; кодтау стандарттарына сәйкестікті тексеру – компанияның кодтау стандарттарына сәйкестігіне кодты тексеру; бағдарламалық кодтың техникалық шолуы.

Модульдік тестілеуді сәтті аяқтағаннан кейін барлық өзгертілген модульдер мен мәтіндер жинағы жобаның мәліметтер базасында сақталады.

Интеграциялық тестілеу. Тестілеудің бұл түрі жеке модульдердің біріккен жұмысын тексеру үшін өткізіледі және бүкіл жүйені бір тұтас жүйе ретінде тестілеуді жасайды. Интеграциялық тестілеу барысында модульдер арасындағы байланыс, олардың сәйкестігі мен функционалдығы тексеріледі. Ол тәуелсіз тестілеушімен жүзеге асырылады және тестілеу сатысы құрамына кіреді.

Интеграциялық тестілеу элементтері:

- функционалдықты тексеру — талаптар сипаттамасында берілген модульдер, функциялар жиынтықтарымен жасалатын жеке функциялардың сәйкестігін тексеру;
- аралық нәтижелерді тексеру – барлық аралық нәтижелер мен файлдардың болуы мен олардың дұрыстығын тексеру;
- интеграцияны тексеру – модульдер бір-біріне ақпаратты дұрыс беруін тексеру.

Интеграциялық тестілеу барысында табылған қателер қателердің мәліметтер базасына енгізіледі. Интеграциялық тестілеу нәтижесі тестілеу циклінің аяқталуында тестілеу барысы жөніндегі есепке қосылады.

Жүйелік тестілеу. Тестілеудің бұл түрі бағдарламалық жүйенің өзін, оның ұйымдастырылуы мен тапсырыс беруші талаптарының сипаттамасына сәйкестігіне жұмыс істеуін тексеру үшін арналған. Оны тәуелсіз тестілеуші интеграциялық тестілеу сәтті аяқталғаннан кейін өткізеді.

Жүйелік тестілеу элементтері:

- шектес тестілеу – шектес жағдайда тестілеу;
- өткізгіш тестілеу – жүйенің шынайы жұмысының барлық функционалдық сипаттамасын тестілеу;
- мақсатты тестілеу – мақсатты платформада тестілеу (мүмкіндік бойынша);
- құжаттаманы тексеру – пайдаланушы құжаттамасының дұрыстығын тексеру;
- тестілеушімен анықталатын басқа тесттер.

Жүйелі тестілеуде табылған қателер жобаның мәліметтер базасына енгізіледі. Жүйелік тестілеу нәтижесі тестілеу барысы жөніндегі есепке қосылады.

Шығыс тестілеу. БӨ-нің тапсырыс берушіге жеткізілуіне әзірлігі тексерілетін тестілеудің соңғы сатысы. Тестілеудің бұл түрін тәуелсіз тестілеуші өткізеді. Шығыс тестілеудің элементтері:

- орнатуды тексеру – орнату бойынша нұқсаулықты анықтық пен дұрыстыққа тексеру;
- құжаттаманы тексеру – барлық қажетті құжаттама әзір және тапсырыс берушіге жіберілуге дайын екендігін тексеру.

Шығыс тестілеуде табылған қателер жобаның мәліметтер базасына енгізіледі. Шығыс тестілеудің сәтті аяқталуында БӨ тапсырыс берушіге тестілеу нәтижесі жөнінде есеппен бірге жеткізіледі.

Қабылдаушы тестілеу. Тестілеудің бұл түрі орнату, бағдарламалық жүйені сүйемелдеу мен түпкілікті пайдаланушыны оқытуға жауап беретін ұйыммен өткізіледі.

10.3. Бағдарламалық қателер

Бағдарламалық қателердің ең кең таралған екі түрі бар:

- бағдарламалық қате – бағдарлама мен оның сипаттамасы арасындағы айырмашылық, бұл ретте сипаттама бар және ол дұрыс болған кезде;
- бағдарламалық қате – бағдарлама пайдаланушы күтетін нәрселерді істемейтін кездегі жағдай. Барлық бағларламалық қателерді сәйкес категорияларға бөлуге болады. Солардың ішінде ең көп кездесетіндерін қарастырайық.

Функционалды кемшіліктер. Аталмыш кемшіліктер бағдарлама керекті нәрсені істемесе, немесе бір функцияны нашар немесе толығымен емес жасаса болады. Бағдарлама функциясы оның сипаттамасында мұқият жазылу керек, және сол бекітілген сипаттамаға негізделіп, тестілеуші өз жұмысын жасайды.

*Пайдаланушы интерфейсінің кемшіліктері.*Пайдаланушы интерфейсінің ыңғайлығы мен жұмысының дұрыстығын тек онымен жұмыс барысында бағалауға болады. Бұл жұмыста пайдаланушының өзі қатысқаны дұрыс. Оған талаптар сипаттамасындағы бекітумен пайдаланушы интерфейсіне барлық талаптарды сәйкестендіру мен жүргізу жасалатын БӨ типтұлғасын жасау арқылы қол жеткізуге болады. Талаптар сипаттамасын бекіткеннен кейін одан ауытқулар немесе орындамаулар қате болып саналады. Ол толық түрде пайдаланушы интерфейсіне қатысты.

Жеткіліксіз өнімділік. Кейбір БӨ жасауда оның өте маңызды сипаттамасы болып жұмыс жылдамдығы саналады, кейде бұл белгі тапсырыс берушінің БӨ-ге талабында көрсетіледі. Пайдаланушыда бағдарлама баяу жұмыс істейді деген ой қалыптасса жаман, әсіресе бәсекелес бағдарламалар жылдам істейтін болса, және одан да жаман, егер бағдарлама талаптар сипаттамасында берілген сипаттамаларға сай келмесе. Бұл - міндетті түрде жойылуы керек қате.

Қателерді жаңылыс өңдеу. Қатені өңдеу процедуралары – бағдарламаның өе маңызды бөлігі. Қатені дұрыс тауып, бағдарлама ол жайында хабар беру керек. Мұндай хабардың жоқтығы бағдарлама жұмысында қате болып саналады.

Шектес жағдайлардың жаңылыс өңделуі. Көптеген әртүрлі шектес жағдайлар бар. «Көбірек» немесе «азырақ», «ертерек» немесе «кешірек», «алғашқы» немесе «соңғы», «қысқарақ» немесе «ұзынырақ» ұғымдары қолданылатын бағдарлама жұмысының кез—келген аспектісі диапазон шектерінде міндетті түрде тексерілу керек. Диапазон ішінде бағдарлама тамаша жұмыс істеуі мүмкін, ал шектеулерде БӨ жұмысының

қатесінеалып келетін күтпеген жағдайлар болуы мүмкін.

Есептеу қателері. Есептеу қателеріне есептеу алгоритмдерін қате таңдау, қате формулалар, өңделетін мәліметтерге қолданылмаймын формулалардан пайда болатын қателер жатады. Есептеу қателерінің ішінде ең кең таралғандар дөңгелектеу қателері.

Ағынды басқару қателері. Бағдарламаның жұмыс істеу логикасы бойынша бірінші әрекеттен кейін екіншісі орындалу керек. Егер оның орнына үшінші немесе төртінші әрекет орындалса, онда ағынды басқаруда қате жіберілген

Жарыс оқиғасы. Жүйеде екі оқиға күтіледі делік: А және Б. А оқиғасы бірінші болса, бағдарлама жасалуын жалғастырады, ал егер Б болса, онда бағдарлама жұмысында жаңылыс шығады. Жасаушылар әрдайым бірінші болып А оқиғасы шығып, Б жарысты жеңіп, ертерек шықпау керек деп күтеді. Жарыстың классикалық оқиғасы осындай.

Жарыс оқиғасын тестілеу айтарлықтай қиын. Олар өзара әрекеттесін процестер мен ағындар параллель жасалатын жүйелер,сонымен қатар шынайы уақыттың пайдаланушысы көп жүйелер үшін типті. Мұндай жүйелердегі қателерді елестету қиын, және оларды табуға әдетте көп уақыт кетеді.

*Шамадан артық жүктеме.*Бағдарлама жұмысында жаңылыс жадтың жетіспеушілігі немесе басқа да қажетті жүйелік ресурстардың болмауы салдарынан болады. Әрбір бағдарламаның өз шегі болады, бағдарлама көтеріңкі жүкті көтере алмайды, мысалы өте көлемді мәліметтерді. Мәселе бағдарламаның шынайы мүмкіндіктері мен оның талаптары бағдарларама сипаттамасы ресурстарына сай келеді ма және ол өзін шамадан артық жүктемеде қалай ұстайтындығында.

Компьютер аппаратурасымен қате жұмыс. Бағдарлама аппаратты құрылғыларға қате мәліметтер жіберуі, олардың қате жөнінде хабарларын елемеуі, бос емес немесе мүлдем жоқ құрылғыларды қолдануға талпынуы мүмкін. Қажетті құрылғы жәй ғана бұзылған болса, бағдарлама оған үндеу талпынысымен «қатып қалмай», оны ұғыну керек.

10.4. Құжаттаманы тестілеу

Құжаттаманы оқып және анализдеп, тестілеуші бірінші кезекте оның дәлдігі, толықтығы, анықтығы, қолданыс жеңілдігі мен оның БӨ-ге қаншалықты сәйкес екендігіне мән береді. Құжаттаманы тестілеу барысында аталған белгілердің әрқайсысы бойынша мәселелер туындауы мүмкін. Сондықтан баспалық басшылық, интерактивті анықтамалық мен басқа да құжаттарды бірнеше рет тестілеуді алдын ала жоспарлаған жөн.

Құжаттамамен жұмыс істейтін тестілеуші оның әрбір сөзінің

техникалық дәлдігі үшін жауап береді. Ол бағдарламаның сипаттама талаптарына сәйкестігі мен бағдарлама тәртібін мұқият тексеру керек. Мәтіннің қиын және шиеленіскен жерлеріне аса назар аударған жөн. Олар бағдарламаның өзінің сәтсіз жобаланған элементтерін көрсетуі мүмкін. Техникалық жазушы өнімді өзі қандай, тура солай етіп жазуға міндетті, сондықтан шиеленіскен жерлерді жоюға жобаны өзгерту ғана көмектесе алады. Мұндай өзгерістің болуын талап ету маңызды, себебі соңында олар өнімді құжаттауды жеңілдетіп қана қоймайды, және оның қолданылуын жеңілдетеді.

Құжаттамада өнімнің қандай да бір функциясы жіберілмегендігін тексеру керек. Техникалық жазушылар сипаттама, жеке жазбалар мен жасаушылармен сұхбаттарға негізделеді. Жасаушылар оларды істің мәнінде ұстауға тырысады, бірақ кейде бағдарламаға енді ғана енгізілген жаңа функциялар жөнінде айтуға ұмытып кетеді. Тестілеушілер техникалық жазушыларға қарағанда мұндай функцияларға ертерек тап болатындықтан, олардың сипаттамасы құжаттамаға түскенін бақылау керек. Сонымен қатар,егер анықталған функция басшылықта жазылса, ол интерактивті анықтамалықта да жазылады деген сөз емес. Ақпарат оңай жоғалып кетуі мүмкін.

Тестілеуші БӨ басшылығына да, БӨ-нің өзіне де өзгерістер енгізуді талап етуге құқығы жоқ. Тестілеуші міндеті –мәселені табу, ал ол мәселемен не істеу керекгігін ол шешпейді. Сондай-ақ, тестілеушінің мәтінде стилистикалық өзгерту енгізуді сұрауға құқығы жоқ. Ол мұндай өзгерту енгізуді ұсына алады, алайда техникалық жазуы бәрі қалай тұр, солай қалдыруды шешуге құқылы және тестілеушіге дұрыс істеп жатқандығы жөнінде дәлелдеуге міндетті емес. Техникалық жазушымен өзара қатынас үшін мәселені қараудың формалды жүйесі әдетте қолданылмайды.Көптеген комментарийлер басшылықтың көшірмесіне енгізіледі.

Техникалық жазушымен келісім бойынша мәтінде өзгертулер мен комментарийлерді көрсету тәсілі таңдалады. Комментарийлер көшірмесін сақтап, және сол бойынша құжаттаманың кезекті нұсқасын тексерген жөн.

10.5. Тесттерді жасау мен орындау

10.5.1. Жақсы тестке талаптар

Жақсы тест келесі талаптарға сай болу керек: *тестпен қателерді табу мүмкіндігі жеткілікті болу керек.* Тестілеу мақсаты - мүмкін болатын қателерді іздеу. Сондықтан, тестілеу мысалдарын жасай отырып, бағдарламаның ауытқуы мен оның

қате жұмысының барлық мүмкін болатын нұсқаларын анализдеу керек;

мәтінді теру көп болмау керек. Егер екі мәтін бір қатені табуға арналса, онда біреуін ғана орындаған жеткілікті;

тест өз категориясында ең жақсы болу керек. Ұқсас тесттер тобында біреуі екіншісінен тиімдірек болуы мүмкін. Сондықтан, тестті таңдауда қатені табуға ықтималдығы жоғарысын алу керек.

тест тым жеңіл немес тым ауыр болмау керек. Үлкен және ауыр тестті түсіну қиын, жасау қиын және оны орындау ұзақ. Сондықтан тестті жеңіл етіп, бірақ бір уақытта тым қарапайым етпей, алтын ортаны ұстаған жөн.

10.5.2. Эквиваленттілік классы мен шектес шарттар

Тестілеу теориясында негізгі ұғымдардың бірі ретінде «эквиваленттілік класстары» мен «шектес шарттар» саналады.

Эквиваленттілік класстары.Эквиваленттілік классы – тесттер жиынтығы олардың орындалуынан бір нәтиже күтіледі. Ең оңай жағдайда, тест тестіленетін бағдарламаға енгізілетін кіріс мәліметтерінің жиынтығынан тұрады. Эквивалентті тесттер жағдайында бұл мәліметтер ортақ қасиеттерге ие.

Тесттер тобы, егер келесі шарттар орындалса, эквиваленттік классты құрайды.

- барлық тесттер бір қатені табуға арналған;
- егер тесттердің бірі қатені тапса, онда қалғандары да соны істейді;
- егер тесттердің бірі қатені таппаса, қалғандары да оны істемейді.

Аталып өткен абстрактілі шарттардан басқа тесттердің нақты тобын бір классқа жатқызуға мүмкіндік беретін практикалық белгілер бар. Мұндай белгілер ретінде келесі шарттарды орындауды қарастыруға болады:

- тесттер өздеріне бір кіріс мәліметтерді қосады;
- тесттерді өткізу үшін бағдарламаның бір операциясы жасалады;
- барлық тесттердің нәтижесінде бір шығыс мәліметтер жасалады;
- немесе ешқандай тест бағдарламаның қатесін өңдеу блогын шақырмайды,немесе бұл блок топтың барлық тестімен шақырылады.

Эквиваленттілік класстарын іздеу – субъектілі процесс. Бір бағдарламаны анализдейтін екі адам класстардың әртүрлі тізімін жасайды. Алайда эквиваленттіліктің неғұрлым көп класстарын табуға талпынса, соғұрлым жақсы болады. Ол келешекте уақытты үнемдеуге септігін тигізеді және

тестілеушіні эквивалентті тесттерді қажетсіз қайталаудан арылтып, тестілеуді тиімдірек етеді. Тесттерді класстарға бөліп, кейін олардың әрқайсысынан тиімді болып көрінетін бір немесе бірнеше тестті бөліп алуға болады; қалғандарын жасаудың қажеті жоқ.

Эквиваленттілік классын іздеу үшін міне қанша ұсыныстар бар:

*Қате немесе ұйғарымсыз кіріс мәліметтеріне ие класстар жөнінде ұмытпаңыз.*Әдетте мұндай кіріс мәліметтері бағдарламады әртүрлі қателер шығарады. Сондықтан, егер сіз неғұрлым көп қате енгізу типтерін тапсаңыз, соғұрлым көп қате табасыз.

*Жасалатын класстар тізімін кесте түрінде ұйымдастырыңыз.*Әдетте эквиаленттілік класстары көп болады, сондықтан жиналған ақпаратты ұйымдастырудың ыңғайлы және ойластырылған тәсілі қажет. Көбінесе барлық ақпаратты үлкен кестеге жинастырады, оның мысалы 10.1 кестеде көрсетілген.

Тізімге тек қана ұйғарынды ғана емес, сонымен қатар ұйғарымсыз немесе стандартты емес кіріс мәліметтері тесттері қосылғанына назар аударыңыз. Кесте түріндегі ақпарат түсінікті, оны қабылдау оңай және тез, ұйғарынды және ұйғарымсыз нұсқаларға бөлу айқын.

10.1 кесте

Эквиваленттілік классының тізімі

Кіріс немесе шығыс оқиғасы	Эквивалент-тіліктің ұйға-рынды класстары	Эквиваленттіліктің ұйғарымсыз класстары
Санды енгізу	1-ден 99-ға дейінгі сандар	0 саны. 99-дан көп сан. Нәтижесі ұйғарымсыз сан болатын көрініс (мысалы,5-5,нәтижесі 0). Теріс сандар Әріптер мен басқа да сандық емес белгілер
Аттың бірінші әрпін енгізу	Бірінші таңба - бас әріп. Бірінші таңба-бас әріп.	Бірінші таңба әріп болып табылады.
Түзуді сызу	Бір нүктеден түзу линиясы 4 см-ға дейін	Суреттің жоқтығы. Линия 4см-ден ұзын. Линия түзу емес

Кіріс мәліметтерінің ұйғарымсыз нұсқаларының барлығы жоғарыда аталған эквиваленттілік класстарымен қамтылғандығын анықтау үшін кестелі түрдегі ақпаратты анализдеу оңай.

Сандық мәннің диапазондарын анықтаңыз. Мәннің әрбір жаңа диапазонымен соған байланысты эквиваленттіліктің бірнеше жаңа класстары пайда болады. Әдетте олардың арасынан үш ұйғарымсыз класс болады: диапазонның төмен шектес мәнінен төмен барлық сандар; сандық емес мәліметтер. Кейде бұл класстардың біреуі болмайды. Мысалы, кез-келген санды енгізуге болады. Бұл жағдайда расында да солай екендігіне көз жеткізу керек. Өте үлкен санды енгізіп, одан не шығатынын көру керек.

Мәндерде диапазон ішілік зерттелетін параметрлердің бар жоқтығын тексеру керек.Әрбір ішкі диапазон эквиваленттіліктің жеке классы болады. Ұйғарымсыз класстар ең төмен диапазоннан төмен және ең жоғарғысынан жоғары орналасады.

Егер өріс немесе параметрлер үшін мәннің бекітілген тізімі болса, осы мәндердің қайсысы тізімге кіретінін анықтаңыз. Егер параметрге мәннің тек анықталған тізімі рұқсат етілсе, эквиваленттіліктің бір классы осы тізімнің барлық мәндерін қоса алады, ал басқасы – қалған мәндерді. Келесіде бұл екі классты кішірек класстар қатарына бөлуге болады.

Тізім мен мәзірден таңдаудың мүмкін болатын нәтижесін анализ жасаңыз. Опция тізімінің ұсынылған бағдарламасының кез-келген элементі эквиваленттіліктің жеке классын көрсете алады. Опция мәзірі немесе тізімінің әрбір элементі бағдарламамен ерекше түрде өңделеді, сондықтан олардың барлығы тексеруге жатады. Ұйғарымсыз мәндер классына тізімде жоқ пайдаланушы жауаптары жатқызылады (егер бағдарлама тек таңдауға емес, сонымен қатар, опция мәнін енгізуге мүмкіндік берсе).

Мысалы, егер бағдарлама «Сіз сенімдісіз бе? (Ц/Н» деген сұрақ қойса, онда эквиваленттіліктің бір классы «Ц» («д»-ны да тексеру керек) деген жауап қайтару керек, ал екіншісі – «Н» («н») жауабы. Қалған жауаптардың барлығы ұйғарымсыз болып табылады (алайда бағдарлама оң болып табылмайтын барлық жауаптар, теріс болып табылатын, яғни «Н» жауабының эквивалиенті ретінде саналатын жауаптарды түсіндіруі мүмкін).

Уақытқа тәуелді мәндердің класстарын іздеңіз. Сіз жүйе бағларламаны жүктегенге дейін, жүктеу барысында, және жүктегеннен кейін бірден аралық пернесін бастыңыз дерлік. Бір қызығы, бірақ мұндай тесттер жүйені бұзуы мүмкін. Бұл жағдайда эквиваленттіліктің қандай класстарын бөліп алуға

болады? Олардың біріншісіне міндетті орындауға біраз уақыт бұрын жасалатын барлық оқиғалар, екіншісіне міндетті орындау алдында қысқа мерзім ішінде жасалатын оқиғалар, үшіншісіне оның жасалу барысындағы оқиғалар және т.б. жатқызылады.

*Нәтижесі нақты жиынтық немесе мәндер диапазонымен шектелетін анықталған есептеулерде бірігіп қатысатын айнымалылар тобын табыңыз.*Үшбұрыштың үш бұрышының шамасын жазыңыз. Үйғарында классқа сомасында 180° беретін мәндер жатқызылады. Үйғарымсыз мәндерді эквиваленттіліктің екі классына бөліге болады - 180° аз сомалық мәнмен және 180° көп.

*Қандай әрекеттерге бағдарлама эквивалентті оқиғалармен жауап беретінін қараңыз.*Жоғарыда тек кіріс оқиғалары қарастырылды, себебі оларды сараптау оңай болды. Шығыс оқиғасының мысалы ретінде 10.1 кестедегі үшінші оқиғаны қарастырамыз. Бағдарлама 4 см-ге дейін линияның сызылуын қамтамасыз етеді, және линия түзу деп болжанады, алайда ол басқа формада болуы да мүмкін.

Қандай кіріс мәліметтер линияның ұзындығы мен формасын басқаратынын анықтау қиындыққа соққызады. Кейде шығыстағы кіріс мәліметтерінің әртүрлі класстары бірдей эффект береді. Егер кіріс мәліметтерінің әрбір классының өңделуінің нақты жолы беймәлім болса, онда оларды әртүрлі класстар деп есептеп, жеке тестілеу керек. Шығыс ақпаратын жасау барысында анықталған кіріс мәліметтеріне жауабына қате түрленсе,онда басқару оның өңделу блогына берілетіндігі әсіресе маңызды.

Операциялық ортаның нұсқаралын ойластырыңыз. Бағдарламаға монитор,принтер, модем, диск құрылғылары немесе жүйеге қосылған кез-келген басқа құрылғылардың типтерінде ғана жақсы жұмыс істейтіндер кездер болады. Бағдарлама жұмысы компьютердің тактілік жиілігіне байланысты болуы да мүмкін. Сондықтан, бағдарламаны сараптай отырып, әсіресе құрылғымен төмен дәрежелі операцияны жасайтын немесе оның анықталған мүмкіндіктеріне бағдарланатын бағдарламарға қатысты жүйенің эквивалентті конфигурация классын анықтау өте маңызды.

Эквиваленттілік класстарының шегі. Эквиваленттіліктің әрбір классы үшін бір-екі тест өткізу жеткілікті. Олардың ең жақсысы класс шегінде жатқан мәндерді тексеретіндер. Бұл мәндер ең үлкен, ең кіші немесе тағыда басқаша болуы мүмкін, бірақ кез-келген жағдайда олар класс параметрінің шекті мәні болу керек. Салыстырудың қате операторлары (мысалы > орнына >) аргументтердің шекті мәндерінде ғана қате жасайды.

Бір уақытта диапазонның аралық мәнінде жаңылысатын бағдарлама оның шекті мәнінде де сөзсіз жаңылысады.

Эквиваленттілік классының әрбір шегін екі жағынан да тестілеу қажетті. Бұл тесттен өтетін бағдарлама аталмыш классқа қатысты қалғандырының бәрінен де өтеді. Міне мысалдар қатары:

1-ден 99-ға дейін мәндер ұйғарынды болса, ұйғарынды мәліметтерді тестілеу үшін 1-ден 99-ға дейін таңдауға болады, ал ұйғарынды еместерді тестілеу үшін - 0 мен 100.

Егер бағдарлама бас әріпті ағылшын әрпін күтсе, A және Z басыңыз. @ таңбасын да тексеріп көріңіз, себебі оның коды A таңбасының кодына сәйкес келеді, және] таңбасының коды Z таңбасының кодынан ереді. Сонымен қатар a және z таңбаларын теңсеріңіз.

Егер бағдарлама ұзындығы 4 см-ге дейін линияның бір нүктесінен сызуды қамтамасыз етсе, бір нүкіте мен ұзындығы тура 4 см линия сызыңыз. Бағдарлама да нөлдік ұзындықтағы линияны сызуды қамтамасыз етуді көрсі.

Кіріс мәнінің сомасы 180-ге тең болса, сомасында 179,180 және 181 беретін мәнді жазып көріңіз.

Бағдарлама кіріс мәліметтерінің анықталған санын алса, анық қажетті санды жазып көріңіз, сонымен қатар бірлікке көп немесе бірлікке аз санды көріңіз.

Бағдарлама B,C және D жауаптарын қабылдаса, A және E жазып көріңіз.

Файлды принтер тағы біреудің тапсырмасын шығарудың алдында және бірден содан кейін басып шығаруға жіберіп көріңіз.

10.5.3. Жағдайлар арасындағы өткелді тестілеу

Әрбір интерактивті бағдарламада бір ақиқат жағдайдан екіншісіне өтулер жүзеге асырылады. Ең оңай мысал ретінде мәзір бола алады. Бағдарламаны қосқаннан кейін онда пәрмендердің бір тізбегі болады. Олардың біреуін таңдағаннан кейін бағдарлама жағдайы өзгереді және мәзірде жаңа жағдайда қол жетімді пәрмендер пайда болады.

Бағдарламамен ұсынылатын әрбір опцияны,мәзір пәрменін тестілеу керек. 10 пәрмені 9 немесе 22 пәрмендері бойынша ашылатын режимде қол жетімді бола алады. Бұл жағдайда 10 пәрменін екі рет – екі режимде тестілеу керек.Алайда мәзір пәрмендері, бағдарламаның барлық мүмкін режимдері мен осы режимдерге өту жолдары өте көп болу мүмкін, және оларды тестілеу мүмкін болмайды. Сондықтан, тесттерді бағдарламаның орындалуы жолын тексеру үшін іріктей отырып,

келесі принциптерді басшылыққа алған жөн:

пайдаланушы әрекеттерінің ықтималды тізбектігін тестілеу;

егер пайдаланушының бір режимдегі әрекеттері мәліметтерді ұсыну немесе басқа режимде бағдарламамен берілетін мүмкіндіктер жиынтығына әсер етуге мүмкіндік берілсе, осы әрекеттерді тестілеу;

жоғарыда көрсетілгендердің ішінен ең қажетті тесттерді өткізуден басқа, бағдарламамен оның орындалу жолдарын кездейсоқ таңдау арқылы еркін режимде жұмыс істеу.

Жағдайлар арасындағы өткелдер мәзір пәрменін таңдауға қарағанда әлдеқайда қиын болуы мүмкін. Мәліметтерді енгізудің құрамы мен кезекті формасының құрылымы өткен формада енгізілген ақпаратқа тәуелді болады, бір өрістің мәні басқасының ұйғарымды мәнін анықтайды, анықталған ақпаратты енгізу қосымша сұраныстар сериясын бастамалайды. Мысалы, 1-ден 99-ға дейін сандарды енгізуде бағдарлама пайдаланушыға үнделген сұраныстың бір формасын шығарады, ал кез-келген басқа сандарды енгізуде – басқасын. Бұл жағдайда эквиваленттілік классы мен оның шектік мәндерімен бірге мәтіндердің толық жиынтығын жасау үшін бағдарламаны жасаудың мүмкін болатын жолдарын сараптау керек.

Мәзір сызбасын жасау өте пайдалы. Мұндай сызбада жағдайлар арасында өткелдерді шақыратын бағдарлама мен пәрменнің барлық жағдайлары көрінеді. Оларға мәзір арқылы белсендірілетін пәрмендер, графикалық құралдар (мысалы, әртүрлі батырмалар) мен анықталған пернелерді басудан кейін жасалатын пәрмендер қосылады. Мысалы, сызбада « Файл» мәзірінен «Ашу» пәрменіне, кейін «Файлды ашу» мен керісінше бағдарламаның негізгі жағдайына оралу жолы көрсетілуі мүмкін. Егер анықталған диалогтық терезені бірнеше тәсілдермен ашуға және одан бірнеше әртүрлі режимдерге шығуға болатын болса, онда бұл сызба өте ыңғайлы. Бұл жағдайда сызбада өткелдердің барлық бағыттарын суреттеп және солар бойынша бағдарламаны тестілеуге болады.Бұл бағдарламаның жағдайының маңызды өзара байланысын өткізіп алу қаупімен жоспарсыз бағдарламамен жұмысқа қарағанда, сенімдірек тәсіл.

10.5.4. Жарыс шарты мен басқа да уақыт тәуелділіктері

Екі жағдай арасында өткел жасап жатқан кезде бағдарлама жұмысына араласып көріңіз. Пернелерді басыңыз, әсіресе пәрмендік. Пернелерді басыңыз немесе бағдарлама мәліметтерді өндеу немесе кіріс-шығыс операциясын жасап жатқанда мәзір пунктілерін таңдаңыз, бағдарламаға параллель түрде тағы бір ақпаратты енгізуді немесе шығаруды ұсыныңыз.Мысалы,

файлды басып шығаруда оны тағы біреуін шығаруды сұраңыз.

Егер бағдарламада ол берілген уақыт ішінде анықталған оқиғаны күткенде, кейіннен ол басқа жағдайда өткенде тайм-аут жағдайы анықталса, оның пайдаланушы әрекетіне реакциясын, жүйе сұранысы немесе тайм-аут интервалы шектерінде күтілетін жағдайдың болуын тексеріңіз.Егер жағдай бағдарлама жағдайды күтуді тоқтатқанға бірнеше секунд қалғанда, немесе бірнеше секундтан кейін болса не болатынын көріңіз.

Жүйені жоғары жүктемеде тестілеңіз.Мультиберілген ортада бірнеше басқа бағдарлама қосып, сіздің бағдарламаңыз өзін қалай ұстайтынын қараңыз, ол өз жұмысын сәтті жасайды ма жоқ па. Процессор әрдайым баспаға қызмет көрсетуге ауысуы үшін принтерге үлкен файл жіберіңіз. Көптеген сыртқы құрылғылар қосып, оларды тоқтату қаншалықты болатындай етіп түрлендіріңіз.Қысқаша сөзбен, қаншалықты мүмкін болғанша компьютерді баялатып, жүктеңіз. Нәтижесінде сіздің бағдарламаңыз баяуырақ жасалып, және мәліметтерді тезірек енгізіп, оның қабылдау мүмкіндіктерін арттыруға болады. Егер жұмыстың қалыпты режимінде бағдарлама жаңылысына қол жеткізу мүмкін емес болса, жоғары жүктемеде оған қол жеткізуге болады.

Аса жоғары жүктемеде бағдарламаны «стандартты» тестілеуді жасай отырып, жарыстың мүлдем күтілмеген жағдайларына тап болуға болады. Егер бұл байланыста бағдарлама осал болса, онда мұндай жағдайда тестілеудің толық циклін өткізу қажетті. Басты мақсат – бағдарламалық қамтамасыз ету жай жұмыс істесе де, бірақ кез-келген жүйеде және кез-келген қосымша жүктеуде ауытқусыз істейтіндей етіп, жасалатын бағдарламалық қамтамасыз етудің осындай сенімділігін қамтамасыз ету. Ең дегенде, бағдарламаны пайдалану үшін жүйенің қандай конфигурациясы шекті болатынын нақты анықтау керек.

10.5.5. Жүктемелік сынақтар

Құжаттамада анықталған БӨ мүмкіншілігінің шектеулерін тестілеуді ұмытпаған маңызды. Бағдарлама жұмыс істей алатын файлардың максималды санын немесе басқа да құрылымдық мәліметтерді ашыңыз, оны сондай жағдайда ұзағырақ пайдаланыңыз. Егер құжаттамада шектеулер жазылмаса, бірақ кейбір параметрлердің логикалық рұқсат етілген мәндері болса, онда оларды да тексеру керек. Егер бағдарлама пайдаланушы енгізе алатын үлкен сандық мәнді жасай алмаса, онда қате жөнінде есеп жасалады. Ал егер бағдарлама параметрлердің өте кішкентай, сонымен қатар, ең үлкен мәндерді қабылдап, өңдесе,

онда оларға шектеу болмауы мүмкін.

Әртүрлі аппараттық ресурстар аяқталса, мысал, диск толып кетсе немесе принтерде қағаз бітсе, онда бағдарлама өзін қалай ұстайтындығын тексеру керек. Жүйеде өте аз орын қалса не болатынын анықтаңыз, компьютерді айтарлықтай жүктеңіз де, не болатынын көріңіз.

Жүктемелік тестілеу – шектес шартты тестілеудің бір түрі болып есептеледі. Оны өткізу сызбасы өте ұқсас. Бірінші бағдарламаны өзі жұмыс істейтін жағдайда қосады, кейін ол бейімделмеген ортада қосады. Әртүрлі ілінген жүктемелерді бөлек-бөлек орындап, бағдарлама олардың бірге болуына шыдамайтындығы әбден мүмкін. Жүйені жүктеп, бір-екі тест жасап қана қоймай, ұзақ және себепті тестілеулер жасаңыз. Осындай жағдайда бағдармаланы біраз уақыт пайдаланыңыз,бірден болмаса да, бірақ ауытқу сонда да болатынын байқайсыз.

10.5.6. Қатені болжамдау

Кейде тестілеуші тесттердің анықталған классы бағдарламаның ауытқуын шақырады дейді, алайда оны логикалық жағынан түсіндіре алмайды. Өз түйсігіңізге сеніңіз және мұндай тесстерді міндетті түрде жалпы жоспарға қосыңыз. Шектес болмаса да, бірақ бағдарламалық ауытқуды шақыратын жағдайлар мен мәндердің бір қатары бар. Мұндай мәндердің типтік мысалы 0 болып табылады. Неліктен анықталған кіріс мәні немесе бағдарламаның орны сізге күмәнді екендігін анықтауға уақыт жоғалтудың қажеті жоқ.Тек қана оны тестілеңіз.

Қиын жағдайда түйсік тривиалды логикаға қарағанда тестілеудің ең жақсы тактикасын айтатын жағдайлар болады. Ассоциативті байланыс орнайтын кездер де болады: сіз мұндай жағдайларда қате тапқансыз, бірақ оны есте сақтамауыңыз мүмкін. Қалай болғанмен, өзіңіздің ішкі сезіміңізге сеніңіз және оны тыңдауды үйреніңіз: тәжірибемен ол одан әрі дамиды және сенімді болады.

10.5.7. Функционалды эквиваленттілікті тестілеу

Функционалды эквиваленттілікті тестілеуде бірдей математикалық функцияның есептеулер нәтижелері әртүрлі бағдарламалармен салыстырылады. «Функционалды эквиваленттілік» терминінің «экиваленттілік классы» терминімен ешқандай ұқсастығы жоқ. Бірдей функцияны есептеудегі екі бағдарлама да бірдей нәтиже берсе, онда оларда

есептеудің эквивалентті тәсілдері қолданылды деген сөз.

Математикалық функцияны есептейтін және нәтижені басып шығаратын бағдарлама тестіленеді дерлік. Ол жай тригонометриялық функция немесе матрицаны теріске шығаратын немесе кейбір мәліметтер жиынтығын көрсететін қисықты жасау үшін коэффициенттерді қайтаратын қиынырақ функция болуы мүмкін. Әдетте мұндай жағдайда сол әрекеттерді жасайтын, бірақ бұл ретте сенімді және уақытпен тексерілген басқа бағдарламаны табуға болады. Екі бағдарламаға кіріс мәліметтерінің бірдей жиынтықтарын өңдеу ұсынылады. Егер нәтижелер ұқсас болса, демек тестіленетін бағдарлама дұрыс жұмыс істейді деген сөз.

Функционалды эквиваленттілікті тестілеуді автоматтандыру. Функционалды эквиваленттілікті тестілеу әдісін қолдануға болатын барлық жерде ол ең жақсы таңдау болады. Егер функция қиын болса, ол уақытты жақсы үнемдеуге және автоматтандырылмаған есептерде пайда болатын қателерді болдырмауға көмектеседі.

Салыстыру процесін де, мүмкіндігінше, автоматтауға болады. Есептер нәтижесін файлдарға шығару және олардың кезекті салыстыруын сәйкес бағдарлама бойынша жасау ең оңай тәсіл болып табылады. Компьютер файлдарды салыстыруды жылдамырақ және ұқыптырақ жасайды. Нәтижелердің ұйғарында қайшылықтарын да, мысалы дөңгелектеу қателігін, анықтауға болады.

Сонымен қатар, тестілеудің шығыс мәліметтерін енгізуден бастап шығыспен салыстырғанға дейінгі бүкіл процесін автоматтау да мүмкін. Егер олай болса, тестілеу процедурасы сол сәтте және сенімді түрде жасалады.

Осындай тесттерді автоматтау аса қиын емес процесс болғанымен, айтарлықтай уақытты талап етеді. Егер бағдарлама файлдың кіріс мәліметтерін санай алса, онда оны дайындау керек. Сонымен қатар, нәтижелердің салыстырылуын жасайтын кішігірім бағдарламашаларды жазуға тура келеді.

Функционалды эквиваленттілікті тестілеу айтарлықтай шығынға әкелуі мүмкін. Нәтижелерді салыстыру үшін пайдаланылатын сенімді эталондық бағдарлама ондай арзан болмауы мүмкін. Сонымен бірге, кейбір бағдарламашалар жазуға тура келеді. Қосымша техника да қажет болуы мүмкін, мысалы екінші компьютер. Мұндай тестілеуді өткізуге қанша ақшалай құрал кететіндігі жөнінде әмбебап белгіні анықтауға болмайды. Алайда арнайы өлшем бар.

Бірінші кезекте, бағдарламаны қолмен тестілеуге қанша күн кететіндігін бағалаңыз. Уақыт есебіне есептеуді жоспарлау, орындау мен тестті өткізуді қосыңыз. Әрбір тестті бірнеше рет өткізу керектігін ұмытпаңыз, себебі сіз қателерді табасыз және

тестілеудің бүкіл процедурасын басынан бастауға тура келеді. Тестілеудің қанша циклі қажет болатындығын санаңыз.

Тестілеудің автоматтандырылған құралы қанша уақытыңызды үнемдейтінін де бағалаңыз. Бүкіл процесті тағы есептеңіз: жоспарлау, бағдарламалау мен тестті ретттеу. Қажетті уақытты барынша шынайы түрде бағалауға тырысыңыз.

Үнемдеуге ұйғарылатын күндер санын өзіңіз екі есе айлығыңызға көбейтіңіз. Айлық сомасы екіге көбейтіледі, себебі компания тестілеу процесін жылдамдату мен оның сенімділігін арттырудан алатын пайда да есепке алынады. Егер сіздің есебіңіз дұрыс болса, онда алынған сома – бұл эталондық бағдарлама көмегімен функционалды тестілеуде үнемдеуге болатынның минимумы.Егер бағдарламаның өзі осы сомадан аз болса, онда кез-келген саналы басшы оның сатып алынуын құптайды.

Сатып алынатын эталондық бағдарлама мен есептер негіздерін түсіндіретін ұсыныс пен презентацияны дайындаңыз. Егер компания оған қосымша ақша салатындай жасауға кететін уақытты қысқартуда қызығушылық туғызбаса, өнімнің сенімділігі мен сапасы арқасында салынған ақшаның орны кепілді түрде толықтырылатынын түсіндіріңіз.

Сезімталдық анализі. Функционалды тестілеуді автоматтау қолмен тестілеуге қарағанда айтарлықтай көп тест өткізуге жол береді. Алайда олар абсолютті толық тестілеу жасау үшін өте мұқият таңдалуы керек, себебі кіріс мәліметтерінің мүмкін болатын мәндер саны әлі де көп болуы мүмкін.Көптеген функцияларда аргументтердің мүмкін болатын мәндердің саны шексіз, сондықтан оларға компьютердің де күші жетпейді. Міндетті түрде шектес мәндерді тексеру керек болады. Ең жақсы тесттерді қалай таңдауға болады? Көбінесе ол үшін сезімталдық анализі қолданылады. Бұл процедура келесілерден тұрады.

Бірінше кезекте, анықтаудың бүкіл саласында орналасқан параметрлер қатары үшін оның мәндерін есептеп, функцияның тәртібі жөнінде жалпы көрініс алады. Кейіннен аргументтердің кішігірім өзгерістері нәтижеленетін мәндердің едәуір шабысын шақыратын анықтау салаларының жерлерін іздейді. Дәл осындай жерлер қателерге ұшырауға жақын болады.

Бағдарламаланатын функция мен оның эталонын тестілеу нәтижесінде алынған мәндер келіспеуі мүмкін. Егер есептеу процесінде операциялар құбылмалы үтірмен орындалса, онда нәтижелерді дөңгелектеуге немесе қиюға тура келеді, демек, кішігірім айырылыстар болады. Әдетте ол қорқынышты емес. Басқа да себептермен пайда болған арттырылған айырылыстарды бекітуге болатындай етіп, дөңгелектеудің ұйғарымды ауытқуларын дұрыс бағалау керек.

Тестіленетін кіріс мәндерінің әрбір диапазонын ішкі диапазондар қатарына (олар 100-ге жуық болуы мүмкін) теңдей бөлу және әрқайсысының ішінде бір мән бойынша тестілеу ұсынылады. Бір нәрсе дұрыс емес кетсе, уақытты босқа жоғалтпас үшін, әрбір мәнді енгізгеннен кейін нәтиже дұрыс болғандығын тексеріңіз.

Функция тәртібінің жалпы көрінісін алып, оны шұғыл әрекеттер затына сараптаңыз. Егер анықтаудың жеке салаларында функция мәні күрт жоғарыласа немесе төмендесе, не ажырау мен секірулер болса, онда оларға аса назар аударған жөн.

Қандай да бір кіріс диапазонында (немесе олардың айырымы эталондық функция мәнімен бірге) функция мәні күрт өсті дерлік. Осы диапазонды 100 бірдей бөлікке бөліп, әрқайсысының ішінде бір мән бойынша тексеріңіз. Егер бәрі дұрыс болса, сіз барлық тестіленетін аргументтер үшін тестіленетін және эталондық функциялар мәні сәйкес келетіндігіне көз жеткізесіз,ал егер олай болмаса, онда сіз қате таптыңыз.

Математикалық функцияны кәсіби тестілеуде ықтималдық теориясынан білім қажет болады. Егер әңгіме бір емес, функцияның бір қатары жөнінде болса, функцияны анықтау саласының критикалық жерлерін іздеудің тиімді және ғылыми дәлелденген технологиялары қажет, яғни, оның мәні күрт өзгеретін немесе эталондықтан айрықшаланатын жерлерде.

Кездейсоқ енгізу. Бірдей жердің анықталған санына функцияны анықтаудың барлық тестіленетін саласын бөлудің орнына кіріс мәнін таңдаудың басқа жолын – кездейсоқ – таңдауға болады. Мәннің кездейсоқ таңдауы тиімдірек,себебі ол олардың толық тең құқылы болуын кепілдейді. Мысалы, 0,02;0,04;0,06 және т.б кіріс мәндерінің осындай мәндерін тестілей отырып, сіз бағдарлама тақ сандарды (мысалы, 0,03) немесе мәнді цифрлардың көп санымен сандарды (мысалы, 0,1415) қалай өңдейтінін ешқашан білмейсіз. Сол уақытта кездейсоқ образда кіріс мәндерін таңдауда функцияны анықтау саласы толық толтырылады, кіріс мәліметі мәнінің барлық типтері мен диапазондары бірдей алынады.

Егер сіз кіріс мәліметтерін таңдау әдісін таңдауда қиналсаңыз немесе тестіленетін функция тәртібіне сенімсіз болсаңыз, кездейсоқ әдіске тоқталыңыз.Ол автоматты тестілеу үшін де сай келеді.

Нақты кіріс мәнін таңдау үшін айқын түсіндірудің болмауын өткізілетін тесттер санымен толтыруға болады. Мұндай ешқандай шектеу жоқ – эквиваленттіліктің әрбір классы үшін қаншалықты көп тест өткізілсе, соншалықты жақсы болады. Әдетте кездейсоқ кіріс мәндерімен автоматтандырылған тестілеуде кем дегенде 1000 есептеу жасалады.

Кездейсоқ сандар генераторын қолдану. Кездейсоқ енгізу « ойға келгеннің бәрі» дегенді білдірмейді, әйтпесе ол функцияны анықтау саласында бірдей қамтуға үміттену үшін тым біржақты болады. Кездейсоқ сандарды шексіз санда түрлендіре алатын компьютерлік бағдарлама сай келеді. Алайда көптеген ұқсас бағдарламалармен қолданылатын алгоритмдер мүлдем кездейсоқ еместігін есте сақтаған жөн. Сонымен қатар, бағдарламаларда тіпті базалық алгоритм нақты жасалмаған. Сондықтан, тіпті беделді компаниямен жасалған немесе бағдарламалаудың стандартты тілдерінің біріне салынған кездейсоқ сандар генераторын таңдамас бұрын, оның жұмыс негізіне қандай алгоритм қойылғанын және ол сіздің қажеттіліктеріңізге сай келетіндігін анықтау керек. Экранда түрлі түсті салют алуға жол беретін оңай бағдарламаға сай болатын нәрсе қиын инженерлік бағдарламаны кәсіби тестілеуге сай келмеуі мүмкін. Кездейсоқ сандар генераторы тіпті өте үлкен болса да, анықталған интервал арқылы олардың тізбектігін қайталайды. Осылардың барлығына байланысты келесі кеңестерге құлақ салыңыз.

Кез-келген бағдарламаға ол бар болғандықтан ғана сенудің қажеті жоқ, басқасын да іздеп көру керек. Бағдарламаны ұғынуға тырысып көріңіз, және ол сізге толықтай түсінікті болғаннан кейін ғана, оны қолданыңыз.

Егер сіз бағдарламау тіліне енгізілген кездейсоқ сандар генераторын қолданайын деп жатсаңыз, оны тағы кішкене жасау керек. Оның көмегімен сандардың көп бөлігін түрлендіріп (100...1000), оларды араластырыңыз: сол генератормен берілетін кезекті сандар көмегімен олардың тәртібін ауыстырыңыз. Ол жұмысты баяулатқанмен, нашар бастапқы генератор жағдайында да нәтиже айтарлықтай жарамды болуы мүмкін.

Эквиваленттілік технологиясын қолдану. Математикалық функцияны тестілеу – эталондық бағдарламаны қолданудың жалғыз емес саласы. Дайын және тексерілген БӨ-мен салыстыру жолымен бағдарлама тәртібінің әртүрлі аспектілерін тестілеуге болады. Олардың кейбір мысалдары төменде сипатталады.

Бұрыннан бар бағдарламалардың біріндегі алгоритмнің негізінде жатқан емлені тексеру бағдарламасы тексерілсе, онда екі бағдарламаға бірдей сөздер жинағын тексеруді ұсынуға болады.

Сөздерді автоматты ауыстыру бағдарламасы тестіленсе, әсіресе егер басқа тіл үшін оның алгоритмінің модификациясы жасалса, онда бағдарламалық қамтамасыз ету нарығында сатылатын тексеріген бағдарламаны тексеріс үшін алуға болады, мәтіннің тар бағанасын жасап,оны екі бағдарламаға ұсыну керек.

Тармақ жалпақтығы бойынша мәтінді туралауды жасайтын бағдарламаны тестілеуде ол сөздерді аралықтармен қаншалықты

бірдей бөлетіндігін тексеру керек. Үлгі ретінде күнделікті мәтіндік процессорды алып, екі бағдарламаға бірдей шрифтпен терілген бірдей мәтінді ұсынуға болады.

Принтерге жіберілетін екілік басқаратын тізбектілікті реттеу үшін қорытындыны файлға бағыттауға болады және эталлондық бағдарламада тура сондай файл жасап, дәл соны жасауға болады. Кейін екі файлды салыстыру керек – олар бірдей болулары керек.

Файлға оңай жіберілетін шығыс мәліметтерін тестілеу қажеттілігі туындаған барлық жағдайда сол мәліметтерді түрлендіре алатын эталондық бағдарламаны қолдануға болады. Оларды оңай салыстыруға болады.

Әрбір нақты жағдайда осы технологияның «жақтаушы» және «қарсы» өзіндік аргументтері болады. Мысалы, оны іске асыруға тым көп уақыт, құрал немесе күш қажет болады. Сонымен қатар, эталондық бағдарламада да қателер болуы мүмкін. Бірақ қандай болғанымен, эквиваленттілікті тестілеу әдіснамасында назар аударған жөн – оны қолдану жұмысты жылдамдатады және оның тиімділігін бірнеше есеге арттырады.

Қателер жөнінде есепке екі бағдарламадан алынған шығыс мәліметтерін қосуды ұмытпаңыз. Олар қате себебін іздеу үшін өте маңызды.

10.5.8. Регрессиялық тестілеу

Жалпы мәліметтер. Тестілеушінің негізгі жұмысы регрессиялық тестілеу болып табылады. Бұл терминнің екі мәні бар, оларды жасалған тесттерді қайталамалы қолдану идеясы біріктіреді.

Тестті өткізгеннен кейін қате табылып, бағдарламашы оны жөндеді деп елестетіңіз. Тестілеуші ол қате толықтай жойылғандығына көз жеткізу үшін ол тестті қайтадан жасайды. Осы - регрессиялық тестілеу.Бағдарламаның жөнделген фрагментін жақсылап тексеру үшін бастапқы тесттің бірнеше вариациясын өткзіге болады. Бұл жағдайда регрессиялық тестілеу міндеті – табылған қате бағдарламашымен толықтай жойылды және ендігі шықпайтындығына көз жеткізуде.

Регрессиялық тестілеуді қолданудың екінші мысалы. Қате табылып, жойылғаннан кейін тесттердің стандарттық сериясы жасалады, бірақ басқа мақсатпен: бағдарламаның бір бөлігін жөндеп, бағдарламашы басқасын құртпағандығында. Бұл жағдайда бағдарламаның жөнделген фрагменті емес, толықтай бағдарламаның өзі тестіленеді.

Қатені жою бойынша ұсыныстар. Қате жөнінде есепті алып, бағдарламашы бағдарламаның бастапқы кодын мұқият сараптап, қате себебін тауып, оны жойып, нәтижені тестілеу

керек.Бұл идеалды нұсқа. Тәжірибеде кейбір бағдарламашылар қатенің өзін емес, есепте көрсетілген қатенің болу белгілерін ғана жояды. Нәтижесінде қатенің кейбір көріністері жойылады, бірақ қалғандары қалады. Кейде бағдарламашы есепті дұрыс түсінбей, қажетсіз нәрсені жоятын кездер болады. Кейбір жосықсыз қызметшілер өз жұмыстарын мүлде тексермейді, қарамай-ақ, түзетулер енгізіп, бағдарламаны тез арада тестілеушіге қайтарады. Нәтижесінде ескі қате қалып, жаңалар пайда болады. Тәжірибелі тестілеуші мұндай жағдайларға дайын болу керек. Бағдарламаға енгізілетін үштен бір түзетулер істемейді немесе істеп тұрған нәрсені бұзады деп саналады.

Бағдарламашымен енгізілген түзетулерді тексеретін тестілеуші алдына қоюға тиісті үш міндет:

қате жойылғандығына көз жеткізу, ол үшін қате табылған және есепте көрсетілген тестті жасау. Егер бағдарлама өтпесе, онда әрі қарайғы тестілеудің мәні жоқ;

байланысты қатені табуға тырысу. Бағдарламашы есепте көрсетілген қате белгілерін жойды, бірақ қатенің өзін жоймады дерлік. Қате белгілерін басқа жолмен арандатуға болады. Бағдарламада басқа ұқсас жағдайлар болуы мүмкін ба? Тесттер санын көбейту керек;

бағдарламаның қалған бөлігін тестілеу. Түзетудің күтілмеген нәтижелері тағы бір жерде шығуы мүмкін. Оларды іздеу формалды емес түрде өткізіледі – алдын ала дайындалған жоспарсыз. Бағдарламаның қандай бөлігіне енгізілген түзетулер әсер еткендігі жөнінде ойланып, оларды тексеру керек.

Тесттердің стандартты сериясы. БӨ-ні тестілеуді бастаудан біраз уақыттан кейін *регрессиялық тесттер кітапханасы* жасалады.Ол бағдарламашылар кезекті жұмыс нұсқасын тапсырғанда әрдайым жасалатын және бүкіл бағдарламаны қамтитын тесттердің толық жинағы. Бұл жағдайда тестілеуге біршама уақыт жоғалтуға тура келеді, бірақ бүкіл процедура еңбек көлемді болмайды.

Тестілеуге бағдарламаның кезекті нұсқасы ұсынылғанда және кітапхананың бүкіл тестін қайталау уақыты болғанда біршама сұрақтар қатары туындайды. Кітапхана қаншалықты кең? Кітапхананың барлық тестілерін қайтадан жасау керектігі рас па? Регрессиялық кітапханаға қандай тесттерді қосу керектігін әрдайым алдын ала анықтау оңай емес.Сондықтан алғашында онда шынымен қажетті тестке қарағанда, көбірек тест болуы мүмкін. Кем дегенде, оған шектес шарт пен уақыт сипаттамаларын тексеру үшін мысалдар кіру керек. Бірақ олардың барлығын әрдайым жасау керек па?

Регрессиялық тестті әдетте жасауға құлқы болмайды, себебі ол тестпен қатені табу ықтималдығы жоғары емес. Тестілеуді алғашқы жасауда тесттердің кейбіреуі қате табуы мүмкін, бірақ олардың толықтай жойылуынан кейін сол тесттерді қайта-қайта

өткізу уақытты бос жоғалту болады. Егер қате ендігі болмаса, ол қайтадан пайда болатындығының ықтималдығы қанша?Бірнеше рет өткізілген және бірде бір рет қате таппаған тесттерді не істейміз? Кейбір компанияларда мұндай тесттерді кітапханадан алып тастайды, тек қате тапқандарды ғана алып қалады.

Әрбір тестті ойлағанша, оңайырақ жолмен жүрген жақсы. Регрессиялық кітапханаға сізге пайдалы болып көрінген барлық тессті қосыңыз.Әрбір үш цикл сайын, сол кітапхананы қарап, олардың болмауы жұмыс сапасын төмендетпейтін тесттерді жойыңыз. Міне бірнеше пайдалы кеңестер:

Кітапхананың басқа тесттеріне эквивалентті тесттерді жойыңыз. Негізі мұндай тесттер кітапханаға мүлде түспеу керек, бірақ ол бірнеше қызметкерлермен жасалса, мұндай жағдайлар болуы мүмкін.

Объектісі жойылған қате болып табылатын тесттер санын азайтыңыз.Егер қате немесе оның кейбір түрлері тестілеу циклінің қатарында пайда болса, онда кітапханаға оларды табу үшін тесттердің жеткілікті санын қосу керек. Ол айтарлықтай қалыпты және орынды құбылыс. Бағдарламаның сәйкес бөлігін қатенің ізі қалмайынша мұқият түрде тестілеу керек. Алайда одан кейін жойылған қатені іздеуге бағытталған тесттердің көп бөлігін кітапханадан жоюға болады.

Тесттерді қиыстырыңыз. Бағдарлама өтетін тестті біреуге біріктіруге болса – соны істеңіз. Тестілеу басында олай істеудің қажеті жоқ, бірақ кейіннен тесттерді біріктіру жұмысты айтарлықтай жылдамдатады.

*Мүмкіндік бойынша тестілеуді автоматтаңыз.*Егер сіз тесттердің анықталған тобы тестілеудің бес немесе он кезекті циклінде жасалатындығына сенімді болсаңыз, онда автоматтауға уақыт бөлген жөн.

Тесттер бөлігін кезекті орындауға бөліңіз. Регрессиялық кітапхананың барлық тесттерін бағдарламаның әрбір өзгерісінен кейін жасаудың қажеті жоқ. Оны сирегірек жасауға болады – әрбір екінші немесе үшінші циклда. Тестілеудің соңғы сатысында бағдарлама қосылуға дайындығына көз жеткізу үшін тесттердің максималды мүмкін болатын санын орындаған жөн, басқа циклдерде бүкіл тесттің жартысы немесе тіпті үштен бір бөлігі жеткілікті.

Регрессиялық кітапхана жасалған тесттердің ішінен ең жақсысын қосу керек, егер ол өте көп болса, жаңа тесттерді жасауға уақыт қалмайды. Бірақ жаңа тесттер жоғары ықтималдықпен әлі табылмаған қателерді табады. Сондықтан жұмысты регрессиялық кітапхана тестілеудің тиімділігін арттыру құралы ретінде, тежеуіші ретінде емес, қызмет ететіндей жоспарлау керек.

Тесттерді орындау.Жақсы тест ойлап табу – ол істің жартысы ғана. Оны тағы дұрыс орындау керек. Міне бірнеше ұсыныстар:

Егер сіз бағдарламаны компьютерге орналастыруда пайдаланушыда конфигурацияны таңдау мүмкіндігі болғанын қаласаңыз, орнату бағдарламасын қосу мен қажетті опцияларды ұсынатындығын қарау ғана жеткіліксіз. Бағдарламаның өзін әрдайым қосып, таңдалған конфигурация расында да орнатылғандығын тексере отырып, орнатудың әрбір нұсқасын жасаңыз.Бағдарлама бұл ретте толықтай жұмысқа қабілетті екендігіне көз жеткізіңіз.

Егер бағдарлама басылатын бет, бос жер мен басқа да ұқсас ақпаратты көрсетуге мүмкіндік берсе, құжат экранда дұрыс бейнеліп тұрғанын көріп, ол іс жасалды деп санамаңыз.Оны басып шығару қажет.

Егер бағдарламада ASCII кеңейтілген жиынтығының таңбалары қолданылса, оларды экранда көру жеткіліксіз. Олар дұрыс басылып шығып, модем арқылы жіберілу керек және т.б. Шығыс мәліметтері жіберілетін барлық бағдарламалық қамтамасыз етуді есепке алған жөн: құрылғы драйверлері, импорт алгоритмі.

Келтірілген ұсыныстардан шығатын негізгі ережені келесідей құрастыруға болады: мәтіндік процедура бағдарламаны енгізілген мәліметті қолдануға және олар дұрыс қолданылатындығын растауға мәжбүрлеу керек.

10.6. Жиналатын метрикалар, қолданылатын тәсілдер, стандарттар мен шаблондар

Тестілеу фазасында жоспарлы мерзім мен көлемдердің шынайы көрсеткіштермен айырмашылығы, өткізілген шолулардың саны, табылған қателер мен дефекттер, сонымен қатар орташа еңбек шығыны мен тестілеу өнімділігін бағалауды жасау керек. Сонымен қатар, ашық және жабық дефекттердің қатынасы мен тестілеу арқылы БӨ өтеуі метрикасын жинау өндірілу керек. Барлық алынған мәліметтерді бағдарламалық топтың ИМБ сақтау керек.

Қолданылатын құрал: құжаттарды даярлау жүйесі (мысалы, MS Word).

Қолданылатын тәсілдер мен стандарттар: ұйымдастыру процесі; метрикалық бағдарлама.

Қолданылатын шаблондар: тестілеу нәтижесі жөнінде есептер; тестілеу барысы жөнінде есептер; БӨ жеткізу туралы есеп; жеткізілетін БӨ бойынша басшылық; тестілеу жоспары; тестілеу процедурасын сипаттау; шолу бойынша есептер; жоба жағдайы жөнінде есептер.

Бақылау сұрақтары

1. Бағдарламалық өнімді жасаудың өміршеңдік циклінде тестілеу сатысының мәні қандай?

2. Бағдарламалық өнімді жасауда өміршеңдік цикл бойында не тестіленеді?

3. Тестілеудің қандай деңгейлері бар және олардың әрқайсысында тестілеуді өткізуге кім жауапты?

4. Тестілеу циклі дегеніміз не, ол өзіне қандай негізгі әрекеттерді қосады?

5. Тестілеудің мақсаты неде және оның негізгі элементтері қандай: а) модульдік; б) интеграциялық;в) жүйелік; г)шығыс?

6. Бағдарламалық қате дегеніміз не?

7. Бағдарламалық қатенің қандай санатын білесіз?

8. Құжаттаманы тестілеудің мақсаты неде?

9. Жақсы тест қандай сипаттамаға ие болу керек?

10. «Эквиваленттілік классы» ұғымына анықтама беріңіз.

11. Тесттерді эквиваленттіліктің бір классына жатқызу үшін қандай шарттар жасалу керек?

12. Эквиваленттілік классын анықтау үшін қандай санаттар қолданылады?

13. «Эквиваленттілік классының шектері» ұғымына анықтама беріңіз.

14. Бағдарламалық өнім жағдайлары арасында өткелдерді тестілеу дегенді қалай түсінесіз?

15. Бағдарламаны орындау жолдарын тексеру бойынша тесттерді таңдау үшін қандай белгілер болады?

16. Жарыс жағдайы мен басқа да уақыт тәуелділіктерін анықтау үшін бағдарламалық өнімге қандай жағдай жасауға болады?

17. Бағдарламалық өнімді жүктемелі тестілеу не үшін өткізіледі?

18. Қатені болжамдау жөнінде не ойлайсыз?

19. «Функционалды эквиваленттілік» ұғымына анықтама беріңіз.

20. Тестілеуді автоматтау қандай артықшылықтар береді?

21. Сезімталдық анализі қалай жасалады?

22. Кіріс мәндерін кездейсоқ енгізудің артықшылықтары неде?

23. Кездейсоқ енгізуді қалай ұйымдастыруға болады?

24. Регрессиялық тестілеудің мақсаты мен мәні неде?

25. Қате сәтті жойылғандығын қалай анықтауға болады?

26. Тестілеу нәтижелері қайда енгізіледі және қатенің жойылуын қалай бақылау керек?

27. Тестілеу сатысында қандай метрикалар жиналады?

28. Тестілеу сатысында қандай тәсілдер, стандарттар мен шаблондар қолданылады?

11 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ СҮЙЕМЕЛДЕУ

11.1. Бағдарламалық өнімнің өміршеңдік циклінде сүйемелдеу сатысының рөлі

БӨ сүйемелдеу – жеткізілетін БӨ-нің жаңа жағдайға бейімделу процессі, оның негізгі функциясын өзгеріссіз етіп сақтауда пайда болған мәселелер немесе модифицкациядағы қажеттіліктерден туындаған БӨ мен сәйкес құжаттамаға өзгеріс енгізу. БӨ сүйемелдеу сүйемелдеуші ұйыммен немесе БӨ жасаған ұйымның сүйемелдеу қызметімен жасалады.

Сүйемелдеумен IEEE-90 стандартына сәйкес табылған қателерді жою, өнімділікті арттыру немесе жұмыс немесе талаптар шартының өзгеруіне бейімделуді арттыру мақсатымен БӨ-ге өзгеріс енгізу болып табылады.

Сүйемелдеу элементтері:

- жеткізілетін БӨ-нің біршама бөліктерін қайта жобалау мен қайта жасау;
- интерфейсті бағдарламалық модульдерді қайта жобалау мен қайта жасау
- БӨ кодының модификациясы, мәліметтер базасының құжаттамасы немесе құрылымы

БӨ сүйемелеу міндетіне БӨ функционалды мақсатын өзгертуге алып келетін жаңару, мен БӨ функционалды мақсатына тиіспейтін өзгерту кіреді. БӨ жаңаруын БӨ өзгеруіне тапсырыс беру жолымен жүзеге асырады. БӨ өзгеруі түзелетін сүйемелдеу (бағдарламада қатені өңдеу, ауыздықтау немесе жою), адаптациялық сүйемелдеу (жаңа ортаға бейімделу) мен жетілдірілетін сүйемелдеулерден (сипаттама мен эксплуатациялық сенімділігін жақсарту) тұрады.

Сүйемелдеу процесі өзіне келесі негізгі әрекеттерді қосады.

1. Келесілерді қарастыратын, дайындық жұмысы:

- сүйемелдеу барысында жасалатын әрекет пен жұмысты жоспарлау;
- сүйемелдеу процесінде пайда болатын ауыздықтау процедурасы мен мәселені шешуді анықтау;

2. Мәселелер анализі мен БӨ модификациясына сұраныстар:

- пайда болған мәселе немесе БӨ модификациясына сұраныстар жөнінде хабар анализі, оның барысында

модификацияны орындау мүмкіндігі, оның типі (түзейтін, жақсартатын, профилактикалық немесе жаңа ортаға бейімделуші); масштабы (модификация өлшемі, құны мен оның іске асу уақыты); сыншылдығы (сенімдік, өнімділік пен қауіпсіздікке әсер ету) зерттеледі;

- жұмысты жасау мен оны өткізу нұсқаларының мақсаттылығын бағалау;
- модификацияның таңдалған нұсқасын бекіту.

3. БӨ құрауыштарын, олардың нұсқалары мен модификацияланатын құжаттамаларын анықтау мен қажетті өзгерістерді енгізуді қарастыратын *БӨ модификациясы*.

4. Модификацияланатын БӨ тұтастығы тексерілетін және енгізілген өзгерістер бекітілетін *тексеріс пен қабылдау*.

5. БӨ-ні жаңа жұмыс ортасына ауыстыру (конвертациялау).

6. БӨ-ні қолданыстан алу

Сүйемелдеу сатысында БӨ-нің жаңа нұсқасын шолу бойынша келесі есептер жасалады: сүйемелдеу метрикасы бойынша; БӨ өміршеңдік циклінің аяқталуы (БӨ өлімі) жөнінде.

11.2. Жиналатын метрикалар, қолданылатын құралдар мен шаблондар

Сүйемелдеу сатысында БӨ жеткізілуінен кейін табылған кінәрәттар санын анықтау, сонымен қатар өзгерістерді енгізудің еңбек қамтуын бағалалау керек. Барлық алынған мәліметтерді жобалық топтың ИМБ сақтаған жөн.

Аталмыш сатыда БӨ-нің өміршеңдік циклінің өткен сатыларының сипаттамаларында айтылған барлық аспаптық құралдар (6.6, 7.6, 8.5, 9.5 мен 10.6 бөлімшелерді қараңыз) және өзгеруге сұраныс шаблоны қолданылады.

Бақылау сұрақтары

1. Бағдарламалық өнімнің өміршеңдік циклінде сүйемелдеу сатысының мақсаты қандай?
2. Сүйемелдеу сатысы өзіне қандай элементтер қосады?
3. Сүйемелдеу сатысында қандай негізгі міндеттер орындалады?
4. Бағдарламалық өнімді өзгертудің қандай типтерін сіз білесіз?
5. Сүйемелдеу сатысында қандай негізгі әрекеттер жасалады?
6. Сүйемелдеу сатысында қандай құжаттар жасалады?
7. Сүйемелдеу сатысында қандай метрикалар жиналады?

12 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМДІ ЖЕТКІЗУДІ БАСҚАРУ

12.1. Жеткізуді басқару жөнінде жалпы мәліметтер

БӨ жасаумен айналысатын әрбір ұйымда БӨ жеткізуді басқару үшін келесілер анықталу керек:

- БӨ жеткізу процедурасы;
- жеткізілетін БӨ классификациясының сызбасы;
- БӨ жеткізу бойынша басшылыққа БӨ сәйкестігін тексеру мен байқау тәсілдері.

БӨ жеткізу процедурасы БӨ жасаумен айналысатын әрбір ұйым ішінде жасалып, бекітіледі, бірақ тапсырыс беруші талаптарына сай ол нақты бір жоба үшін модификациялануы мүмкін. Бағдарламалық өнімді жеткізу процедурасы олардың тізбектілігі көрсетілген жеткізу барысында жасалатын негізгі әрекеттер мен міндеттер тізімін қосады.

Жеткізілетін БӨ классификацисының сызбасы қажетті жеткізу мен олардың құрамының тізімінен, сонымен қатар, БӨ нұсқасының нумерациясының қолданылатын принциптерінің сипаттамасынан тұрады.

БӨ-ні тексеру мен жеткізу бойынша басшылыққа сәйкестігін бақылау әдісі жеткізу процедурасын орындау, жеткізілетін БӨ классификациясы сызбасын сақтау мен БӨ жеткізу бойынша басшылықты жеткізуге сәйкес болуға бағытталады.

12.2. Жеткізілетін бағдарламалық өнімдердің классификациясы

Жеткізуге дайын БӨ келесі типтердің бірі ретінде классификациялану керек:

ES-жеткізу — прототипті жеткізу;

PA-жеткізу — Альфа-нұсқаны жеткізу;

PB-жеткізу — Бета-нұсқаны жеткізу;

RP-жеткізу — соңғы жеткізу.

Прототип жүйеде бекітілген концепцияны көрсету,талаптар нұсқаларын тексеру, сонымен қатар жасау барысында, және БӨ қолданысы барысында пайда болатын мәселелерді іздеу, мен оларды шешудің мүмкін болатын нұсқаларын іздеу үшін

қолданылатын БӨ-нің бастапқы нұсқасы болып табылады.

Пайдаланушылар мүмкіндігінше ертерек тәжірибе жасау үшін БӨ прототипін жылдам жасау өте маңызды.

Альфа-нұсқа прототипіне қарағанда БӨ барлық немесе барлығы дерлік БӨ-нің барлық негізгі функцияларын іске асырады, бірақ олардың кейбіреулері әлі болмауы немесе қатемен жасалуы, тұрақсыз болуы мүмкін. БӨ даралығы ендігі толықтай жасалды, оның негізгі ерекшеліктері мен мүмкіндіктері көрінген.

Спецификация, конструкторлық құжаттама, анықтамалық жүйемен пайдаланушы құжаттамасы дайын деуге болады. Альфа-нұсқаны әдетте тек тапсырыс берушілерге береді.

Бета-нұсқада БӨ-нің жоспарланған функциясының толық жиынтығы іске асырылады. БӨ-де фатальды қате жоқ, айтарлықтай қателер де өте аз. Барлық жобалық құжаттар дайын және БӨ тапсырыс беруші талаптарына сай. Анықтамалық жүйе мен пайдаланушы құжаттамасы толықтай дайын. Өнімді үшінші пайдаланушыларға тестілеуге беруге немесе жарнамалық мақсатта таратуға болады.

Соңғы жеткізу БӨ жасау бойынша жоба жұмысының аяқталуы мен сүйемелдеу сатысына өтуді білдіреді. Өнім өзімен бірге жүретін құжаттамасымен бірге тапсырыс берушіге беріледі.

12.3. Бағдарламалық өнімді жеткізуде жасалатын әрекеттер

Жеткізуге дайын өнім жоғары басшылық пен процесс тобының қатысуымен жасалатын шолуда ұсынылу керек.

Мұндай шолуды жасау үшін келесілер қажет:

жеткізілетін тауарға идентификатор беру;

базалық нұсқа жасау;

жеткізілетін өнім бойынша басшылық дайындау.

Жеткізілетін өнімді шолу үшін ұйымдастыру процесінде анықталған процедураға сәйкес шолуды дайындау мен өткізуге жауапты басшы тағайындалады.

Шолу барысында жеткізілетін өнім толықтығы мен жеткізу процедурасының дұрыс жасалуы тексеріледі. Барлық шығарылатын өнім БӨ жеткізу бойынша басшылыққа сәйкес классификациялану керек.

Бақылау сұрақтары

1. Жеткізу процедурасы өзіне не қосады?
2. Жетізілетін бағдарламалық өнім классификациясының сызбасы қандай мақсатпен қолданылады?
3. Жеткізудің қандай типтерін білесіз?Олардың әрқайсысын сипаттаңыз.
4. Жеткізілетін өнімді шолуға дайындауда қандай әрекеттер жасау керек?
5. Шолуды өткізуге және соған дайындалатын өнімге кім жауапты?

13 БӨЛІМ

БАҒДАРЛАМАЛЫҚ ӨНІМ СЕНІМДІЛІГІН ҚАМТАМАСЫЗ ЕТУ

13.1. Қолданылатын терминдер

Сенімділікті анықтауда БӨ келесі қабылданған терминдермен пайдаланады.

Сенімділік – қате жасау сәтінде зақымданудан құтылуға жол беретін жағдай. БӨ қателері дефекттер, немесе жоба қателері, кодтау, ұйымдастырушылық қателер, ақылға қонымсыз ретке келтірулермен тестілеу қателері («қате» және «дефект» ұғымдарына анықтама 5.4.3 бөлімшеде берілген) салдарынан болады.

БӨ-нің бұзылуға тұрақтылығы – бағдарламаны жасауды жалғастыру мүмкіндігін сақтауда жеке қатені түзеу мүмкіндігінде болатын БӨ қасиеті.

Мәселе – берілген техникалық сипаттама немесе күтілетін нәтижелерден ауытқу.

Өңдеудегі қате – өңдеу процесін жасауда дұрыс емес нәтижелерді шығару.

Процесс – өзара байланысты әрекеттердің шекті қатары, оны жүзеге асыру барысында бастапқы өнімнің бір немесе бірнеше типтері қолданылады,кейіннен бір немесе бірнеше өзгерістер көмегімен тапсырыс беруші үшін құндылықты ұсынатын соңғы өнім жасалады.

Процесті жасау барысында бас тарту – бастапқы өнімдегі қате арқылы процесте қолданылатын оқиға соңғы нәтижеде анық болатын шығыста қатені шығарады.

Процесті жасаудағы ауытқу - процесте қате кіріс мәліметпен қолданылатын және процесте немесе процесс қатысты болатын жүйеде қате жағдайды шақыратын ауытқу.

Тұрақтылық – 8.4 бөлімшеден анықтаманы қараңыз.

13.2. Бағдарламалық өнім сенімділігі мен оны қамтамасыз ету жөнінде негізгі ұғымдар

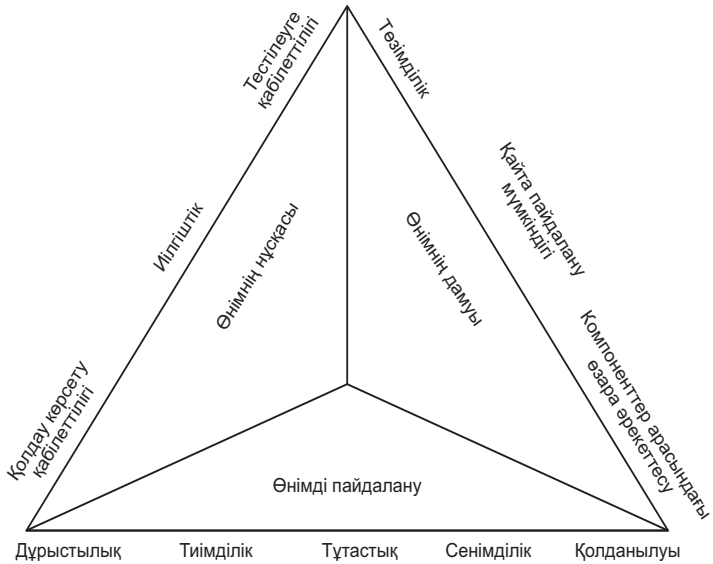
Сенімділік БӨ сапасының негізгі көрсеткіші болып табылады. БӨ сенімділігін қамтамасыз етуде бұл көрсеткіш соңғы пайдаланушы үшін аса маңызды екендігіне назар аударылады, ал БӨ өзгеруі мен оның жаңа нұсқаларын жасауға қатысты сапа факторлары БӨ жасаушылары мен техникалық қолдау

тобы үшін анықталған мәнге ие. Толықтай жұмысқа жарамсыз немесе қате жұмыс істейтін бағдарлама ешкімге ұнамасы анық.

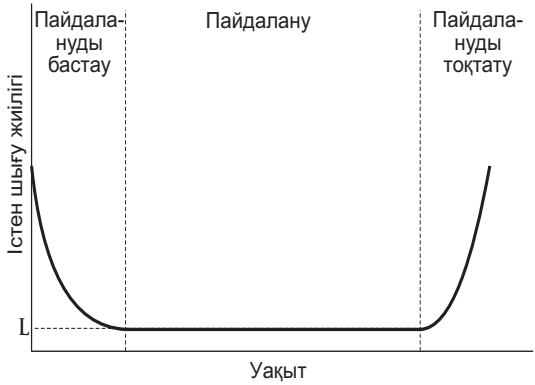
13.1 суретте 1977 жылы Р.Маккол, Б.Ричардс пен С. Уолтерспен ұсынылған сапа факторларының типологиясы көрсетілген.

«IEEE. Standart Glossary of Software Engineering Terms» («IEEE. Бағдарламалық инженерияда қолданылатын стандартты терминдер тізімі») баспасы БӨ сенімділігін жүйе немесе оның құрауышының берілген уақыт мерзімі ішінде ұсынылған жағдайда талап етілетін функцияларды орындау қабілеті ретінде анықтайды. БӨ сенімділігі дәрежесі жасау процесінің жетілуіне тәуелді болады. БӨ сенімділігіне ықпал ететін негізгі көрсеткіш – жасалатын бағдарламалардың қиындығы.

Сенімді БӨ жасау процесі, аппараттық қамтамасыз етуге қарағанда,аппараттық қамтамасыз ету мен 13.2 және 13.3 суретте көрсетілген, мұндағы X – ауытқулардың жоспарлы саны, U-түрлі дәстүрлі қисықтар түріне ие БӨ сенімділігінің модельдерін көрсететін уақытқа тәуелді емес.



13.1 сур. Бағдарламалық өнім сапасы факторының топологиясы



13.2 сур. Аппаратты қамтамасыз ету сенімділігінің U-түрлі қисығы

БӨ сенімділігін бағалау тәсілі әлі толықтай жасалмаған, бірақ егер болған ауытқу туралы мәліметтерді сәйкес бағалауды жасамаса, сенімділіктің шынайы деңгейін анализдеу үшін қажет кеңейтілген статистикалық модельдерді жасау да мүмкін емес болады. Әлі күнге дейін шектеудің шектен тыс санына ие емес сенімділікті бағалаудың бір де бір сенімді сандық тәсілі жасалмады.

БӨ сенімділігі дәрежесін әртүрлі тәсіл арқылы жақсартуға болады, дегенмен, жасау уақыты, оның бюджеттік құны мен БӨ-нің қол жеткізілген сенімділігі үшін төленген жоғары болып көрінетін құны арасында қажетті қатынасқа қол жеткізу қиын болады.

Аппараттық қамтамасыз етуге қарағанда БӨ уақыт өте келе «тозбайды», жай ғана оның жаңа дефектері шыға береді. 13.3 суретте көрсетілген U-түрлі қисық бағдарламаны қолдану бойында ауытқуларды орналастыруды көрсетеді. БӨ-де ауытқулардың пайда болуы аппаратура ауытқулары сипатынан ерекшеленеді, олардың ықтималдығы нөлден жоғары болады.

13.3 сур. Бағдарламалық өнім сенімділігінің U-түрлі қисығы

БӨ-нің қажетті сенімділігін қамтамасыз ету үшін талапты инвестиция размерін анықтауда сенімсіз бағдарламаның пайда болу қаупіне байланысты мәселелерді есептеу керек. БӨ жобалауда шығын мен қол жеткізілетін сенімділік арасындағы балансы көрсетілген қауіптің пайда болуы белгісімен анықталады. Егер қол жеткізілетін сенімділік қауіпті айтарлықтай азайтуға рұқсат етпеса, онда оны қамтамасыз етуге қосымша құрал жұмсаудың қажеті жоқ.

БӨ сенімділігі мәселесінің көбі өміршеңдік маңызды болып табылмайды. Сенімділікке қол жеткізуге байланысты мәселелердің негізгі массасы БӨ тестілеуге жатқызылады.

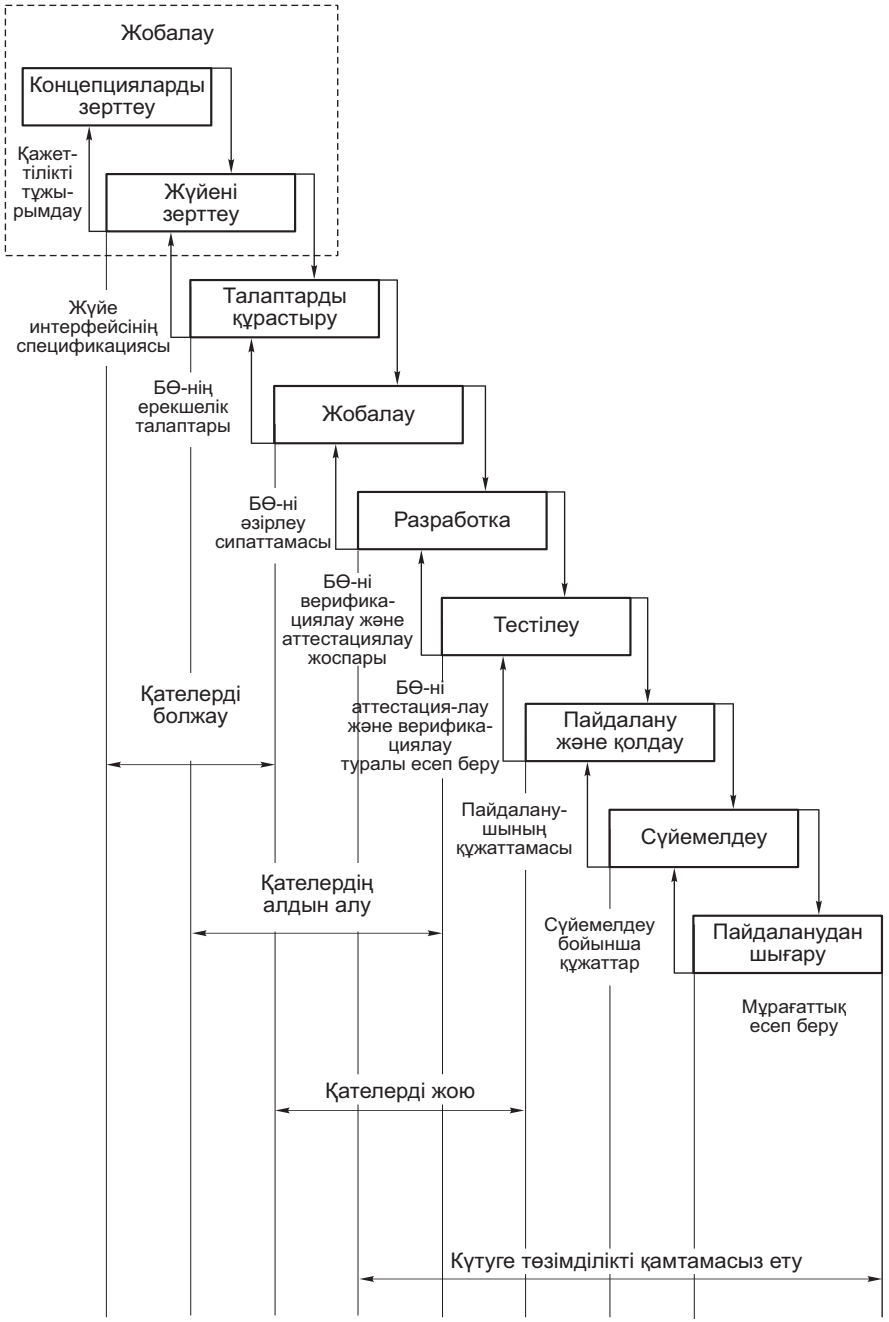
Жоғары берікті БӨ жасауды қамтамасыз ететін төрт әдісті қарастыруға тоқталайық:

- қатені болжамдау — сенімділік моделін жасау, тарихи мәліметтер анализі, қате жөнінде ақпарат жинау, операциялық ортаны профильдеу;
- қатенің алдын алу – формалды әдістерді қолдану, бағдарламаны қайта қолдану, бағдарламаны құрастыру құралдарын қолдану;
- қатені жою – формалды тексеру, верификация мен аттестация;
- бұзылуға тұрақтылықты қамтамасыз ету – мониторинг тәсілін қолдану, шешім верификациясы, артықты, ерекше жағдайлар анализі.

13.3. Бағдарламалық өнімді жасаудың өміршеңдік циклінің әртүрлі сатысында сенімділікті қамтамасыз ету әдістері

БӨ сенімділігін жобаны жасаудың алғашқы сатыларында жоспарлау керек. Жасалатын БӨ сенімділігін анықтау процесі ақпараттың үлкен көлемін жинауды талап етеді.Метрикалық мәліметтерді жинау мен анализ бойынша әрекеттер 5 бөлімде көрсетілген. Өлшеу әдістері БӨ жасаушыларымен бүкіл өміршеңдік циклдің ішінде жасалады. БӨ жасаудың өміршеңдік циклінің әртүрлі сатысында жүзеге асырылатын сенімділікті қамтамасыз ету әдістері 13.4 суретте көрсетілген.

Қатені болжамдау жоспарлау мен талаптарды жасау сатысында орындалады, қателердің алдын алу – талаптарды жасау, жобалау мен жасау сатыларында, қателерді жою – жобалау, жасау мен тестілеу сатыларында. Бұзылуға тұрақтылық кезеңі жасау сатысында басталып, БӨ-нің өміршеңдік циклінің аяқталуына дейін жалғасады.



13.4. сур. БӨ жасаудың өміршеңдік циклінің әртүрлі сатысында жүзеге асырылатын сенімділікті қамтамасыз ету әдістері

Бағдарламалық өнімді жасаудың өміршеңдік циклі сатыларымен сенімділікті қамтамасыз ету әдістерінің байланысы

БӨ жасаудың өміршеңдік циклдерінің сатылары	Негізгі әрекеттер	Қателерді болжамдау	Қателердің алдын алу	Қателерді жою	Бұзылуға тұрақтылықты қамтамасыз ету
Талаптарды жоспарлау мен жасау	Функционалды профильді анықтау	+	+		
	Қателерді анықтау мен классификациясы	+	+		
	БӨ сенімділігінің талапты деңгейінің қамтамасыз етілуінде тапсырыс берушілердің қажеттілік терін анықтау	+	+		
	Альтернативті оқу курстарын өткізу	+	+		
	Сенімділікті қамтамасыз етумен байланысты мақсаттарды анықтау	+	+		
Жобалау мен жасау	БӨ-нің барлық құраушылары арасында сенімділікті қамтамасыз ету бойынша функцияларды бөлу		+	+	+
	Сенімділікке қол жеткізу бойынша мақсаттарды бекіту үшін инженерлермен кездесу		+	+	+
	Профильдің функционалдығы негізінде ресурстарды жинақтау		+	+	+
	Қатені енгізу мен таралуын бақылау		+	+	+
	Алынған БӨ сенімділігін өлшеу		+	+	+

БӨ жасаудың өміршеңдік циклдерінің сатылары	Негізгі әрекеттер	Қате-лерді болжам дау	Қате-лердің алдын алу	Қате-лерді жою	Бұзылуға тұрақ тылықты қамта-масыз ету
Тестілеу	Қолданыстағы профильді анықтау			+	+
	Сенімділікті арттыру сатысын тестілеу			+	+
	Тестілеуді жасау барысын бақылау			+	+
	Жобаның қосымша тестілеуіндегі қажеттілікті анықтау			+	+
	Сенімділіктің талапты сатысына қол жеткізілді ма? Соны анықтау			+	+
Қолдану мен сүйемелдеу	Жобаның соңғы сатыларында персонал қажеттілігін анықтау				+
	БӨ жасау мақсаты мен оның сенімділігін салыстырмалы тексеру				+
	Сенімділіктің қол жетікізілген деңгейімен тапсырыс беруші қанағатталығық деңгейін бақылау				+
	Жаңа қабілетті қосуда сенімділікті тексеру				+
	Қол жеткізілген сенімділікті бір мезгілде өлшеумен өнім мен процесті жақсартуды басқару				+

БӨ жасаудың өміршеңдік циклі сатыларымен сенімділікті қамтамасыз ету әдістерінің байланысы 13.1 кестеде көрсетілген.

13.4. Қатені болжамдау

Қателерді болжамдау сенімді БӨ жасаудың күтпелі тәсілін білдіреді. БӨ жасауда мамандандырылатын жетілген ұйымдар қатені болжамдауды БӨ жобасы/процесін бағалаудың құрамдас бөлігі ретінде жасайды. Болжамданатын модельдер үшін дәлдіктің тіпті кішігірім деңгейіне де қол жеткізудің жалғыз тәсілі мәліметтердің беріктігін қамтамасыз етудің сәйкес тарихи моделіне рұқсатты ұсынудан тұрады. Тарихи мәліметтер анализі мен қателер жөнінде мәліметтер жинағы осы тәсіл үшін маңызды әрекет болып табылады. 13.2 кестеде (шаблон) қатені болжамдау нәтижесі көрсетілу керек.

Функционалды профильді анықтау қатені болжамдауда бірінші әрекет болып табылады. Модульден модульге, функциядан функцияға өту жағдайын бақылай отырып, жүйенің ең осал жерін анық табуға болады. Егер алынған ақпаратты функционалды профильмен біріктірсе, оны қолданудың берілген жағдайында жүйе қаншалықты сенімді болатынын анықтауға болады.

Бағдарламаларды жасауда модульдер арасында бақыланатын өткелдері жасалады.Қатемен толтырылған бағдарламалық модульдерге өтуде сәтсіздік қаупі артады. Мұндай өткелдерді модельдеу қандай да бір стохастикалық процесс арқылы жүзеге асырылады.Нәтижесінде БӨ әртүрлі модульдерінің жасалатын функцияларымен шартталатын БӨ тәртібін математикалық сипаттауда жасалатын функция беріктігін сипаттауды жасау мүмкін

13.2 Кесте

Жоспарлау мен талаптарды жасау сатыларында қателерді болжамдау нәтижелерін есептеу үшін шаблон

Жоспарлау мен талаптарды жасау сатыларында қателерді болжамдаудағы негізгі қателер	Қатені болжамдау нәтижелері
Функционалды профильді анықтау	(Нәтижелерді тізбектеу)
Қателерді анықтау және классификация	Дәл сол
БӨ сенімділігінің талапты деңгейін қамтамасыз етуде тапсырыс беруші қажеттілігін анықтау	»
Альтернативті оқу курстарын өткізу	»
Сенімділікті қамтамасыз етумен байланысты мақсаттарды анықтау	»

Бағдарламалық жүйе өзіндік жасалатын функциялардан тұрады. Жасалатын функция мен жүйемен осы функциялар арасында машиналық уақытты бөлу реті сенімділігі жөнінде көрініс алып, жүйенің өзінің сенімділігі жөнінде қорытынды жасауға болады.

Қатені анықтау мен классификациясы – оларды болжамдаудағы екінші әрекет. 13.1 бөлімшеде көрсетілгендей, БӨ қателері жоба дефекттері немесе қателері, кодтау дефекттері немесе қателері, ұйымдастырушылық қателер, лайықсыз ретке келтіру мен тестілеу қателері нәтижелері болып табылады. Қателерді анықтау олардың қайнаркөзін бекітуді білдіреді. Қателерді классификациялау олардың қиындық деңгейі бойынша жасалады. Борис Бейцермен жасалған классификация тәптіштеудің сәйкес деңгейлеріне ие.

1. Әлсіз —белгілер көбінісе эстетикалық сипатта болады.

2. Бірқалыпты — шығыс мәліметтер дұрыс емес немесе артық.

3. Тітіркендіргіш – жүйенің қате әрекеті.

4. Өте маңызды – бірнеше пайдаланушыны қамтитын жүйе жұмысындағы мәселе (мысалы, клиенттің төлеу түбіртегін есептеудің орнына, жүйе ақшаны басқа шотқа аударады).

5. Экстремалды – пайдаланушылардың кең ортасын қамтитын жүйе жұмысындағы мәселе.

6. Шыдамсыз – мәліметтер базасы ұзаққа және жөндеусіз реттен шығады.

7. Апатты – бағдарламаны орындауды тоқтату жөнінде шешім пайдаланушымен ұсынылады, жүйе бұзылады.

8. Инфекциялық – оның жеке жұмысқабілеттілігі әлі де бұзылмаса да, жүйе басқа жүйелерді бұзады.

Қате көздерінің матрицасы мен олардың классификациясын жасауды аяқтап, 13.3 кестеде (шаблон) көрсетілгендей, қате мәліметтерді сипаттаушы метрикалық көрсеткіштерді бақылау керек.Кестенің әрбір ұяшығы ұқсас жобалар мен өнімдер үшін қолжетімді тарихи ақпаратқа сәйкес қателер жөнінде қорытынды мәліметтерден (қорытынды) тұрады.

БӨ сенімділігінің талапты деңгейін қамтамасыз етуде тапсырыс беруші қажеттілігін анықтау қатені болжамдаудағы үшінші әрекет болып табылады. Бұл қажеттіліктер алын ала анықталған және тапсырыс берушінің алғашқы талаптарында құжаттандырылған. Сенімділіктің талапты деңгейінде тапсырыс беруші қажеттіліктері өлшеу әдістері қамтамасыз ететін дәлдікпен бекітілу керек.

Қате жөнінде қорытынды мәліметтерді есептеу үшін шаблон

Қате деңгейі	Жоба қатесі немесе дефекті	Кодтау қатесі немесе дефекті	Ұйымдас тырушылық қателер	Лайықсыз ретке келтіру	Тестілеу қателері
Әлсіз	Қорытынды	Қорытынды	Қорытынды	Қорытынды	Қорытынды
Бірқалыпты	»	»	»	»	»
Тітіркендіргіш	»	»	»	»	»
Өте маңызды	»	»	»	»	»
Төтенше	»	»	»	»	»
Шыдамсыз	»	»	»	»	»
Апатты	»	»	»	»	»
Инфекциялық	»	»	»	»	»

Барлық талаптар өлшемді болып, сенімділікті қамтамасыз етумен байланысты бақылау, талаптар мүмкіндігін қамтамасыз етуге қарамастан, анықталған нөмірлер беріледі. Төменде нөмірлерге лайықты сенімділікке қатысты бекіту мысалдары көрсетілген.

1. Спутниктермен басқару жүйесінің сенімділігі пайдалы жүктемені бөлу сәтінен 15 жылға тең кезеңге 0,999-ға 95% сенімнен бағаланады.

2. Авиациялық жүйе үшін критикалық қателер арасында орташа 500 ұшу сағаттары сай.

3. Өзін өзі басқарудың кірістірілген жүйесі 60% ықтималдықпен ракетаны жарамсыз деп анықтайды

4. Өзін өзі басқарудың кірістірілген жүйесі 1%-дан аз ықтималдықпен ақаусыз ракетаны ақаулы деп есептейді.

5. Жұмыс істейтін БӨ-ні жөндеудің орташа уақыты 30 мин немесе одан да аз болады.

6. Өзін өзі басқарудың кірістірілген жүйесінің жұмысын аяқтаудың максималды уақыты 20 с.құрайды.

7. БӨ-ні толық жүктеу мен толықтай өзін өзі басқаруды іске

асырудың орташа уақыты 10 мин құрайды.

8. Құжаттаманың бір интерактивті бетін жаңартудың орташа уақыты 30 мин құрайды.

Альтернативті оқу курстарын өткізу – қатені болжамдаудың төртінші әрекеті. Клиенттік функционалды профиль мен өткен жүйелерден алынған қателерді классификациялау жөнінде ақпарат арқылы мақсаттар тарихи мәліметтерді қолдай ма, соны анықтау үшін ерекше талаптар сараптанады. Талаптарда жасалған сенімділік мақсаттарына қол жеткізу ықтималдығын анықтау үшін курстар анализі жасалады.Ұқсас өнімдер мен жүйелермен байланысты тарихи мәліметтердің жоқтығында сенімділіктің жасанды бекітілген деңгейіне қол жеткізу ықтималдығы өте төмен. Бұл сатыда жоба басшысына жаңа БӨ сенімділігінің мүмкін болатын деңгейлерін анықтау үшін қолданылатын экстенсивті жүйелік модельдерді жасау керек. Формалды әдістер сияқты құралдар критикалық қосалқы жүйеге байланысты математикалық дәлелдеулер үшін сенімділік талаптарына қолданылады. Бұл қымбат тұратын процесс сенімділік жөнінде тарихи мәліметтер көзінің болмауы жағдайында ғана қолданылады.

Сенімділікті қамтамасыз етумен байланысты мақсаттарды анықтау – альтернативті оқу курстарын өткізу нәтижелеріне негізделетін қателерді болжамдаудың бесінші әрекеті. Сенімділікті қамтамасыз ету мақсаттарының соңғы жиынтығы қолда бар талаптарды өзгертуге мүмкіндік бере отырып, талаптарды анықтау процесіне беріледі. Осы әрекет арқасында жиналған ақпарат пен өткізілген анализ нәтижелері талаптарды сипаттау үшін беріледі. Сенімділікке қатысты мақсаттар мен талаптар бүкіл жүйенің сенімділігін бекіту мен пайдаланушы мақұлдауын алу үшін қолданылады.

13.5. Қателердің алдын алу

Қателердің алдын алу өзіне жүйелік талаптардың интерактивті нақтылауы мен БӨ-нің техникалық сипаттамасын жоба әдіснамасы мен кодтаудың қолайлы тәсілдерімен тексерілетін модельдеумен тең жасауды қосады. Сенімділікті қамтамасыз етудің бұл әдісі БӨ талаптары, жобалау мен жасауды жүзеге асыру сатыларында қолданылады. Қателердің алдын алу жобалау процесінің белсенді бөлігі болып табылады. Қателердің алдын алудың алғашқы сатысы жүйенің өзі мен БӨ талаптарын зерттеуден тұрады. Алғашқы қадамдар оның сенімділігіне қол жеткізуге жәрдемдесу үшін шақырылған тәсілдерден тәуелсіз БӨ сенімділігіне қатысты талаптар мен мақсаттарды анализдеу

үшін қажет.

Қателердің алдын алудың екінші негізгі сатысы – жобаны жобалау мен жасау. Бұл ретте сенімділікті қамтамасыз ету бойынша функциялар БӨ-нің барлық құрауыштары арасында бөлінеді. Нәтижесінде құрауыштар жасалатын жобалау процесі анықталады. Модульдердің параллельдік деңгейін жақсарту мен байланыстылығын арттыру құрауыштар сенімділігін арттыруды қамтамасыз етеді. Жобалауға қатысты ең жақсы атқарымдар жасаған жөн, және олар құрылымдық немесе объектілі – бағдарлы болуына қарамастан. Осының арқасында сенімділіктің жоғары деңгейіне ие БӨ жасау мүмкіндігі қамтамасыз етіледі.

БӨ жасаудың өміршеңдік циклінің алғашқы сатыларында сенімділікті қамтамасыз етудің мақсаты мүмкін болатын қателердің алдын алу болып табылады. 13.4 кестеде (шаблон) қателердің алдын алу бойынша жұмысты қолдау үшін ұйымда жасалатын барлық әрекеттер көрсетілді.

Алдын ала анықталған және құжатталған сенімділікті қамтамасыз етудің мақсаты жобалау сатысы барысында барлық модульдер үшін бекітілу керек. Жоба басшысына ресурстарын осы жобалау барысында қолданылатын және аттестациядан өтетін функционалды профиль негізінде жинақтау керек.

13.4 Кесте

Талаптар, жобалау мен жасауды құрастыру сатыларында қатенің алдын алу бойынша әрекеттерді есептеу үшін шаблон

Талаптар,жобалау мен жасауды құрастыру сатыларында қателердің алдын алу бойынша негізгі әрекеттер	Қатенің алдын алу бойынша әрекет құрамы
Функционалды профильді анықтау	(Әрекеттер тізбегі)
Қателерді анықтау және классификациясы	Дәл солай
БӨ сенімділігінің талапты деңгейін қамтамасыз етуде тапсырыс беруші қажеттілігін анықтау	»
Альтернативті оқу курстарын өткізу	»
Сенімділікті қамтамасыз етуге байланысты мақсаттарды анықтау	»
БӨ-нің барлық құрауыштары арасында сенімділікті қамтамасыз ету бойынша функцияларды бөлу	»
Сенімділікке қол жеткізу бойынша мақсаттарды бекіту үшін инженерлермен кездесу	»

Функционалды профиль негізінде ресурстарды жинақтау	»
Кіріс пен қателердің таралуын басқару	»
Алынған БӨ сенімділігін өлшеу	»

Қателердің белсенді алдын алудың жалғыз тәсілі – кіріс пен қателердің таралуын басқару.

Экспертті бағалар мен инспекциялық тексерулер – бір сатыда пайда болған қателерді белсенді қысқарту мен олардың басқа сатыға өтуінің алдын алудың дәстүрлі тәсілі. Тағы бір маңызды әрекет – алынған БӨ сенімділігін өлшеу. Internet желісімен байланыстырылған инструменталды құралдарды кең қолдануда,және бірігіп қолданылатын қолжетімді кітапханаларда бағдарламалық қамтамасыз ету (SOUP — Software Of Uncertain Pedigree) өнімнің бөлігі болады. Жобаны жасау тобы SOUP-құрауыштардың верификациясы, аттестациясы, сенімділікті қабылдау мен бағалау процесіне мұқтаж болады. Бұл «нөлден» бастап жасалған БӨ құрауыштарымен болған жағдайдағыдай,конфигурацияны бақылаудың сол деңгейімен формалды проесс болу керек. Бағдарламаны қайталап қолдану жасаушы өнімділігін арттыруды қамтамасыз етеді. Алайда, бұл ретте жасырысн мәселелер мен дефекттер туындауы мүмкін.

13.6. Қателерді жою

«Алдын алуға кеткен бір күн, салдарды жөндеуге кеткен жылға тең» деген мақал осы жерде орынды болады. Жүйеде пайда болатын қатені жою бастапқы сатыда оның алдын алуға қарағанда 10-100 есе қымбат тұрады.

Қатені жою процесі БӨ жобалау сатысында басталып, жасау мен тестілеу сатыларына таралады. Функционалды профильдеу бойынша жұмыс аяқталғаннан кейін келесі қадам жасалады – жүктемедегі сынақ деген атауға ие болған БӨ сенімділігін арттыру деңгейін тестілеу. Мұндай тестілеудің мақсаты – жүйе реттен шығатын жүктеме жиынтғын анықтау. Ол тестілеуді жасау барысын бақылауда арнайы мәнге ие болатын формалды процесс. Тест нәтижелері жобаның бастапқы сатыларында сенімділікті болжамдау үшін қолданылатын модельдердің қайта калибрлеуі үшін оларды қолдану мақсатымен анализденеді. Жоба басшысының міндеттерінің бірі – жасау процесін тұрақты жетілдіруді қолдау. Өткен жобаларды жасауда алынған мәліметтерді қолданудың арқасында ағымдағы жобада ұйымның оқу мен жетілуге қабілеті қолдау табады.

Жобалау, жасау мен тестілеу сатыларында қателерді жою бойынша әрекеттер есебі үшін шаблон

Жобалау, жасау мен тестілеу сатыларында қателерді жою бойынша негізгі әрекеттер	Қатені жою бойынша әрекеттер жинағы
БӨ барлық құрауыштары арасында сенімділікті қамтамасыз ету бойынша функцияларды бөлу	(Әрекеттер тізбегі)
Сенімділікке қол жеткізу бойынша мақсаттарды бекіту үшін инженерлермен кездесу	Дәл солай
Функционалды профиль негізінде ресурстарды жинау	»
Кіріс пен қатенің таралуын басқару	»
Алынған БӨ сенімділігін өлшеу	»
Қолданыс профилін анықтау	»
Сенімділіктің арту деңгейін тестілеу	»
Тестілеуді жасау барысын бақылау	»
Жобаның қосымша тестілеуінде қажеттілікті анықтау	»
Сенімділіктің талапты деңгейіне қол жеткізілді ма, соны анықтау	»

БӨ жасаудың өміршеңдік циклінің ортаңғы сатысында сенімділікті қамтамасыз ету бойынша күш дефекттерді жоюға жұмылдырылады. 13.25 кестеде (шаблон) жасаудың өміршеңдік циклінің негізгі сатыларында жасалатын қателерді жою бойынша ұйымда қабылданатын барлық әрекеттер тізбектелу керек. Қажетті тестілеуді өткізетін модульді жобалау тестіленетін мәліметтердің анализ нәтижелері болып табылады. Қосымша жүктememен тестілеу нәтижелері сәйкес болмауы мүмкін, нәтижесінде сенімділікті қамтамасыз етудің барлық мақсаттарының жүзеге асырылуын бекіту қиын болады. Кейбір модульдер, мүмкін, «қара» немесе «ақ» жәшік тәсілімен қайта тестілеуден өтуі керек болады. Бұл ретте, көлемде регрессиялық тестілеу кеңейтіліп, артылу керек. Қатені жоюдың аяқталу сәтінде жоба басшысы сенімділікті қамтамасыз ету мақсатына қол жеткізілгенді туралы көрсететін нәтижемен.

13.7. Бұзылуға тұрақтылықты қамтамасыз ету

БӨ бұзылуға тұрақтылығына қол жеткізу принциптері (13.1 бөлімшеде анықтамасын қараңыз) аппараттық қамтамасыз етудің бұзылуға тұрақтылығына қол жеткізу принциптерінен айтарлықтай деңгейде айрықшаланады. Аппараттық қамтамасыз етудің бұзылуға тұрақтылығы негізінде жасалған жүйеде параллельді түрде аппараттық құралдардың бірнеше жиынтығы қызмет етеді. Бұл ретте бір құрылғы істен шықса, басқасы жұмысын жалғастыру үшін барлық құрылғылардың қосарлануы жасалады.

Сол түрде БӨ бұзылуға тұрақтылығын іске асыру талпынысы, яғни әртүрлі процессорлармен бірдей бағдарламаны параллель жасау, сол бағдарламаның екінші көшірмесі бірінші көшірмесінен бірнеше секундқа қалатындығына әкеледі. Бағдарламаның жеке көшірмесін жай жасау БӨ бұзылуға тұрақтылығына әсер етпейді.

БӨ жасаудың өміршеңдік циклінің ортаңғы сатыларынан бастап, сенімділікті қамтамасыз ету бойынша күш бұзылуға тұрақтылыққа қол жеткізуге жұмсалады. 13.6 кестеде (шаблон) жасаудың өміршеңдік циклінің негізгі сатыларында орындалатын бұзылуға тұрақтылықты қамтамасыз ету бойынша ұйымда шара қолданылатын барлық әрекеттер көрсетілу керек.

Бұзылуға тұрақтылықты қамтамасыз ету процесі БӨ жасау сатысында басталып, тестілеу, қолдану мен сүйемелдеу сатыларына таралады. Бұл процестің аяқталуы БӨ тозуы сәтімен анықталады. БӨ қалыпты режимде жұмыс істеп тұрғанша, сенімділікті қамтамасыз етудегі бұзылуға тұрақтылық тәсіл айтарлықтай кең қолданылады.

Бұзылуға тұрақтылықты қамтамасыз ету қатені жою сатысының логикалық жалғасуы болып табылады. Бұл жағдайда қатені жоюда қолданылатын барлық әрекеттер іске асырылады. Айырмашылық БӨ жасау процесі аяқталғаннан кейін қуылатын мақсаттан тұрады. БӨ сүйемелдеумен айналысатын персонал қажеттілігі БӨ-нің алдыңғы нұсқаларын жасауға қатысты ақпарат есебімен қана анықталады. Ұйымның басқа БӨ жасаудан кейін табылған қателер жөнінде мәліметтер базасына рұқсаты болу керек.

Қателерді жою үшін қабылданған әрекеттер мен қателерді сипаттауды өзіне қосатын ақпараттық құрылым БӨ сүйемелдеу сатысында қажетті еңбек шығынын бағалау үшін қолданылады. Жасау сатысында табылған қателерді бағалау нәтижелері жөнінде хронологиялық мәліметтерді қолдану және жаңа өнім (LOC) көлемі мен басқа БӨ көлемі арасында қатынасты білу арқылы өнімнің жаңа нұсқасында қателерді жою үшін қажетті қалған қате мен еңбек шығынының жылдам бағасын жасауға

болады.

Жасалатын БӨ үшін мәлімттер жиынтығын қамтамасыз ету үшін жоба басшысы БҚ жасау мақсаттарын ескере отырып, қолданыс сынақтары барысында БӨ сенімділігін бақылау керек. Ол тапсырыс берушілердің қанағаттануы деңгейін сенімділіктің қол жеткізілген деңгейімен бақылауға жол береді. Соңғы пайдаланушы – БӨ сенімділігі жөнінде ақпараттың ең жақсы көзі.

Жасау, тестілеу, қолдану мен сүйемелдеу сатыларында бұзылуға тұрақтылықты қамтамасыз ету бойынша әрекеттер есебі үшін шаблон

Кесте 13.6

Жасау, тестілеу, қолдану мен сүйемелдеу сатыларында бұзылуға тұрақтылықты қамтамасыз ету бойынша негізгі әрекеттер	Бұзылуға тұрақ тылық бойынша әрекеттер жинағы
БӨ барлық құрауыштары арасында сенімділікті қамтамасыз ету бойынша функцияны бөлу	(Әрекеттер тізбегі)
Сенімділікке қол жеткізу бойынша мақсат тарды бекіту үшін инженерлермен кездесу	Дәл солай
Функционалды профиль негізінде ресурстарды жинау	»
Кіріс пен қатенің таралуын басқару	»
Алынған БӨ сенімділігін өлшеу	»
Қолданыс профилін анықтау	»
Сенімділіктің арту деңгейін тестілеу	»
Тестілеуді жасау барысын бақылау	»
Жобаның қосымша тестілеуінде қажеттілікті анықтау	»
Сенімділіктің талапты деңгейіне қол жеткізілді ма, соны анықтау	»
Жобаның соңғы сатыларында персонал қажеттілігін анықтау	»
БӨ жасалу мақсаты мен оның сенімділігін салыстырмалы тексеру	»
Сенімділіктің қол жеткізілген деңгейіне тапсырыс берушінің қанағаттану дәрежесін бақылау	»
Жаңа қабілеттерді қосуда сенімділікті тексеру	»
Бір уақытта өнім мен процестің жақсаруын басқару мен қол жеткізілген сенімділікті өлшеу	»

Жоба басшысы БӨ-нің жаңа қабілетін тапсырыс берушіге ұсыну уақытын анықтау керек. Осы қабілетке қатысты барлық қателер жойылмайынша, тапсырыс берушіге өнімнің жаңа қабілетін ұсынудың керегі жоқ. Жаңа қабілеттер жинағы мен жойылған қателермен БӨ-ні жеткізу БӨ жасаушы-ұйымдар үшін ең тиімді тәжірибе болып табылады.

Бір уақытта өнім мен процесті жақсарту мен қол жеткізілген сенімділікті өлшеу тапсырыс беруші тәжірибесіне негізделіп жиналған ақпарат негізінде сенімділікті арттыру процесін толықтырады. Бұл ретте БӨ қателері жойылады және жасаудың үздіксіз процесі жасалады. Сенімділікті бағалау айтарлықтай қымбат тұратын процедура болып табылады. Оның нәтижелері оқытушы ұйымға жіберіледі.

13.8. Бағдарламалық өнім сенімділігін қамтамасыз ететін құралдар. Сенімділікті қамтамасыз ету жоспары

Сенімділікті қамтамасыз етудің барлық құралдары үшін негіз болып оның статистикалық анализі саналады, сондықтан мәліметтер жиынтығын анализдеу мен элементарлы статистикалық есептеулерді (Мысалы, MS Excel) жасауға қабілетті кез-келген бағдарламалық құралдар осы міндеттер үшін қолданыла алады.

БӨ сенімділігін қамтамасыз ету бойынша қызмет өте қымбат тұратын қызмет болып саналады. БӨ-ні жасау бойынша кез-келген қызмет сияқты ол жоспарланып, құжаттануы керек. Төменде БӨ сенімділігін қамтамасыз ету жоспарының сызбасы көрсетілген.Ұсынылатын жоспар IEEE, SEI және ISO-мен жарияланған көптеген жоспардың түрлендірушісі болып табылады.

Сенімділікті қамтамасыз ету жоспарының сызбасы

1. Сенімділікте қажеттілікке қатысты қорытынды.
2. Сенімділікті қамтамасыз етуге қатысты анықтамалар, акронимдер мен қысқартулар, сұрақтарға сілтемелер.
3. Қаіптерді басқару бойынша әрекеттермен өзара байланыс:

a. Спецификалық қауіпті азайту;
b. Жоба бюджетін сақтау;
c. БӨ сенімділігінің ықпалы;
d. Сенімділікті қамтамасыз ету үшін арналған тәсілдерді сипаттау (қатені болжамдау; қатенің алдын алу; қатені жою;бұзылуға тұрақтылықпен қамтамасыз ету).

4. Қатені болжамдау әдісі:

a. Функционалды профильді анықтау;

b. Қатені анықтау;

c. Қате мен бұзылу классификациясының сызбасы;

d. Сенімділіктің талапты деңгейін қамтамасыз етуде тапсырыс беруші қажеттілігін анықтау;

e. Альтернативті оқу курстарын өткізу жоспары;

f. БӨ сенімділігін қамтамасыз етумен байланысты мақсаттарды анықтау.

5. Қатенің алдын алу әдісі:

a—f жоспардың 4т. көрсетілгендей;

g. БӨ барлық құрауыштары арасында сенімділікті қамтамасыз ету бойынша функцияларды бөлу;

h. Сенімділікті қамтамасыз ету мақсатын қанағаттандыратын процесті жасау;

i. Функционалды профиль негізінде ресурстардың жиналу жоспары;

j. Кірісті басқау мен қатенің таралуы жоспары.

6. Қатені жою әдісі:

a—d жоспардың 5 т. g—j тармақшасына сай келеді;

e. Сенімділікті бағалау жоспары;

f. Қолданыс профилін анықтау;

g. Сенімділіктің артылу деңгейін тестілеу жоспары;

h. Тестілеуді жасау барысын бақылау жоспары;

i. Қосымша тестілеу жоспары;

j. Сенімділік мақсаттарын сертификациялау процесі.

7. Бұзылуға тұрақтылықты қамтамасыз ету әдісі:

a—j жоспардың 6т. сәйкес;

k. Жобаның соңғы сатысында персонал қажеттілігін анықтау;

l. БӨ жасау мақсаты мен тапсырыс берушіде сынақта оның сенімділігінің салыстырмалы тексерісі;

m. Сенімділіктің қол жеткізілген деңгейіне тапсырыс берушінің қанағаттануы деңгейін бақылау;

n. БӨ сенімділігін бақылау арқылы БӨ жаңа қабілетін ұсыну уақытын есептеу;

o. Бір уақытта өнім мен процесті жақсартуды басқару мен қол жеткізілген сенімділікті өлшеу.

8. Сенімділікті қамтамасыз ету жоспарын мақұлдау.

Бақылау сұрақтары

1. «Бағдарламалық өнім сенімділігі» ұғымына анықтама беріңіз.

2. Бағдарламалық өнімді қолдану уақытында қатенің таралу қисығын сызып, түсіндіріңіз.

3. Жоғары сенімді бағдарламалық өнімді жасаудың негізгі әдістерін атап, сипаттаңыз.

4. Кезеңдерде сенімділікті қамтамасыз етудің қандай әдістері қолданылады:

а) талаптарды жоспарлау мен жасау;

б) жобалау мен жасау;

в) тестілеу;

г) қолдану мен сүйемелдеу.

5. «Бұзылуға тұрақтылық», «мәселе», «өңдеудегі қате», «процесс», «процесті жасауда бұзылу», « процесті жасауда ауытқу», «тұрақтылық», «бағдарламалық өнім қатесі» ұғымдарына анықтама беріңіз.

6. Бағдарламаны болжамдау сатыларын тізбектеп, сипаттаңыз.

7. Қатенің негізгі класстарын атаңыз.

8. Қате көзінің матрицасының мақсатын түсіндіріңіз.

9. Келесіге бағытталған негізгі әрекеттерді атап өтіңіз:

а) қатенің алдын алу;

б) қатені жою;

в) бұзылуға тұрақтылықты қамтамасыз ету.

10. Сенімділікті қамтамасыз ету жоспарының негізгі тармақтарын атап, түсіндіріңіз.

14 БӨЛІМ

UML ТІЛІНІҢ НЕГІЗГІ ҰҒЫМДАРЫ МЕН МАҚСАТЫ

14.1. UML тілінің мақсаты

UML тілі - бағдарламалық қамтамасыз ету, бизнес-процестер мен басқа да әртүрлі жүйелер құрауыштарын сипаттау, визуализация, жобалау мен құжаттау үшін жасалған визуалды модельдеудің жалпыға ортақ тілі. UML тілі модельдеудің бір уақытта оңай және мықты құралы болып табылады. Ол әртүрлі мақсатты бағыттар үшін қиын жүйелердің концептуалды, логикалық және графикалық модельдерін жасау үшін тиімді қолданыла алады. Бұл тіл өзіне үлкен және қиын жүйелерді модельдеуде соңғы жылдары табыспен қолданылған бағдарламалық инженерия әдістерінің ең жақсы қасиеттерін жинады.

UML тілі объектілі-бағдарлы анализ бен жобалау (ОБАЖ) әдістерімен таныс көптеген бағдарламашылар мен жасаушылармен зерттеліп,қолданылатын базалық ұғым санына негізделген. Бұл ретте базалық ұғымдар объектілі модельдеу мамандары қосымшаның әртүрлі салаларында үлкен және қиын жүйелердің модельдерін өздері жасау мүмкіндігін алатындай қиыстырылып, кеңейтіледі.

UML тілін конструктивті қолдану объектілі-бағдарлы анализ бен жобалау процесінің ерекшеліктері мен қиын жүйелерді модельдеудің жалпы принциптерін түсінуге негізделеді. Қиын жүйе моделін жасау үшін көрінетін құралды таңдау модельдердің мәліметтерін қолдану арқылы шешілетін міндеттердің алдын алады. Бұл ретте қиын жүйенің моделін жасаудың негізгі принципінің бірі ретінде *абстракциялау* принципі саналады, ол өз функциясының жүйесі немесе оның мақсатты бағытын жасауға тура қатысы бар жобаланатын жүйе аспектілерін ғана модельге қосуды көрсетеді. Бұл ретте барлық екінші кезекті бөлшектер алынған модельдің анализі мен зерттеу процесін қиындатпас үшін жіберіледі.

Қиын модельдерді жасаудың басқа принципі – *көпмодельділік* принципі. Бұл принцип ешқандай жалғыз модель баламалықтың жеткілікті деңгейінде қиын жүйенің әртүрлі аспектілерін сипаттай алмайтындығы жөнінде пайымдау береді.ОБАЖ

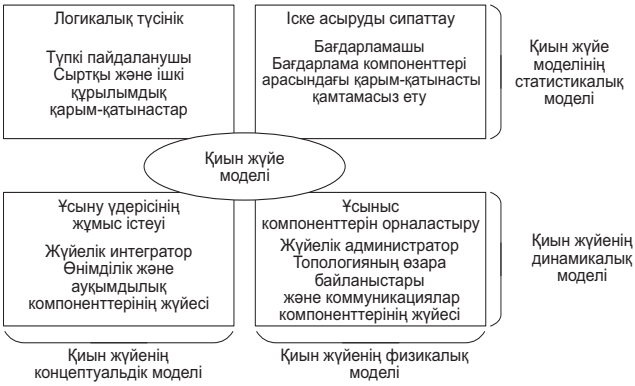
әдіснамасына қатысты ол қиын жүйенің толық моделі жүйенің әрекеті немесе құрылымының кейбір аспектісін баламалы көрсететін өзара байланысқан көріністер (views) санын болдыратындығын білдіреді. Бұл ретте қиын жүйенің ең ортақ көрінісі ретінде бірнеше басқа жеке көріністерге бөлінетін статикалық және динамикалық көріністерді санауға болады. Қиын жүйенің феномені оның ешқандай жалғыз көрінісі оның барлық ерекшеліктерін баламалы көрсету үшін жеткіліксіз болатындығынан тұрады.

Қолданбалы жүйелік анализдің тағы бір принципі қиын жүйе *моделінің иерархиялық жасалуы* принципі болып табылады. Бұл принцип бекітілген көріністер аясында абстракциялау немесе тәптіштеудің әртүрлі деңгейлерінде модельді жасау процесін қарастыруды білдіреді. Бұл ретте қиын жүйенің бастапқы немесе алғашқы моделі жалпы көрініске (метакөрініске) ие болады. Мұндай модель жобалаудың бастапқы сатысында жасалады және модельденетін жүйенің көп бөлшектеріне ие болмауы мүмкін.

Сөйтіп, ОБАЖ процесін жалпы модельдер мен концептуалды деңгейдегі көріністерден логикалық және физикалық деңгейлердің жеке және тәптіш көрінстеріне деңгей бойынша өту ретінде қарастыруға болады. Бұл ретте ОБАЖ-дың әрбір сатысында аталмыш модельдер тізбекті түрде қиын жүйенің нақты іске асуының әртүрлі аспектілерін көрсетуге жол беретін бөлшектердің көп санымен толықтырылады. ОБАЖ модельдердінің өзара байлансуының жалпы сызбасы 14.1 кестеде көрсетілген.

ОБАЖ бен UML тіліндегі «физикалық модель» термині жүйелер модельдерінің жалпы классификациясында жалпы қабылданғаннан ерекшеленетін түсіндірмеге ие болады.

Соңғы жағдайда жүйенің физикалық моделі ретінде түпнұсқа формасындағы қабілеттерге ие кейбір материалдық конструкция деп түсінеді. Мұндай модельдердің мысалы ретінде техникалық жүйе моделі (ұшақ, кеме), сәулет ғимараттары (ғимараттар, аудандар) саналуы мүмкін. Бұл терминнің ОБАЖ бен UML тілінде анықтамасына келсек, мұнда физикалық модель кейбір техникалық база мен нақты өндірушілердің есептеуіш платформасында оның іске асырылуы көзқарасынан жобаланатын жүйенің құрауыш бөлігін көрсетеді.



14.1 сур. Объектілі-бағдарды анализ бен жобалау процесінде қиын жүйе моделі мен көрінісі өзара байланысының жалпы сызбасы

UML тілін жасау келесі мәселелерді шешуді қарастырды

1. Пайдаланушы билігіне ұсынуға әртүрлі мақсатты бағыттың қиын жүйесі моделін жасау мен құжаттау үшін арнайы арналған визуалды модельдеудің оңай қабылданатын және анық тілін беру. ОБАЖ әдіснамасының әрі қарайғы дамуы мен жаппай қолданылуының маңызды факторы болып модельдеудің сәйкес тілінің негізгі конструкциясының интуитивті анықтығы мен түсініктілігі табылады.UML тілі өзіне жүйелердің метамодельдерін ұсыну үшін тек абстрактілі конструкциялар қосып қана қоймай, сонымен қатар анық семантикаға ие нақты түсініктердің бір қатарын қосады.Ол UML тіліне бір уақытта әртүрлі қосымшалар үшін модельді ұсыну әмбебаптылығына ғана емес, сонымен қатар нақты жүйелерге қатысты осы модельдерді іске асыру бөлшектерін сипаттау мүмкіндігіне қол жеткізуге мүмкіндік береді.

Жүйелік модельдеу тәжірибесі кейбір метадеңгейде тілді абстрактілі сипаттау нақты уақытта жүйе жобасын іске асыруды мақсат етіп қоятын жасаушылар үшін жектіліксіз екендігін көрсетті.Қазіргі таңда қиын жүйені модельдеудің жалпы әдіснамасы мен қосымшаны жылдам жасаудың инструменталды құралдары арасында кейбір концептуалды айырмашылық бар. Дәл осы айырмашылық UML тілін толықтыру үшін шақырылған.

UML тілінің базалық конструкциясын дұрыс түсіну үшін объектілі-бағдарлы бағдарламалаудың кейбір дағдыларын иемдену ғана жеткіліксіз, сонымен қатар жүйе моделін жасау процесінің жалпы проблематикасын елестеткен де жақсы. Осы көріністердің интеграциясы ОБАЖ жаңа парадигмасын

жасайды, оның практикалық нәтижесі мен орталық өзегі UML тілі болып табылады.

2. UML тілінің бастапқы ұғымын нақты пәндік салада жүйе моделін нақты ұсыну үшін кеңею мен мамандану мүмкіндігімен жабдықтау. UML тілі ресми тіл болғанымен – сәйкестендіру тілі, оны сипаттау ресмилігі дәстүрлі формальнологиялық тілдерден де, бағдарламалаудың танымал тілдерінен де айрықшаланады. OMG (Object Management Group — 1989 жылы UML тілінің индустриалды стандартын жасау үшін құрылған консорциум) жасаушылары UML тілі басқа тілдер сияқты нақты пәндік салалар үшін дағдылануы мүмкін деп санайды. Ол UML тілінің сипаттамасының өзінде тілдің өз элементі болып табылатын және кеңейту ережесі формасында өзіндік сипаттамаға ие базалық ұғымдардың кеңею механизмі жатқандықтан мүмкін болады.

Сол уақытта OMG жасаушылары қандай да бір себеп бойынша тілдің базалық ұғымдарын қайта анықтауды қажетсіз деп санайды. Ол олардың семантикасының қайталанба интерпретациясы мен мүкін болатын шатасуға алып келуі мүмкін. UML тілінің базалық ұғымдарын ендігі өзгертудің керегі жоқ, оны кеңейту үшін қажеттілікке қарағанда. Барлық пайдаланушылар кеңею механизмін қолданбай UML тілінің базалық конструкциясын ғана қолданумен әдеттегі қосымшалардың көбісі үшін жүйе моделін жасауға қабілетті болулары керек. Бұл ретте жаңа ұғымдар мен нотацияларды қолдағы базалық ұғымдар жүйе моделін жасау үшін жеткіліксіз болған жағдайда ғана қолдану керек.

UML тілі базалық ұғым мамандандыруына да жол береді. Мәселе нақты қосымшаларда пайдаланушылар қолдағы базалық ұғымдарды UML тілінде осы ұғымдардың семантикасына қарсы болмайтын жаңа сипаттамалар немесе қабілеттермен толықтыра білу керектігі жөнінде.

3. UML тілінің сипаттамасы бағдарламаудың нақты тілдері мен бағдарламалық жүйені жобалаудың инструменталды құралдарына тәуелді емес модельдердің мұндай сипаттамасын қолдады. UML тілдерінің конструкциясының ешқайсысы бағдарламалаудың танымал тілдерінде оны іске асыру ерекшеліктеріне тәуелді болмау керек. UML тілінің жеке ұғымдары соңғысымен тығыз байланыста болғанымен, бағдарламалаудың қандай да бір конструкциясы формасында оның қатаң интерпретациясы дұрыс деп саналмайды.Басқаша сөзбен айтқанда, OMG жасаушылары UML тілінің қажетті қабілеті ретінде оның контекстті-бағдарламалық тәуелсіздігін санайды.

Басқа жағынан, UML тілі бағдарламалаудың кез-келген тілінде өз конструкциясын іске асырудың әлеуетті мүмкіндігіне ие болу керек. Әрине, бірінші кезекте, C++, Java, Object Pascal сияқты объектілі-бағдарлы жобалау концепциясын қолдайтын тілдер жөнінде айтылады. UML тілінің осы қабілеті оны қиын жүйені модельдеу мәселесін шешудің қазіргі құралы етеді. Сол уақытта UML тілінің конструкциясын бағдарламалық қолдау үшін арнайы инструменталды CASE-құралдар жасалуы мүмкін. Соңғылардың болуы UML тілінің кең таралуы мен қолданылуы үшін принципиалды маңызға ие.

UML тілінің сипаттамасы ОБАЖ жалпы ерекшеліктерін түсіну үшін семантикалық базисті қосуды қамтамасыз ету. Бұл ерекшелікті айтқанда UML тілінің өз-өзіне жеткіліктілігін базалық конструкцияны ғана емес, сонымен қатар, ОБАЖ жалпы принциптерін ұғыну үшін айтылады. UML тілі бағдарламалаудың тілі ретінде емес, жүйені объектілі-бағдарлы модельдеу мәселелерін шешу үшін құрал ретінде қызмет ететіндіктен оның сипаттамасы ОБАЖ үшін мүмкіндігінше өзіне барлық қажетті түсініктер қосу керек. Бұл қабілетсіз UML тілі пайдасыз және қиын жүйенің ОБАЖ мәселесімен таныс емес пайдаланушыларға қажет етілмейтін тіл болып қалуы мүмкін.

Басқа жағынан, UML тілін ұғыну үшін маңызды ақпаратқа ие қосымша қайнаркөздерге қандай да болсын сілтемелер, OMG жасаушыларының ойынша, алынып тасталуы керек. Ол ОБАЖ процесінің принципалды ерекшеліктерін бірнеше рет түсіндіру мен осы ерекшеліктерді UML тілінің базалық конструкциясы түрінде жүзеге асыруды болдырмауға мүмкіндік береді.

4. Объектілі инструменталды құралдар нарығының дамуын қолдау. Күштері OMG аясында жинақталған 800-ден астам бағдарламалық және аппараттық құрал өндірушілер қазіргі ақпараттық технологиялр мен объектілі технологияны қолдайтын инструменталды құрал нарығында кең дамудың коммерциялық табысының негізінің даму болашағын көреді. Объектілі технология жөнінде айтқанда, бірінші кезекте, OMG жасаушылар CORBA жобалау жүйесі мен UML тілінің технологиялық шешімі жиынтығы жөнінде сөз қозғайды. Бұл көзқараста, UML тіліне CORBA әртүрлі объектілі компоненттерін суреттеу мен құжаттау үшін базалық құрал рөлі беріледі.

5. Объектілі технология мен ОБАЖ сәйкес ұғымдарының таралуына себепкер болу. Бұл міндет алдындағысымен тығыз байланысты және онымен ұқсастықтары көп. Егер қарастырудан UML тілінің иірімділігі мен күштілігін алып тастап, осы тіл

мүмкіндіктерінің объективті көрінісін жасаса, онда келесі қорытындыға келуге болады. Жасаушылардың айтарлықтай үлкен тобының күштері өздерін тәжірибеде сәтті көрсеткен визуальды модельдеудің көптеген танымал технологиясының UML тілі аясында интеграцияға бағытталды. Бұл құрылымдық жүйелік анализдің танымал нотацияларымен салыстырмалы түрде UML тілінің қиындауына алып келгенімен, қиындық үшін төлем болып UML тілінің алғашқы нұсқасының жоғары иірімділігі мен бейнелі мүмкіндіктері саналады. Өз кезегінде, барлық практикалық мәселелерді шешу үшін UML тілін қолдану оның кезекті жетілуі, демек, ОБАЖ объектілі технологиясы мен практикасының кезекті дамуына алып келеді.

6. ОБАЖ жаңа және ең жақсы практикалық жетістіктерін біріктіру. UML тілі үздіксіз түрде жасаушылармен жетілдіріледі, және бұл жұмыстың негізі оның қазіргі модельдік технологиямен кезекті бірігуі болып табылады. Бұл ретте жүйелік модельдеудің әртүрлі әдісі ОБАЖ аясында қолданбалы ұғынуды алады. Келешекте бұл әдістер ОБАЖ ең жақсы практикалық жетістіктерін толық көрсететін қосымша базалық ұғым түрінде UML тілі құрамына қосылады.

Жоғарыда айтылған мәселелерді шешу үшін UML тілі эволюцияның өзіндік қысқа тарихынан өтті. Нәтижесінде UML тілінің сипаттамасы таптаурын емес болды, себебі базалық ұғым семантикасы өзіне тілдің басқа ұғымы мен конструкциясымен қиылысты байланыстың бір қатарын қосады. Осыған байланысты линиялық немесе тізбекті деп аталып UML тілінің негізгі конструкцияларын қарастыру мүмкін емес болды, себебі бірдей ұғымдар әртүрлі диаграмма немесе көрініс жасауда қолданыла алады. Сол сәтте модельдің әрбір көрінісі тілдің базалық ұғымының семантикасына із қалдыратын жеке семантикалық ерекшеліктерге ие болады.

UML тілін суреттеу қиындығын айтқанда, базалық примитивтердің сипаттамасы үшін табиғи тілді пайдалану деңгейі қажеттілігінен шығатын барлық ресми тілдерге қатысты олардың қатал мәселесінің қиындығын атап өтуге болады. Бұл жағдайда табиғи тілге метатіл рөлі беріледі, яғни ресмі тілді сипаттау үшін тіл. Табиғи тіл ресми тіл болмағандықтан, қандай да бір деңгейде ресми тілді сипаттау үшін оны қолдану дәлсіздікке әкеледі. UML тіліне сәйкес логико-лингвистикалық бөлшектердің анализі кірмесе де, бұл ерекшеліктер UML тілін сипаттау құрылымына әсер етті, оның ішінде, оның барлық негізгі ұғымдарының сипаттау стилін жартылай ресми етті.

14.2. UML тілінің жалпы құрылымы

Ең ортақ көзқараста UML тілін сипаттау екі өзара әрекеттесетін бөліктен тұрады: семантика және нотация.

UML тілінің семантикасы UML тілінде абстрактілі синтаксис пен объектілі модельдеу ұғымының семантикасын анықтайтын кейбір метамодельді білдіреді.

UML тілінің нотациясы UML тілінің семантикасын визуалды көрінісі үшін графикалық нотацияны білдіреді.

UML тілінің абстрактілі синтаксисі мен семантикасы UML нотациясының ішкі жиынтығын қолданумен сипатталады. Осыған қосымша ретінде UML нотациясы графикалық нотацияның семантиканың базалық ұғымдарына сәйкестігін сипаттайды. Сөйтіп, функционалды көзқараста бұл екі бөлік бір бірін толықтырады. Бұл ретте UML тілінің семантикасы үш жеке көрініске ие кейбір метамодель негізінде сипатталады: абстрактілі синтаксис, көріністердің дұрыс қойылуы ережесі мен семантика. UML тілінің семантикасын қарастыру базалық ұғым мен олардың кеңею ережесін ұсыну үшін табиғи және ресми тілдерді біріктіретін баяндаудың жартылай ресми стилін білдіреді.

Семантика объектілі модельдің екі түрі үшін анықталады: құрылымдық модель мен әрекет моделі. Статикалық деп те алатын құрылымдық модельдер олардың класстары, интерфейстері, атрибуттары мен қатынастарын қоса отырып, кейбір жүйенің болмыстары немесе құрауыштары құрылымын сипаттайды. Кейде динамикалық деп аталатын әрекет моделі олардың әдісі, өзара әрекеті мен арасындағы ынтымақтастық, сонымен қатар жеке құрауыштар мен жүйе жағдайларының өзгеру процесін қоса отырып, жүйе объектісінің әрекеті немесе қызмет етуін сипаттайды.

Модельдеу мәселесінің осындай кең диапазонын шешу үшін графикалық нотацияның барлық компоненттері үшін айтарлықтай толық семантика жасалды. UML тілі семантикасы талаптары диаграмманың жеке түрлерін жасауда нақтыланады. UML тілі нотациясы диаграмманы жасауда қолданылатын жеке семантикалық элементтерді сипаттауды қосады.

UML тілінің өзін ресми сипаттау төрт деңгейге ие модельдік көріністің кейбір жалпы иерархиялық құрылымына негізделеді: метаметамодель; метамодель; модель; пайдаланушы объектілері.

Метаметамодель деңгейі барлық метамодельдік көрініс үшін бастапқы негіз жасайды. Бұл деңгейдің басты мақсаты метамодельді сипаттау үшін тілді анықтаудан тұрады. Метаметамодель UML тілі моделін абстракцияның ең жоғарғы

деңгейінде анықтайды және оның сипаттамасының ең тұтасы болып табылады. Басқа жағынан, метаметамодель қосымша ұғымдарды қосудың әлеуетті иірімділігіне қол жеткізілетін бірнеше метамодельдерді сипаттай алады. Бұл деңгей ұғымының мысалы ретінде метакласс, метаатрибут, метаоперация саналады.

Метаметамодель семантикасы UML тілінің сипаттамасына кірмейтіндігін атап өткен жөн. Бір жағынан, ол UML тілін зерттеу үшін оңай етеді, себебі ресми тілдер мен ресми логиканың жалпы теориясын білу талап етілмейді. Екінші жағынан, метаметамодельдің болуы UML тіліне қарсы ресми тіл болу үшін қажетті ғылыми дәреже береді.Егер бұл ерекшеліктер көптеген бағдарламашылар үшін қызықсыз болса, онда инструменталды құрал жасаушылары оларды елемей қоюы мүмкін емес.

Метамодель метаметамодельдің үлгісі немесе нақтылауы болады. Бұл деңгейдің басты мақсаты – модельді сипаттау үшін тілді анықтау. Аталмыш деңгей алдындағыға қарағанда конструктивті болып табылады, себебі базалық ұғымның дамыған семантикасына ие. UML тілінің барлық негізгі ұғымдары – ол метамодель деңгейінің ұғымдары. Мұндай ұғымдардың мысалдары: класс, атрибут, операция, құрауыш, ассоциация және т.б.

UML тілі контекстінде *модель* өз жағдайына қатысты оларды нақтылап, метамодель ұғымын ғана қолдануға тиісті жүйенің кез-келген нақты моделі ұғымында метамодель үлгісі болып табылады. Бұл деңгей нақты пәндік сала жөнінде ақпаратты сипаттау үшін қызмет етеді. Алайда егер модельді жасау үшін UML тілі ұғымы қолданылса, онда метамодель деңгейінің UML тілінің базалық ұғымымен модель деңгейі ұғымымен толық сәйкестігі керек. Модель деңгейі ұғымының мысалы болып жобаланатын мәліметтер базасының өңірі аттары қызмет ету мүмкін – қызметкер аты-жөні, жасы, лауазымы, мекенжайы, телефоны. Бұл ретте ұғым мәліметтері сәйкес ақпараттық атрибут аттары ретнде ғана қолданылады.

Модель ұғымдарын нақтылау *пайдаланушы объектісі* деңгейінде жасалады.Бұл контекстте объект модель үлгісі болады, себебі модель ұғымына расында сәйкес келуіне қатысты нақты ақпаратқа ие. Объект мысалы ретінде жобаланатын мәліметтер базасындағы келесі жазба оған дәлел бола алады: «Илья Петров, жасы 18-де, бағдарламашы, Пионерская к.,5 үй, 1к., 23 п., тел. 123-45-67».

UML семантикасын сипаттау метаметамодель деңгейінің мысалы немесе жеке жағдайын ғана көрсететін метамодель деңгейінің ғана базалық ұғымдарын қарастыруды білдіреді. UML

метамоделі өз кезегінде физикалық немесе іске асыру моделіне қарағанда логикалық модель болып табылады. Логикалық модель ерекшелігі – ол модельдің нақты физикалық іске асуын жіберіп, декларативті немесе концептуалды семантикаға назарын аударуында. Бұл ретте осы логикалық метамодельді қолданатын жеке іске асулар оның семантикасымен келістірілуі керек, сонымен қатар жеке логикалық модельдердің импорты мен экспортын қолдау керек.

Сол уақытта логикалық метамодель сәйкес инструменталды құралдардың өнімділігі мен сенімділігінің талапты деңгейін қамтамасыз ету үшін әртүрлі әдістермен іске асырылу керек. Семантика деңгейінде оның тиімді кезекті іске асырылуы үшін міндетті талаптары жоқ логикалық модельдің кемшілігі осында. Алайда метамодельдің іске асырудың нақты модельдерімен сәйкестігі UML тілін қолдауды қамтамасыз ететін бағдарламалық құрал жасаушылары үшін міндетті болады.

14.3. UML тіліндегі пакеттер жөнінде жалпы мәліметтер

UML тілінің метамоделі тілдің жаңа нұсқасы шығуымен саны артатын 90 метакласқа жуық, 100 метассоциациядан астам және 50 стереотипке жақын қиын құрылымға ие. UML тілінің бұл қиындығына төтеп беру үшін, оның барлық элементтері логикалық пакеттерге жиналған. Сондықтан метамодельдік деңгейде UML тілін қарастыру оның үш жалпы логикалық блоктары немесе пакеттерінен тұрмайды: «Негізгі элементтер», «Әрекет элементтері», «Жалпы механизмдер».

Тілдің жалпы сипаттамасы контекстінде пакеттер жайында айтқанда,біз, іс барысында, UML тілінің графикалық нотациясын қарастырамыз. UML тілін сипаттау үшін тілдің өзінің құралдары қолданылады және бұл құралдың бір түрі ретінде пакет саналады. Жалпы жағдайда пакет модель элементтерін топтау үшін қызмет етеді. Бұл ретте бір пакетке жатқызылған негізсіз болмыс бола алатын модель элементі біртұтас болады.

Модельдің басқа элементтері сияқты пакеттер басқа пакеттерге салына алады. UML тілінің маңызды ерекшелігі ретінде UML моделі элементтерінің барлық түрі пакеттерге ұйымдастырылуы мүмкіндігі саналады.

Пакет – UML тілінде модельдер элементтерін ұйымдастырудың негізгі тәсілі. Әрбір пакет өз элементтеріне ие, яғни оған қосылған элементтерге ие. Пакеттің сәйкес элементтері жайында олар пакетке тиесілі немесе оған кіреді

деп айтады. Бұл ретте әрбір элемент бір пакетке ғана тиесілі болады. Өз кезегінде, бір пакет басқа пакетке салына алады. Бұл жағдайда біріншісі ішкі пакет деп аталады, себебі ішкі пакеттің барлық элементтері жалпы пакетке тиесілі. Сөйтіп, модель элементтері үшін иерархия көрсететін пакеттердің салыну қатынасы беріледі.

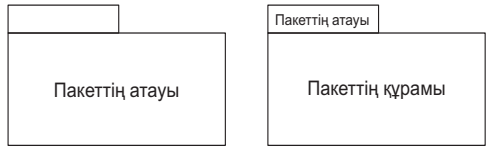
Пакет-ішкі пакет қатынастарын жиынтық – ішкі жиынтықтың жалпы қатынасына ұқсатқан жөн. Пакетті жиынтықтың жеке жағдайы ретінде қарастыруға болғандықтан, мұндай түсіндіру пакеттер арасында сәйкес қатынасты ұсыну үшін графикалық құралды қолдануға көмектеседі.

Иерархияның графикалық ұсынысы үшін ағаштар деп аталатын арнайы түрдегі графтар қолданыла алатындығы мәлім. Алайда UML тілінде бұл графикалық белгілер бастаушы жасаушылар үшін айтарлықтай қиындық туғызуы мүмкін жалпытеориялық ұғымдармен сәйкес ассоциациясы болатындай модификацияланған. Дегенмен, UML тілінің арнайы конструкциясын жүйелі аналитик ойының стилін жасайтын жиынтық теориясы мен жүйелік модельдеудің сәйкес ұғымдарымен ұқсатуды білген өте маңызды. Әйтпесе, өкінішті қателер пәндік саланың концептуализациясының алғашқы сатысында ғана емес, сонымен қатар жүйенің әртүрлі көрінісін жасау процесінде туындайды.

UML тілінде пакеттер визуализациясы үшін арнайы белгі немесе графикалық нотация жасалған. Пакеттердің графикалық көрінісі үшін диаграммаларды арнайы графикалық белгі қолданылады – үлкен тікбұрыштың жоғары жағының сол бөлігіне жалғанған кішкене тікбұрышы бар үлкен тікбұрыш (14.2 сур.). Визуалды түрде пакет белгісі танымал графикалық интерфейстегі папка пиктограммасын еске салады. Үлкен тікбұрыш ішінде аталмыш пакетке қатысты ақпарат жазылуы мүмкін. Егер мұндай ақпарат болмаса, онда үлкен тікбұрыш ішіне қарастырылатын модель шегінде бірегей болатын пакет аты жазылады (14.2, а сур.). Мұндай ақпараттың болуында пакет аты жоғарыдағы кішкентай тікбұрышта жазылады (14.2, б сур.).

Пакет атын айтқанда UML тіліндегі аттар жөніндегі жалпы келісімге тоқталып өту керек. Бұл жағдайда пакет аты ретінде кез-келген әріп, сан немесе кейбір арнайы белгі санына ие мәтін жолы (немесе бірнеше жолдар) болуы мүмкін.

Пакеттерді сәйкестендіру мақсатымен олардың аттары ретінде бір немесе бірнеше зат есімдерді қолдану керек, мысалы, «бақылаушы», «графикалық интерфейс», « мәліметтер



14.2 сур. UML-тіліндегі пакеттердің графиктік көрінісі

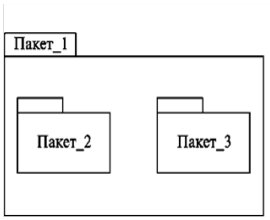
а) осы пакетке қатысты мәлімет жоқ кезде б) мәлімет бар кезде

енгізу формасы». Пакет аты алдында кейбір кілтті сөздерге ие мәтін жолы енгізілуі мүмкін. Мұндай кілтті сөздер ретінде стереотиптер деп аталатын UML тілінде алдын ала анықталған сөздер саналады. Пакеттер үшін мұндай стереотиптер болып facade (фасад), framework (тірек), stub (бітеуіш) және topLevel (жоғарғы деңгей) табылады. Пакеттің құрамы ретінде олардың жеке элементтерінің аттары мен олардың қасиеттері шыға алады.

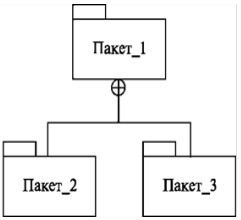
Өздігінен пакеттер шекті қолданыс таба алады, себебі олардың құрамына кіретін модель элементі жөнінде ғана ақпаратқа ие. Жеке пакеттер арасында орын ала алатын қатынасты графикалық елестету де өте маңызды. Графтар теориясында да, UML тіліндегі қатынастарды көру үшін сырты мағыналық құрамға ие линия бөліктері қолданылады.

Пакеттер арасында қатынастың бір типі ретінде пакеттердің бір-біріне салынуы немесе қосылуы қатынасы саналады. Бір жағынан, UML тілінде бұл қатынас линияны қолданусыз көрсетілуі мүмкін,яғни бір тікбұрыш-пакеттің екінші тікбұрыш пакетке салынуы (14.3 сур). Сөйтіп, бұл жағдайда Пакет_1 атты пакет екі ішкі пакетке ие: Пакет_2 және Пакет_3

Екінші жағынан, бұл қатынас линия бөліктері көмегімен



14.3 сур.Пакеттердің бір-біріне салынуының графиктік көрінісі



14.4 сур. Қосу қатынасын көру көмегімен пакеттерді бір-бірі не салудың графиктік көрінісі

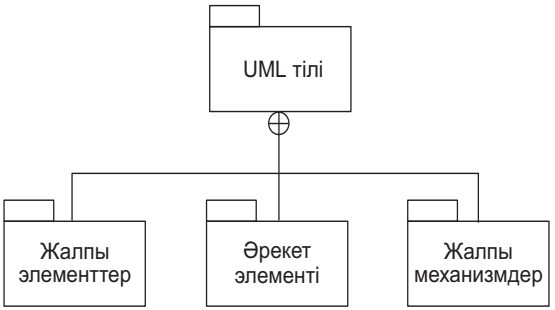
ағаштың графиктік көрінісіне сәйкес көрсетілуі мүмкін (14.4 сур). Бұл жағдайда үлкен пакет (метапакет немесе контейнер) суреттің жоғарғы бөлігіне салынып, оның ішкі пакеттері төменірек деңгейде болады. Метапакет ішкі пакеттермен тұтас линия бойынша жалғанады, метапакетке келетін аяқ жағында арнайы белгі көрсетіледі – шеңбердің ішіндегі «плюс» белгісі. Бұл белгі ішкі пакеттер «меншік» болады немесе контейнер бөлігі және олардан басқа контейнерде ешқандай ішкі пакет жоқ екендігін білдіреді.

Графиктік диаграммаларда пакеттер арасында қатынастардың басқа да типтері көрсетілуі мүмкін.

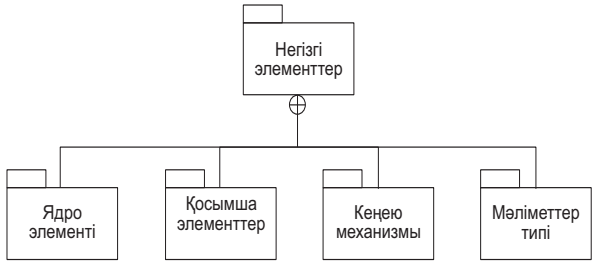
14.4. UML тілінің метамоделінің негізгі пакеттері

14.3 тармақшада көрсетілгендей, метамодельдік деңгейде UML тілін көрсетудің негізі оның үш логикалық блоктарын немесе пакеттерін сипаттау болып табылады: «Негізгі элементтер», «Әрекет элементтері» және «Жалпы механизмдер» (14.5 сур.).

Бұл пакеттер, өз кезегінде, жеке ішкі пакеттерге бөлінеді. Мысалы, «Негізгі элементтер» пакеті «Ядро элементтері», «Қосымша элементтер», «Кеңею механизмі» мен «Мәліметтер типі» (14.6 сур.) сияқты ішкі пакеттерден тұрады. Бұл ретте «Ядро элементтері» пакеті базалық мәліметтерді және метамодель құрылымына метакласстар, метаассоциациялар мен метаатрибут сияқты тілдің негізгі ұғымдарын қосу принциптерін сипаттайды.



14.5 сур.UML тілі метамоделінің негізгі пакеттері



14.6 сур. UML тілінің «Негізгі элементтер» пакетінің ішкі пакеттері

«Қосымша элементтер» пакеті тәуелділіктер, шаблондар, физикалық құрылымдар мен көрініс элементтерін сипаттау үшін базалық элементтерді кеңейтетін қосымша конструкцияларды анықтайды. «Кеңею механизмдері» пакеті модельдердің базалық элементтерінің нақтылау мен семантикасын кеңейту ережелерін береді. «Мәліметтер типтері» пакеті UML тілі үшін мәліметтердің негізгі құрылымдарын анықтайды.

«Негізгі элементтер» пакеті. Төменде «Негізгі элементтер» пакеті құрамына кіретін аталып өткен әрбір ішкі пакеттердің элементінің қысқаша сипаттамасы беріледі.

***«Ядро элементі» пакеті* UML тілінің «Негізгі элементтер» пакетіне кіретін барлық ішкі пакеттердің ішінде ең фундаментальдысы болып табылады. Бұл пакет объектілі модельдерді жасауға қажетті негізгі абстрактілі және нақты құрауыштарды анықтайды. Бұл ретте метамодельдің абстрактілі құрауыштары үлгі немесе мысалдарға ие болмайды және модельдің басқа құрауыштарын анықтау үшін ғана қолданылады. Метамодельдің нақты құрауыштары үлгілерге ие және объектілі модель жасайтын тұлғаларды ұсыну ерекшеліктерін көрсетеді.**

Білімді көрсетудің дамыған тілдері, соның ішінде UML тіліне тиесілі көрнекті мүмкіндіктерінің бір мәнді еместігін атап өткен жөн. Әңгіме бірдей модельденетін болмыс немесе жүйе UML тілінің құралдарымен әртүрі ұсынылуы мүмкін болуында. Бұл ретте әртүрлі жасаушылар өзінің көрінісі формасымен ғана емес, сонымен қатар, модельде қолданылатын құрауыштар құрамымен ерекшеленетін бірдей жүйенің объектілі моделін жасай алады.

«Ядро элементтері» пакеті бастапқы метамодельді сипаттау үшін қажетті базалық конструкцияларды сипаттайды және метакласстар, метаассоциациялар мен метаатрибуттар сияқты тілдің қосымша конструкцияларын қосу үшін сәулетті «қаңқа»

анықтайды. «Ядро элементтері» пакеті UML тілінің барлық қалған бөлігін анықтау үшін жеткілікті семантикаға ие болғанымен, UML метамоделі болып табылмайды.

Бұл пакетке UML тілінің негізгі метакласстары кіреді: класс (Class); атрибут (Attribute); ассоциация (Association); ассоциация-класс (AssociationClass); ассоциацияның аяғы (AssociationEnd); әрекет қасиеті (BehavioralFeature); классификатор шектеу (Constraint); мәліметтер типі (DataType); тәуелділік (Dependency); элемент (Element); элементке құқық (ElementOwnership); қасиет (Feature); жалпылау (Generalization); жалпылау қатынастарының элементі (GeneralizableElement); интерфейс (Interface); әдіс (Method); модель элементі (ModelElement); аттар кеңістігі (Namespace); операция (Operation), параметр (Parameter); құрылымдық қасиет (StructuralFeature); көріністі жасау ережесі (Well-formedness rules).

«Қосымша элементтер» пакеті «Ядро элементтері» пакетін кеңейтетін UML тілінің қосымша конструкциясын сипаттайды. Қосымша элементтер тәуелділіктер, шаблондар, физикалық құрылымдар мен көрініс элементтері үшін ұғымдық базисті қамтамасыз етеді.

Бұл пакетке келесі метакласстар кіреді: байланыстыру (Binding); комментарий (Comment); құраушы (Component); түйін (Node); презентация (Presentation); нақтылау
тәуелділіктер тізбегі (Trace); тұтыну (Usage); көрініс элементі (ViewElement); тәуелділік (Dependency); модель элементі (ModelElement); көріністі жасау ережесі (Well-formedness rules). Соңғы үш метакласстар «Ядро элементтері» пакетінен алынған және қалғандарын сипаттау үшін қолданылады.

Бұл пакет UML бастапқы нұсқаларында өзіндік мәнге ие болғанымен, соңғы нұсқа жобаларында оның элементтері «Ядро элементтері» пакетімен бірікті. Оған себеп ретінде әрбір элементтің бір пакетке кіруінің қатаң талабы болды.

«Кеңею механизмдері» пакеті элементтер моделіне нақтыланған семантикамен қосылу тәртібін, сонымен қатар модельденетін жүйе сипаттамасын нақты көрсету үшін UML тілдерінің жеке құрауыштарының модификациясын сипаттайды.Кеңею механизмі стереотиптер, шектеулер мен белгіленген мәндер үшін семантиканы анықтайды. UML типикалық бағдарламалық жүйені модельдеу үшін көптеген ұғым мен нотацияға ие болғанымен, жасаушы шынайы түрде модельге UML тілінде анықталмаған қосымша қасиеттер немесе нотациялар қосу қажеттілігіне тап болуы мүмкін. Бұл ретте жасаушылар модельге графиктік ақпаратты қосу қажеттілігімен, мысалы, қосымша белгі мен безендіру сияқтылар, жиі тап болады.

Бұл мақсат үшін UML тілінің метамоделінде сипатталғаннан ерекшеленетін семантика, нотация мен қасиеттер элементтерімен модельдің жаңа элементтерін анықтау үшін бірігіп немесе жеке қолданыла алатын кеңеюдің үш механизмі UML тілінде қарастырылады. Мұндай механизмдер шектеу (Constraint), стереотип (Stereotype) және белгіленген мән (Tagged Value) деп аталады.

UML тілінің кеңею механизмдері келесі міндеттерді орындау үшін қажет:

UML тілінде модельдерді жасауда қолда бар модельдік элементтерді нақтылау:

жеткілікті түрде қызық болмайтын немесе UML метамоделінің элементі ретінде анықтау үшін қиын болатын UML тілінің сипаттамасында стандарттық құрауыштарды анықтау:

модельденетін процеске немесе бағдарламалық кодты жүзеге асыру тіліне тәуелді UML тілін кеңейтуді анықтау;

негізсіз семантикалық немесе семантикалық емес ақпаратты модель элементіне қосу.

Кеңеюдің кіріктірілген механизмдерінің ішінде ең маңыздылары «стереотип» ұғымына негізделеді. Стереотиптер объектілі модель деңгейінде модельдік элементтерді классификациялау әдісі мен UML тіліне жаңа атрибуттар мен семантикамен «виртуалды» метакласстар қосуды қамтамасыз етеді. Кеңеюдің басқа кіріктірілген механизмдері белгіленген мәндер мен шектеулерге ие «қасиеттер тізімі» ұғымына негізделеді.

Бұл механизмдер пайдаланушыға қосымша қасиеттер мен модельдің жеке элементтеріне семантика қосу мүмкіндіктерін береді.

«Мәліметтер типі» пакеті UML тілінде қолданылатын мәліметтердің әртүрлі типтерін сипаттайды. Бұл пакет басқа пакеттермен салыстырғанда оңай құрылым мен сипаттамаға ие, себебі сәйкес ұғымдардың семантикасы белгілі деп есептеледі.

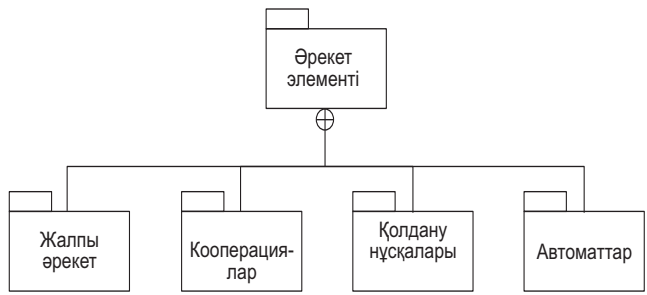
UML метамоделінде мәліметтер типтері атрибутты класстар типін жариялау үшін қолданылады. Олар диаграммаларда мәтін жолы түрінде жазылады және «мәліметтер типі» деген жеке белгіге ие болмайды. Осының арқасында ақпаратты жоғалтусыз диаграмма өлшемін кішірейту жүзеге асырылады.Алайда кейбір мәліметтер типі үшін бірдей жазбалардың әрқайсысы модельдегі мәліметтердің бірдей типіне сәйкес болу керек. Бұл ретте UML тілін сипаттауда қолданылатын мәліметтер типі жасаушы өз моделі үшін UML тілінде анықтайтын мәліметтер типінен ерекшеленуі мүмкін. Соңғы жағдайда мәліметтер типі жеке жағдай немесе метамодельде анықталған «мәліметтер типі» метаклассының үлгісі болып есептеледі.

Мәліметтер типін көрсетуде айтып шығу деп аталатын бейресми конструкция қолданылады. Әңгіме реттің кейбір қатынасымен бөлінетін атрибуттың рұқсат етілетін мәнінің жиынтығы жөнінде. Бұл ретте мәндердің реттілігі тізімнің бірінші және соңғы элементтерін анық көрсетеді немесе мәліметтердің типтері жағдайында анық емес көрсетіледі, мысалы, натурал сандардың жиынтығы үшін. «Мәліметтер типі» пакетінде атрибуттардың ұйғарынды мәнін дұрыс көрсету үшін айтып шығу сипаттамасының тәсілі анықталады.

UML тілінде мәліметтердің әртүрлі типін анықтау үшін оңай конструкциялар қолданылады: толық сан (Integer), жол (String), ат (Name), булев (Boolean), уақыт (Time), қысқалық (Multiplicity), көрініс типі (Visibility Kind), қысқалық диапазоны (MultiplicityRange) және қиындар көрсетіледі: көрініс (Expression), бульдік көрініс (BooleanExpression), агрегациялау типі (AggregationKind), өзгеру типі (ChangeableKind), геометрия (Geometry), көрсету (Mapping), көрініс-процедура (Procedure Expression), псевдожағдай типі (Pseudostate- Kind), уақыт көрінісі (TimeExpression), үздіксіз (Uninterpreted).

«Әрекет элементтері» пакеті. Бұл пакет UML тілінің өзіндік құрауышы болып табылады, және атауынан байқалып тұрғандай, UML нотациясында әрекет динамикасын сипаттайды. «Әрекет элементі» пакеті төрт ішкі пакеттерден тұрады: «Жалпы әрекет», «Кооперация», «Қолдану нұсқалары» және «Автоматтар» (14.7 сур.). Төменде әрбір ішкі пакеттің сипаттамасы берілген.

«*Жалпы әрекет» пакеті* барлық пакеттердің ішінде ең фундаменталдысы болып табылады және әрекеттің барлық элементтері үшін қажетті ядроның базалық ұғымдарын анықтайды. Бұл пакетте әрекет элементінің басқа ішкі



14.7 сур. UML тілінің «Әрекет элементтері» пакетінің ішкі пакеттері

пакеттеріне қосылған динамикалық элементтер үшін семантика сипатталады. «Жалпы әрекет» пакеті келесі элементтер кіреді:объект (Object), әрекет Action), әрекеттер тізбектігі (Action Sequence), аргумент (Argument), үлгі (Instance), алып тастау (Exception), байланыс (Link), сигнал (Signal), мәліметтер мәні (Data Value), атрибуттар байланысы (Attribute Link), шақыру әрекеті (Call Action), жасау әрекеті (Create Action), жою әрекеті (DestroyAction).

Жоғарыда аталған элементтердің ішінен ең маңыздысы UML тілінде жеке үлгі немесе осы объектті жасайтын класспен толық анықталатын класс, құрылым немесе әрекет ретінде ұғынылатын объект саналады. Бірдей класспен пайда болған барлық объектілер бірдей құрылым мен әрекетке ие, алайда болардың әрқайсысы атрибуттардың жеке байланыс жиынтығына ие бола алады. Бұл ретте атрибуттың әрбір байланысы кейбір үлгіге жатқызылады, әдетте ол мәліметтер мәні.Бұл жиынтық классты сипаттауда жеке атрибутты сипаттауға байланысты модификациялануы мүмкін.

UML тілінде әрекет ретінде олардың мәндері бойынша операцияны жасау нәтижесінде объектілердің атрибуттарының өзгеру процесі ғана емес, сонымен қатар объектілердің жасалуы мен жойылуы сияқты процедуралар ұғынылады. Бұл ретте олардың әрекетін анықтайтын объектілердің өзара әрекеттесуі динамикасы «сигнал» және «әрекет» сияқты арнайы ұғымдар көмегімен сипатталады.

«Кооперациялар» пакеті жеке міндетті орындау үшін оларды қолдануда модель элементтерінің әрекетінің контекстін сипаттайды. Мұндай келесі сұраққа жауап үшін қажетті ұғымдар семантикасы беріледі: «Модельдің әртүрлі элементтері құрылым көзқарасында өзара қалай әрекеттеседі?». Бұл пакет «Негізгі элементтер» мен «Жалпы әрекет» пакеттерінде анықталған конструкцияларды қолданады.

Сондай-ақ, «Кооперация» пакетіне келесі элементтер кіреді: кооперация (Collaboration), өзара әрекеттесу (Interaction), хабар (Message), ассоциация рөлі (AssociationRole), классификатор рөлі ассоциация соңының рөлі (AssociationEndRole). Пакет атауынан байқауға болатындай, оның элементтері кооперация диаграммасын жасауда қолданылады. «Кооперация» ұғымы классификатор және ассоциация көзқарастарында модель элементтерінің өзара әрекеттесуін ұсыну үшін маңызды мәнге ие.

«Қолдану нұсқалары пакеті UML тілінде актерлер және қолдану нұсқалары деп аталатын оған арнайы элементтерді қосуда модель әрекетін сипаттайды. Бұл ұғымдар жүйе сияқты модельденетін болмыстың қызмет етуін анықтау үшін әрекет етеді. Бұл пакет элементтерінің ерекшелігі – олар оның ішкі

құрылымын сипаттамай-ақ әрекетінің алғашқы анықтауы үшін қолданылады.

UML тілінде жобаланатын жүйе мен осы элементтер үшін қолданылатын айтарлықтай бай семантикамен оның ішкі құрауыштарын құрылымдауға бастапқы талаптарды концепциялау құралы қиын жүйенің модельдерін жасау үшін маңызға ие. Расында, дәстүрлі модельдердің шектілігі бір уақытта жүйенің статикалық немесе динамикалық қасиеті мен оның әрекетінің динамикасын сипаттауға жол беретінінде. Мәселелерді бірігіп шешу талпынысы мағынасы бойынша жақын жүйелік ұғымдарды көрсету үшін біріккен белгілеудің болмауына тап болады. UML тілі мұндай модельдерді жасауда қажетті базалық ұғымдарды жақсы көрсетеді.Сонымен бірге, егер бұл ұғымдар бір нақты жобаны жасау үшін жеткіліксіз болса,онда жасаушының өзі базалық ұғымдарды кеңейтіп, UML тілінің метамоделімен келісілген модельге жеке конструкциялар қоса алады.

«Қолдану нұсқалары» пакетіне жоғарыда аталған актер (Actor) және қолдану нұсқасы (UseCase) элементтерінен басқа да келесі элементтер кіреді: кеңею (Extension), кеңею нүктесі (ExtensionPoint), қосу (Include) және қолдану нұсқасының үлгісі (UseCaselnstance).

«Автоматтар» пакеті жағдайлардың соңғы жиынтығы үшін өткелдер жүйесін қолданумен модельдерді жасауда элементтер әрекетін сипаттайды. Онда жағдайлар мен өткелдердің соңғы санымен дискретті кеңістік түрінде модель әрекетін ұсыну үшін қажетті ұғымдар жиынтығы анықталған.

UML тілінде қолданылатын автомат формализмі өзінің объектілі ориентациясының автоматтар теориясының формализмінен ерекшеленеді. Автоматтар UML тілінің әртүрлі элементтерінің әрекетін модельдеудің негізгі құралы болып табылады. Мысалы, автоматтар класстардың үлгісі сияқты жеке болмыстардың әрекетін модельдеу үшін, сонымен қатар кооперациялар сияқты болмыстар арасында өзара әрекеттесуді сипаттау үшін қолданылуы мүмкін. Автоматтар формализмі автоматтың жеке жағдайы болып табылатын қызмет графтары үшін семантикалық базисті қосымша қамтамасыз етеді.

«Автоматтар» пакетіне келесі элементтер кіреді:жағдай (State), өткел (Transition), оқиға (Event), автомат (StateMa- chine), жай жағдай (SimpleState), құрамдас жағдай (CompositeState), псевдожағдай (PseudoState), соңғы жайдай (FinalState) және тағы басқалар.

Жүйені динамикалық қасиеттерін модельдеуде ең маңызды болып «жағдай» ұғымы саналады. UML тілінде жағдай

ретінде кейбір инвариантты жағдайды жасау болатын (әдетте анық емес) жағдай немесе процесті модельдеу үшін қолданылатын абстрактілі метакласс ұғынылады. Мұндай жағдайдың себебі болып объекттің кейбір сыртқы оқиғаны орындалуын күту жағдайы табылады, мысалы, басқаруды сұрау немесе жіберу. Екінші жағынан, жағдай кейбір операцияны орындау процесі сияқты динамикалық жағдайды модельдеу үшін қолданылуы мүмкін. Бұл жағдайда операцияны орындаудың басталу сәті болып объекттің сәйкес жағдайға өтуі табылады.

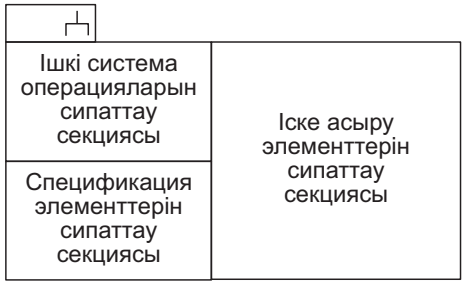
14.8 сур. "Жалпы механизмдер" пакетінің құрамы

«Жалпы механизмдер» пакеті. Бұл пакетте UML тілінің барлық модельдеріне қатысты жалпы механизмдер анықталған. Пакет «Модельдерді басқару» (14.8 сур)атты жалғыз ішкі пакеттен тұрады, ол модель, пакет және ішкі жүйеге элементтерді ұйымдастыру тәсілін сипаттау үшін әрекет етеді.

«Модельдері басқару» пакеті (Model Management) барлық модельдік көріністерді жасау үшін қажетті UML тілінің базалық элементтерін сипаттайды. Осыда модель (Model), пакет (Package) және ішкі жүйе (Subsystem) семантикасы анықталады. Бұл элементтер модельдің басқа элементтерін топтау үшін өзіндік контейнер болады.

Пакет UML тілінде метакласс болып табылады және жоғарыда аталғандай, басқа пакеттер, классификаторлар мен ассоциациялар сияқты модельдің басқа элементтерін ұйымдастыру үшін арналған. Пакет пакеттің өзінде модель элементтері арасында шектеулер мен тәуелділіктерге ие болуы мүмкін. Ол пакет сыртында оның ешқандай элементі қолданылмау керек дегенді білдіреді, егер импорт немесе пакеттің жеке элементіне рұқсат сияқты қосымша нұсқаулары болмаса. Өзінің барлық құрамымен пакеттер модельдің барлық элементтерінің аттарын қолданудың жалғыздығы берілетін аттар кеңістігінде анықталған. Басқаша сөзбен айтқанда, модельдің әрбір элементінің аты жалғыз немесе бірегей болу керек, модельдің элементі болатын аттардың кейбір кеңістігінде аттардың ортақ кеңістігіне қосылуы мүмкін.

Модель пакеттің ішкі классы болады және анықталған мақсат үшін арналған физикалық жүйе абстракциясын ұсынады. Осы мақсат модельге қосылуы керек құрауыштарды және қарастыру міндетті болмайтындарды алдын ала анықтайды.Басқаша сөзбен айтқанда, модель қойылған мақсатқа қол жеткізуде ықпал ететін физикалық жүйенің релевантты аспектілерін көрсетеді.



14.9 сур. UML тіліндегі ішкі жүйенің графиктық көрінісі

Қолданбалы міндеттерде мақсат жүйеге бастапқы талаптар формасында беріледі, олар, өз кезегінде, UML тілінде жүйені қолдану нұсқасы түрінде жазылады.

UML тілінде бірдей физикалық жүйе үшін әртүрлі көзқараспен әрқайсысы жүйені сипаттайтын әртүрлі модельдер анықталуы мүмкін. Мұндай модельдердің мысалы ретінде логикалық модель, жобалау моделі, қолдану нұсқаларының моделі және т.б. болады. Бұл ретте әрбір осындай модель физикалық жүйеге өзіндік көзқараска ие және абстракцияның өзіндік деңгейі бар. Өз жағынан, пакет бірдей жүйенің бірнеше әртүрлі модельдерін қоса алады және UML тілінде модельдерді жасаудың маңызды механизмдерінің бірі осыдан тұрады.

Ішкі жүйеде физикалық жүйенің кейбір оңай әрекетін сипаттайтын модель элементтерінің тобы бар. UML метамоделінде ішкі жүйе пакеттің де, классификатордың да ішкі классы болады. Ішкі жүйе элементтері екі бөлікке бөлінеді: әрекет сипаттамасы оны іске асыру.

Ішкі жүйені графиктік көрсету үшін арнайы белгі қолданылады – пакет жағдайындағыдай тікбұрыш, бірақ үш секцияға қосымша бөлінген (14.9 сур.). Бұл ретте жоғарғы кішкентай тікбұрышта түрі шанышқыға ұқсас, ішкі жүйені көрсететін белгі салынады.

Ішкі жүйе аты міндетті емес кілтті сөзбен немесе стереотиппен бірге үлкен тікбұрыш ішіне жазылады. Алайда үлкен тікбұрыш ішінде мәтін жолдары болса, ішкі жүйе аты «шанышқы» белгісінің жанына жазылуы мүмкін.

Ішкі жүйе операциялары сол жақтағы жоғары секцияда жазылады, төменірек сипаттама элементтері көрсетілерді, ал вертикалды линияның оң жағында – іске асыру элементтері. Бұл ретте соңғы екі секция сәйкес белгілермен белгіленеді: «Сипаттама элементтері» мен «Іске асыру элементтері». Операция секциясы белгіленбейді.

Егер ішкі жүйеде мұндай секциялар болмаса, онда олар сызбада мүлде көрсетілмейді.

Бақылау сұрақтары

1. UML тілі дегеніміз не?
2. UML тілінің негізіне модельдеудің қандай принциптері қойылған?
3. UML тілінің негізгі міндеттерін атап өтіңіз.
4. UML тілінің бастапқы ұғымдарының кеңеюі мен мамандануы мүмкіндіктері дегеніміз не?
5. UML тілінің бағдарламаудың басқа тілдерінен тәуелсіздігі қажеттілігі қалай түсіндіріледі?
6. UML тілін сипаттау қандай бөліктерден тұрады?
7. UML тілінің иерархиялық құрылымы құрамына қандай деңгейлер кіреді?
8. Метамодель, модель мен объект арасындағы өзара байланысты түсіндіріңіз.
9. UML тілінде қолданылатын «пакет» ұғымын түсіндіріңіз.
10. Пакеттің графиктік белгісін салып, түсіндіріңіз.
11. Пакеттердің салымы нені білдіреді?
12. UML тілінің метамоделінің негізгі пакеттерін атаңыз.
13. «Негізгі элементтер» пакетіне қандай пакеттер кіреді?
14. «Қосымша пакеттер» пакетінің мақсаты неде? Ол қазіргі таңда қолданылады ма жоқ па?
15. «Ядро элементтері» пакетінің мақсаты неде?
16. UML тілінде қандай мәліметтер типі қолданылуы мүмкін?
17. «Әрекет элементтері» пакетіне қандай пакеттер кіреді?
18. «Жалпы әрекет» пакетінің мақсаты неде?
19. «Автоматтар» пакеті не үшін керек?
20. «Жалпы механизмдер» пакетіне қандай пакеттер кіреді?
21. «Модельдерді басқару» пакетінің мақсаты неде?

ӘДЕБИЕТТЕР ТІЗІМІ

1. *Буч Г.* UML. Руководство пользователя / Г. Буч, Дж. Рамбо, А.Дже- кобсон. — М. : ДМК-Пресс, 2000.

2. *Вендров А. М.* Проектирование программного обеспечения экономических информационных систем / А. М. Вендров. — М. : Финансы и статистика, 2000.

3. *Канер С.* Тестирование программного обеспечения / С.Канер, Д. Фолк, Е. Кек Нгуен; ағылш. тілінен аударма — Киев: ДиаСофт, 2000.

4. *Леоненков А.В.* Самоучитель UML / А.В.Леоненков. — СПб. : БХВ- Петербург, 2001.

5. *Скотт К.* UML. Основные концепции / К.Скотт. — М.: «Вильямс» баспа үйі, 2002.

6. *Соммервилл И.* Инженерия программного обеспечения / И. Соммер- вилл. — М. : СПб. : Киев: «Вильямс» баспа үйі, 2002.

7. *Фридман А.Л.* Основы объектно-ориентированной разработки программных систем / А. Л. Фридман. — М. : Финансы и статистика, 2000.

8. *Boehm B.* Software Engineering Economics / B. Boehm. — London: Prentice-Hall International Inc., 1981.

9. *Booch G.* Software Metrics: Establishing a Company-Wide Program / G. Booch, P. Caswell, L.Deborah. — London: Prentice-Hall International Inc., 1987.

10. *Humphrey W.* Managing the Software Process / W.Humphrey, S.Watts. — Santa Clara: Addison-Wesley Publishing Company Inc., 1990.

Myers G. The Art of Software Testing / G. Myers, J. Glenford. — New Jersey: A Wiley-Interscience Publication, 1979.

МАЗМҰНЫ