Қазақстан Республикасы Білім және ғылым министірлігі

А.Байтұрсынов атындағы Қостанай мемлекеттік университеті

Бағдарламалық қамтамасыз ету кафедрасы

Г.Жармагамбетова

БАҒДАРЛАМАЛАРДЫ ӨҢДЕУДІҢ ҚҰРАЛ - ЖАБДЫҚТАРЫ

Оқу құралы

Қостанай, 2018

УДК 004.4(075)
ББК 32.973.26-018.1я73
Ж 31

Құрастырушы:
Жармагамбетова Гульшат Омаровна, жаратылыс ғылымдарының магистрі, бағдарламалық қамтамасыз ету кафедрасының аға оқытушысы

Пікір берушілер:
Жунусов Куат Муратович - экономика ғылымдарының кандидаты, М.Дулатов атындағы Қостанай инженерлік-экономикалық университеті ақпараттық технологиялар және автоматика кафедрасының меңгерушісі
Исмаилов Арман Оразалиевич - техника ғылымдарының кандидаты, А.Байтұрсынов атындағы Қостанай мемлекеттік университеті оқу үрдісін жоспарлау және ұйымдастыру басқармасының басшысы
Жикеев Азамат Айтпаевич - техника ғылымдарының кандидаты, А.Байтұрсынов атындағы Қостанай мемлекеттік университеті ақпараттық жүйелер кафедрасының доценті, тіркеу бөлімінің басшысы

Жармагамбетова Г.О.
Ж31 Бағдарламаларды өңдеудің құрал-жабдықтары: оқу құралы. - Қостанай: А.Байтұрсынов атындағы Қостанай мемлекеттік университеті, 2018. – 101 б.

ISBN 978-601-7985-10-3

Оқу құралында бағдарламаларды өңдеудің құрал-жабдықтарының негізгі ережелері кірістірілген, сонымен қатар пәннің әр тақырыптарына бақылау сұрақтары мен тест тапсырмалары ұсынылған.
Оқу құралы 5В070400 – Есептеу техникасы және бағдарламалық қамтамасыз ету мамандығының студенттеріне арналған. Студенттерге пән бойынша толық мәліметтер берілген.

УДК 004.4(075)
ББК 32.973.26-018.1я73

А.Байтұрсынов атындағы Қостанай мемлекеттік университетінің Оқу-әдістемелік кеңесімен бекітілген, хаттама № __. __.2018 ж

ISBN 978-601-7985-10-3

Мазмұны

Кіріспе

«Бағдарламаларды өңдеудің құрал - жабдықтары» пәнін оқытудың мақсаты: бағдарламалық аспаптарды атап өту және оның классификациясын үйрету, аспаптық бағддарламалық қамтаманың құралдары және әдістерінің құрамы мен қолдану аймағын анықтау, өндеу аспаптармен жұмыс істеу тәсілдері, компиляциялау, жүктемелеу, программаны орнатуды білу, аспаптарды қолдану мінездемесі мен мүмкіндіктері, олардың ақпараттық қамтамасын талдау, бағдарламалық қамтаманы қолдау және енгізу, жүктеу, қазіргі заманғы өңдеу құралдарын қолдануды практика жүзінде қалыптастыру

Пәннің мақсаты және міндеті:

Пәннің мақсаты: студенттерді бағдарламалық жүйелерді жобалау және программалардың өмірлік циклын қамтамасыздандыру, оларға бизнес-үрдістерді модельдеудің негіздерін меңгерту және жобалаудың қазіргі заман талабына сай технологияларын (Computer-Aided Software/System Engineering (CASE)-технологиялар) тәжірибеде қолдануды үйрету.

Пәннің міндеті: пәнді оқыту нәтижесінде студенттер:

Білуі керек

- бағдарламалық жүйелерді жобалау технологияларын;
- жобалау аймағындағы негізгі бағыты, аспаптық құралдар жиынын таңдау және бағдарламалық өнімдерді өңдеудің өміршеңдік циклін қамтамасыздандыру;
- бағдарламалық өнімді өңдеудегі қолданылатын мемлекетіміздің және шетелдердің стандарттарын қолдануды;
- жабдықтың ақпараттық құрылымын мен интерфейсін құрудағы классикалық және қазіргі заманда қолданылатын әдістерді.

Жасауы керек

- UML модельдеуінің унифицирленген тілін және бағдарламалық жүйелерін жобалауда CASE-құрылымдарын қолдану (BPwin, Erwin, Aris, ModelMart, Rational Rose, Microsoft Office Visio 2007);
- Бағдарламалық жабдықты құру стандарттарын қолдану;
- Жабдықтармен жұмыс істеудегі объектілі-бағытталған және құрылымдық тарату әдістері.

Қолдана алуы керек

- қазіргі заман талабына сай бағдарламалық жүйелерді жобалу технологиялары (CASE)-технологиялары) жөніндегі мағлұматары болуы әрі қолдана алуы керек.

Құзыретті болуы керек

- Жобалаудың қазіргі заман талабына сай технологияларын пайдалануда (Computer-Aided Software/System Engineering (CASE) - технологиялары).

1 Аспаптық құралдар классификациясы

Мақсаты: жобалау әдістеріне шолу, объектілі-бағытталған бағдарламалаудың негізгі түсініктері мен анықтамалары

1.1 Программаларды өңдеудің аспаптық құралдарының даму тарихы

Бағдарламалау технологиясы дегеніміз – бағдарлама құру технологиясындағы қолданылатын әдістер мен тәсілдер жиынын айтамыз.

Басқа технологиялар сияқты бағдарламалау технологиясы – техникалық нұсқаулардан тұрады:

-технологиялық операциялардың қайталанбалы орындалу көрсеткіші

-шарттардың аударылып, осы және басқа операциялардың орындалуы

-осы операциялардың суреттелуі, әр операцияларға негізгі берілгендері, шешімдері, және де қолдану нұсқаулары, нормативтер стандарттар анықталған. Бағдарламалық технологияны толық түсіну үшін осы процесті бірнеше сатыларға бөліп қарастыруға болады:

1 саты. 1940-1960 жылдың ортасына дейінгі аралықты қамтиды. Осы аралықта бағдарламалаудың құрылысы қарапайым болды. Себебі сол кездегі бағдарламалау тілінің мүмкіндіктері шектеулі болғандықтан, күрделі модельді есептерді шешуге мүмкіндік бермеді. Ассемблер тілі пайда болғанға дейін жазылған бағдарламаларды оқудың өзі күрделі процесс ретінде қарастыруға болады. Себебі бағдарламалар 2 немесе 16 код түрінде жазылды.

2 саты. 1970-1980 жж. құрылымдық (структуралық) бағдарламалау сатысы. Құрылымдық бағдарламалау негізінде декомпозициялық әдісті қолданады. Декомпозициялық әдіс дегеніміз - күрделі жүйелерді бірнеше 40-50 операторлардан тұратын ішкі бағдарламаларға бөлу. Осы айтылған әдіс декомпозициялық процедуралық әдіс деп аталады. Құрылымдық бағдарламалау негізінде процедуралық бағдарламалау тілін қарастыруға болады. Құрылымдық бағдарламалауға келесі тілдерді жатқызуға болады: PL/1, ALGOL-68, Pascal, C . Ауқымды мәліметтермен жұмыс істеуді жеңілдету үшін модельдік бағдарламалау тілі пайда болды. Модельдік бағдарламалау бір ғана ауқымды мәліметтерді бірнеше ішкі бағдарламалардың топтарына қолдануға мүмкіндік береді. Бір модельдік бағдарламада 100 мың операторға дейін қолдануға болады. Компьютерлік техниканың дамуы бағдарлама өлшемін ұлғайтуға, өте күрделі жобаларды шешуге және осы жоба үшін бірнеше ішкі бағдарламаларды кең көлемде қолдануға жол ашты. Бірақ ішкі бағдарлама көбейген сайын ішкі бағдарламаларды шақыру интерфейсінде қателіктер көбейе бастады.

3 саты. 1980-1990 жылдардың аяғына дейінгі кезеңді қамтитын нысанды бағытталған бағдарламалау сатысы. Нысанда бағдарламалаудың технологиясы күрделі бағдарламаларды құруға арналған. Бағдарламаны құруға арналған нысандар өздерінің типтері, қасиеттеріне байланысты белгілі бір иерархиялық тәртіппен орналасқан нысандардың жиынтығын құрайды. Мұндай жүйеде бағдарламалық нысандар бір-бірімен хабарлар беру арқылы байланыс

орнатады. Нысанды бағытталған бағдарламаудың моделі басқа түрлі бағдарламаудан негізгі артықшылығы, бағдарламалардың декомпозицияларын және күрделі жобалы нысандарды тез құруға мүмкіндік береді. Бұл тәсіл күрделі бағдарламалардың бөліктерін бір-біріне тәуелсіз жеке-жеке құруға және визуальды бағдарламалау ортасында жұмыс істеуге жол ашты. Бірақ нысанды бағытталу тілдері объект Pascal, C++ тілдерінен көптеген артықшылықтармен қатар кемшіліктері де жеткілікті.

4 саты. 90 жылдардың ортасынан бастап қазіргі кезге дейінгі кезеңді қамтитын компоненттік әдіс және CASE операторы. Компоненттік әдіс әр түрлі компоненттерден тұратын бір-біріне қатыссыз бағдарлама бөліктерін бір стандартты библиотекаға біріктіруге арналған, осы бөліктер бір-бірімен екілік интерфейстердің стандарттары арқылы байланыс орната алатын бағдарлама құруға арналған. Қарапайым нысандардан компоненттер нысандарының айырмашылығы осы нысандардың қасиеттері бойынша жинақтап шақыруға арналған файлдар күйінде беріледі. Динамикалық библиотекадағы нысандар екілік код түрінде берілгендіктен кез-келген бағдарламалық тілде қолдана беруге болады. COM (компонент объект модуль) базасын технологиясының және қосымшаларды таратудың CORBA технологиясын құруға компоненттік әдісті негіз ретінде алынған. Компьютерлер арасындағы ақпараттарды алмастыруды қамтамасыз ететін модификацияланған COM түрін DCOM деп атайды. Қарапайым нысандардан әрбір COM нысанының айырмашылығы өрістер арқылы функцияларға қатынас орнатуға арналған бірнеше интерфейстерді қолдана алады.

1.2 Ұғымдарды анықтау

Бағдарламалау технологиясы — берілген есепті шешуге құралған бағдарламада қолданатын барлық әдістерді, құрылымдарды және модельдердің жиынтығын айтады.

Компьютерлік бағдарлама — орындалуға (өңделуге) тиіс реттелген командалар тізбегі, есеп шығару алгоритмін сипаттайтын бағдарламалау тілінің сөйлемдер жиыны. Есеп шығаруға, сондай-ақ берілген мәселені шешуге арналған, қабылданған синтаксиске сәйкес жазылған компьютер командаларының (нұсқауларының) реттелген тізбегі;

Бағдарлама жасақтамасы(software) — компьютердің жұмыс істеуіне арналған компьютер бағдарламалары жиынтығы, яғни бағдарламалық жасақтама.

Тапсырма (problem, task) — шешілуі тиіс мәселе.

Қолданба (application) — компьютерде 1 мәселені шешудің бағдарламалық қамтамасыз етуді іске асыру.

Бағдарламаларды жасау процесі мынадай іс-шаралар тізбегі ретінде ұсынылуы мүмкін:

1. мәселенің тұжырымы;
2. мәселенің алгоритмдік шешімі;

3. бағдарламалау.

Міндеттің қойылуы (problem definition) — бұл кіріс және шығыс ақпарат сипаттамасымен компьютерде мәселені шешудің нақты тұжырымы болып табылады.

Алгоритм — алға қойылған мақсатқа жету үшін немесе берілген есепті шешу үшін түсінікті де нақты ережелер бойынша орындаушыға жинақы түрде берілген,реттелген нұсқаулар тізбегі.

Бағдарламалау (programming) — компьютерде есеп шешу үшін оны алдын ала дайындау процесі.

Бағдарламалық жасақтамаға қатысты, компьютер пайдаланушылар мынадай топтарға бөлінеді:

1. Жүйелік бағдарламашы. Жүйелік бағдарламалық қамтамасыз етуді дамыту, пайдалану және қызмет көрсетумен айналысады;
2. Қолданбалы бағдарламашы. Көбіне ойындар, есептеу бағдарламалары, редакторлар мен мессенджерлерді қамтамасыз ететін бағдарламаларды жабдықтаумен айналысады.
3. Соңғы пайдаланушылар. Олар негізгі компьютерлік дағдылар мен қосымшалармен жұмыс істейді;
4. Желілік администратор. Компьютерлік желілердің жұмысына жауапты;
5. Мәліметтер басқарушысы. Мекемеде (кәсіпорында) пайдаланылатын мәліметтер жайлы толық мағлұматы бар және оларды сақтауға, жаңарту мен пайдалануды ұйымдастыруға жауапты адам.

Бағдарламаны қолдау — бағдарламаны жұмыс істеу қалпын қолдау, жаңа нұсқасына көшу, өзгерістер енгізу, қателерді түзеу.

Абстракция

Абстракция(лат. Abstractio - қаз. бұру, аудару) - ойлаудың негізгі операцияларының бірі. Жеке мақсатқа қатысты ақпаратқа жұмылдырылған және басқа ақпаратты ескермейтін көзқарас.

Инкапсуляция

Инкапсуляция – Осы деректерді өңдеу алгоритмдері мен деректерді бір бүтінге біріктіру. ОББ деректері көлемінде – объект жолы, алгоритмдері – объектілік әдістер алынады.

Мұрагерлену

Мұрагерлену – обьектілер қасиеті мұрагерлерлікпен туындайды және оларға өздерінің қасиеттерін үнсіздікпен ұсынады. Объект – мұрагерленуші, ол қасиеттерді толықтыра алады немесе басқамен оны ауыстыра (орынбастыра) алады.

Полиморфизм

Полиморфизм – туыстық обьектілердің қасиеттері (яғни бір объектіден тараған бір түбірі бар обьектілер) негізінен бір – біріне ұқсас, мәні жағынан бірдей проблемаларды әртүрлі әдістермен, оның уақытына және орналасу ретіне байланысты шешу.

Класс

Класс объектінің негізін құрайды. Ол объектіні жасайтын айнымалылар

мен осы айнымалыларды пайдаланатын функциялардан құралады.

Объект

Объект — алдын-ала құрылып қойылған кластан туындайтын экземпляр.

Прототип

Прототип — түп тұлға — бұл үлгі түрі және ұқсастығы бойынша басқа объектілерде құрылатын объект

Бағдарламалардың негізгі ерекшеліктері:

1. алгоритмдік күрделілік;
2. ақпараттық-өңдеу функциялары;
3. бағдарлама арқылы қолданылатын файлдар көлемі;
4. дискілік кеңістік көлемі, процессор түрі, операциялық жүйе нұсқасы, компьютерлік желінің қол жетімділігі және т.б..

1.3 Жобалау технологияларына шолу

MS Visio беттері және олардың қолданылуы

Дұрыс құрылған класс тек бір ғана абстракцияны бере алады. Мысалы, студент туралы мәлімет сақталған, сонымен қатар барлық оқу барысында студент өткен курстар тізімі функциясы көрсетілген класты сәтті құрылған деп айта алмаймыз, өйткені ол екі әртүрлі операциялар тобын қамтиды. Сондықтан мұндай класты екіге – «студент» және «студент өткен курстар» бөлсе жақсы болар еді. (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (ORACLE), New Era (Informix), SQL Windows (Gupta), Delphi (Borland) және т.б.

Қосымша типтері:

- жобалау және басқару құралы (SE Companion, Microsoft Project және т.б.);
- конфигурациялық басқару құралы(PVCS (Intersolv));
- тестілеу құралы(Quality Works (Segue Software));
- құжаттау құралы (SoDA (Rational Software)).

Қазіргі таңда бағдарламалық қамтамасыз ету CASE- құралдарға ие:

- Vantage Team Builder (Westmount I-CASE);
- Designer/2000;
- Silverrun;
- ERwin+BPwin;
- S-Designor;
- CASE Аналитика.

Қазақстанда бизнес-процестерді модельдеу және анализдеу үшін келесі модельдеу құралдары жеткілікті түрде кең қолданылады: Rational Rose, Oracle Designer, AllFusion Process Modeler (BPWin) және AllFusion ERwin Data Modeler (ERWin), ARIS, Power Designer. Шет елдерде, аталғандардан басқа, белсенді мына құралдар қолданылады: System Architect, Ithink Analyst, ReThink және тағы басқалары. 1-кестеде талқылауға қатысатын аспапты құралдар тізбегі көрсетілген:

- аспапты құралдар атауы;
- аспапты құралдардың қысқаша сипаттамасы.

1 кесте – Құралдар тізбегі

Аталуы	**Жабдықтаушы**	**Қысқаша сипаттамасы**
BPWin және ERWin	Computer Associates компаниясы	BPWin - бизнес – процестерді көзбен шолып модельдеудің аспабы
Аталуы	Жабдықтаушы	Қысқаша сипаттамасы
Oracle Designer	Oracle компаниясы http :// www. oracle. Com	Заттық аймақты сипаттау үшін арналған функциялық құралдар. Аспапты құралдар кешеніне программалар жүйесін және мәліметтер базасын жобалауға арналған Oracle 9i Developer Suite кіреді. Ол өзіне меншікті Oracle «CDM» компаниясының АЖ
Rational Rose	IBM компаниясы http :// www. ibm. Com	Объекттті - бағдарлы ақпараттық жүйелерді жобалау құралы. Ақпаратты жүйелерді жобалау кезінде кез-келген есепті шешуге мүмкіндік береді бизнес - процестерді анализдеуден бастап, белгілі бағдарламалау тілінде кондогенерациялауға дейін жоғарғы деңгейлі
ARIS	IDS компаниясы Scheer AG http://www.ids-scheer.com	Бизнес процестерді модельдеудің интегралданған құралы, жүйелердің анализі және модельдеудің сан - алуан әдістерін біріктіреді. Бірінші кезекте, жобалау құралынан гөрі, бұл сипаттау, анализдеу, бизнес - процестерді құжаттау және үйлесім деу құралы болып табылады. Дүние жүзі нарығының көшбасшысы. Оқшауланған.
System Architect	Telelogic компаниясы (алдында Popkin Software болған, кәзіргі уақытта Telelogic компаниясының бөлімшесі болып табылады) http://www.telelogi	System Architect әмбебап CASE - құралы болып табылады, мәліметтерді жобалауда ғана емес, сонымен қатар құрылымдық жобалауды жүзге асыруға мүмкіндік береді. Мәліметтерді жобалау және ER диаграммасын құру құралы осы өнімінің құрама белшектерінің бірі болып табылады. Дүние жүзі көшбасшыларының бірі, бірақ әлі Қазақстан нарығында танылмаған. Оқшауланууы 2006 жылдың шілде айына

	c.com	бағдарланып койылған.

Кестеде келтірілген аспапты құралдар тізімінен көрсетілген критерилерді қанағаттандырылатын анағұрлым толық анализ жасауға арналған бағдарламалық өнімдерді бөліп аламыз. Бұл жағдайда одан арғы талқылауға BPWin/ERWin, Oracle Designer, Rational Rose, Power Designer, ARIS осылар бойынша төменде толық сипаттама келтірілген. BPWin және ERWin, Computer Associates компаниясы. Computer Associates International. Inc (CA) бағдарламалық қамтамасыз етуді өндіретін бес жетекші компания қатарына кіреді. Модельдеу құралдарын, резервті көшірмесін, кәсіпорынның инфра құрылымын басқаруда (желімен, сервермен) ақпараттың қауіпсіздікті, business intelligence және т.б. ұсынады. BPWin дестесі IDEF әдістемесіне негізделген және кәсіпорынның қызметін анализдеу мен функционалды модельдеу үшін қолданады. IDEF әдістемесі АҚШ-тың ресми федералды стандарты болып табылады, қандайда бір заттық аймақтың объектісін функционалды модельдеу құру үшін арналған рәсімдердің, ережелердің, әдістердің, жиынтығын көрсетеді IDEF функционалды моделі объектің функционалды құрылымын бейнелейді яғни олардан шығарылатын іс - әрекеттер мен осы іс - әрекеттер арасындағы байланыс. ERWin дестесі бұл МБ концептуалды модельдеу құралы. «Маңыз - байланыс» диаграммасы негізінде ерікті күрделі мәліметтер базасын модельдеу және құру кезінде қолданылады. Қазіргі уакытта ERWin әртүрлі кластың СУБД кең спектрін қолданудың арқасында мәліметтерді модельдеудің анағұрлым атақты дестесі болып табылады. Oracle Designer, Oracle компаниясы. Oracle Designer аспапты құралдарының жиынтығы клиент / серверлі үстеме және Web-ке арналған корпоративті деңгейдегі қолданбалы жүйені жасауға арналған интегралданған шешім ұсынады. Oracle Designer бағдарламалық қамтамасыз етудің жасалуының өмірлік циклының әр бір сатысында қатысады: бизнес - процестердің модельдеуден бастап еңгізуге дейін қолданылады. Rational Rose, IBM компаниясы. IBM Rational Rose - IBM Rational Suite дестесінің құрамына кіреді және кең ауқымды аспапты құралдармен платформаларды қолданатын бағдарламалық жүйелерді модельдеу үшін арналған. PowerDesigner, Sybase компаниясы. Sybase компаниясы құрылған уақыттан бері дүние жүзі нарығының қаржы институттарын ақпаратты технологиялармен дәстурлі жетекші жабдықтаушысы болып табылады: дүние жүзі нарығындағы бағалы қағаздар компанияларының 90 %- і. дүние жүзі банктерінің 60 % -і, Wall Street компанияларының 68 % - і Sybase технологиясын қолданады. 1996 жылдан бері

Мәскеуде кеңсе ашылғаннан бері, Sybase Қазақстанда және ТМД елдерінің басқа да мемлекеттерінде белсенді жұмыс үстейді. ARIS, IDS Scheer AG компаниясы. Қазіргі уақытта әр түрлі методтарды модельдеу және жүйені анализдеу интеграция тенденциясы байқалады, интегралданған құрал модельдеу құру ретінде көрсетілді. Әрбір қарастырылатын өнімнің оның көптеген колданылуларының арасынан негізгі арналғанын көрсетейік:

- мәліметтер базасын модельдеуге көбінесе құралдар қолданылады;
- жасалынып жатқан үстемелердің компоненттерін модельдеу үшін Oracle Designer, Power Designer және Rational Rose қолданады.

ERwin орындалатын міндеттерге арналған (деректер моделін) визуалды көрсетуді жасайды. Бұл ұсыныс жасау циклінде қажетті құжаттың бөлігі ретінде нақты саралауға, дәлелдеуге және таратуға қолданылуы мүмкін. ERwin тек салу құралы ғана емес, ол деректер қорын автоматты түрде жасайды.

Erwin диаграммасының негізгі компоненттері – бұл мәндер, атрибуттар және байланыстар. Әрбір мән даналар деп аталатын, жеке нысандар тәрізді жиынтықтар болып табылады. Әрбір дана дербес және басқа барлық даналардан өзгеше болуы қажет. Мәліметтер моделін құру мәндер мен атрибуттар анықтамасымен болжамдалады, яғни нақты мәнде немесе атрибутта қандай ақпарат сақталатынын анықтап алу қажет. Егер диаграмманы пәндік аймақтың ережелерінің графикалық көрсетілімі ретінде қарастырса, онда мәндер мен атрибуттар зат есім, ал байланыс етістік болып табылады. BPwin – функциональдық модель (немесе процестер моделі). BPwin семантикалық жағын ескере отырып, модель құру процесінің есебін автоматтандырады. BPwin ұйымның жұмыс барысы туралы ақпарат жинап және сол ақпараттың бүтін үлгі ретіндегі графикалық суретін ұсынады. **BPWin** бизнес – үдерісті талдау үшін, осы үдерістің функционалды моделін басқа сөзбен айтқанда үдеріс үлгісін жобалауға мүмкіндігін береді. BPWin диаграммаларында бизнестің үдерістері жұмыстың өзара байланысы және тәуелділігі, жұмыстардың нәтижелері материалдар, ақпарат және жұмыс орындалуы үшін қажетті ресурстар, жұмысты жүзеге асыратын жұмыскерлер және әр жұмыс түрінің құны осы жұмысқа жіберілген шығындар сомасы көрсетіледі. **ERWin** модельдерді диаграмма арқылы көрсетеді. Диаграммалар бизнес – үдерісі қалай орындалу керектігін, бизнесте қолдану үшін қажетті ақпаратты көрнекті көрсетеді, сонымен қатар орындалатын жұмыстардың әрқайсысы үшін жауапты қызметтерді белгілейді. Осы мүмкіндіктердің нәтижесінде жұмыстың орындалу барысында кейбір жіберілген қателіктер жойылып отырады. **ERWin** қосымшаның қызметші бөлігі үшін **ERWin** мәліметтердің логикалық моделін қалыптасутыруды қамтамасыз етумен бірге құруға мүмкіндік береді. Осыдан кейін **ERWin** жүйесі көмегімен таңдап алынған мәліметтерді басқару жүйесінде бүтіндікті ескере отырып физикалық нысандарды жасауға болады. Осыны пайдалана отырып логикалық модельмен нақты мәліметтер қорын сәйкестендіру жедел түрде екі бағытта да қамтамасыз етіледі.

BPwin үш методологияны қолдайды: бизнесті үш көзқарас тұрғысынан саралайтын IDEF0, DFD және IDEF3.

Жүйенің функционалдылығы тұрғысынан. IDEF0 (Integration Definition

for Function Modeling) әдістемесінің тұрғысынан бизнес-процесс өзара әрекеттесетін жұмыс-элементтер ретінде, сонымен қатар ақпараттық,әр жұмыста қолданылатын ресурстар жиыны ретінде бейнеленеді.

Ақпарат ағымы көзқарасы тұрғысынан. DFD(Data Flow Diagramming) диаграммалар IDEF3 модельде көрсетілген ақпаратты тек қана толықтыра алады, себебі ол ақпарат ағымын бақылауға мүмкіндік береді. Бірақ DFD диаграммалар бизнес функциялар арасындағы әрекеттесуді елемейді.

Орындалатын жұмыстар реттілігі көзқарасы тұрғысынан IDEF3 диаграммаларының көмегімен дәл суреттеме алуға болады. Бұл әдіс әрекеттердің орындалу ретіне көңіл бөледі. IDEF3-те бизнес-процестердің даму барысын талдауға мүмкіндік беретін логикалық элементтер қосылған.

BPwin құрылғалы отырған модельді таңдалған методология синтаксисі тұрғысынан тексере алады, диаграммалар арасындағы сілтемелік тұтастықты тексереді және дұрыс модель құру үшін басқа да тексерулерді орындайды. Сонымен қатар, негізгі артықшылық қарапайымдылық пен құру жеңілдігі сақталады.

Rational Rose – дегеніміз автоматтандыру процестерін талдау және ПО жобалау үшін арналған, сонымен қатар әртүрлі тілдердегі кодтарды генерациялауға және жоба - құжатнамаларды шығаруға арналған Rational Software Corporation фирмасының объекттілі - бағытталған Case құралдары. Rational Rose UML тіліне негізделіп жобалау және обьектілі бағытталған талдау әдістерін қолданады. Rational Rose осы болжамасы C++, Visual C++, Visual Basic, Java, PowerBuilder, CORBA Interface Definition Language (IDL) бағдарламалар үшін кодтар генерациясын және ANSI SQL, Oracle, MS SQL Server, IBM DB2, Sybase үшін мәліметтер қорының генерация бейнеленуін, сонымен қатар диаграмма түріндегі жобалау құжаттарын және егжей-тегжейлерін іске асырады. Rational Rose жаңа жобаларда бағдарламалық компоненттерінің қайта қолдануын қамтамасыз ететін бағдарламалар мен мәліметтер қорының реверстік инжинирингтің құралдарынан тұрады.

2 Әдістер және аспаптар

Мақсаты: CASE- құралдарын енгізу және игеру технологиясы. CASE – құралдарының сипаттамасын қарастыру. Объектілі - бағытталған CASE – технологиясы.

2.1 Қазіргі заман талабына сай CASE-технологиялар

CASE-құралдары (Computer Aided Software Engineering) талдау, талаптарды қалыптастыру, қолданбаны және мәліметтер қорын жобалау, кодты генерациялау, тестілеу, сапаны қамтамасыз ету, конфигурацияны және жобаны басқару сияқты ақпараттық жүйелерді сүйемелдеу және құру үрдістерін қолдайтын программалар. Яғни, CASE-құралдары жай мәліметтер қорын жобалау тапсырмаларын ғана емес, өте үлкен көлемдегі тапсырмаларды шешуге мүмкіндік береді. Delphi жүйесі де CASE типіне жатады, өйткені қолданбаны жүзеге асыруды автоматтандыруға мүмкіндік береді. CASE жүйесін CASE-құралдар жиынтығы ретінде анықтауға болады.

Мәліметтер қорын жүзеге асыру үшін қолданылатын CASE-құралдардың жіктелуі келесі белгілері бойынша жүргізіледі:

- өмірлік цикл кезеңдеріне бағыну;
- функционалдық толықтық;
- қолданылатын үлгілер типі;
- мәліметтер қорын басқару жүйесінен (МҚБЖ) тәуелсіздік деңгейі;
- платформа.

Өмірлік цикл кезеңдеріне бағыну бойынша CASE жүйесінің келесі негізгі типтерін атап өтуге болады: жобалық спецификацияларды қолдайтын және қамтамасыз ететін талдау және жобалау жүйелері, мысалы, Vantage Team Builder (Cayenne), Silverrun (Silverrun Technologies), PRO-I (McDonnell Douglas); негізгі МҚБЖ-лері үшін мәліметтерді үлгілеу және мәліметтер қорының сызбасын жасауды қамтамасыз ететін мәліметтер қорын жобалау жүйелері, мысалы, ERwin (Logic Works), SDesigner (SPD), DataBase Designer (Oracle);

Қолданбаны жасау жүйелері, мысалы, Uniface (Compuware), JAM (JYACC), PowerBuilder (Sybase), Developer/2000 (Oracle), New Era (Informix), SQL Windows (Centura), Delphi (Borland) Функционалдық толықтық бойынша CASE жүйелері шартты түрде келесі топтарға бөлінеді:

Өмірлік циклдің бір немесе бірнеше кезеңдеріндегі жекеленген есептерді шығаруға арналған жүйелер, мысалы, ERwin (Logic Works), S-Designer (SPD), CASE.Аналитик (МакроПроджект) және Silverrun (Silverrun Technologies);

Ақпараттық жүйенің барлық өмірлік циклін қолдайтын интегралданған жүйелер, мысалы,Vantage Team Builder (Cayenne) жүйесі және Designer/2000 (Oracle) жүйесі;

Қолданылатын үлгілер типі бойынша CASE жүйелері үш түрге бөлінеді: құрылымдық, объектілі-бағытталған және комбинарлық.

2.2 CASE – құралдарын бағалау

Тарихи бірінші құрылымдық және модульдық бағдарламалау, құрылымдық талдау және синтез әдістеріне негізделетін құрылымдық CASE жүйелері пайда болды, мысалы, Vantage Team Builder (Cayenne). Объектілі-бағытталған CASE жүйелері XX ғасырдың 90-шы жылдарының басынан бастап кең тарала бастады. Олар өңдеу мерзімін қысқартуға, сонымен қатар ақпараттық жүйенің функционалдық тиімділігін және сенімділігін жоғарлатуға мүмкіндік береді. Объектілі-бағытталған CASE жүйелерінің мысалдары болып, Rational Rose (Rational Software) және Object Team (Cayenne) табылады.

Комбинарлық CASE жүйелері бір уақытта құрылымдық және объектілі-бағытталған бағдарламалауды қолдайды, мысалы, Designer/2000 (Oracle). МҚБЖ-нен тәуелсіздік деңгейі бойынша, CASE жүйелері екі топқа бөлінеді: тәуелсіз жүйелер; МҚБЖ-не орнатылған жүйелер. Тәуелсіз CASE жүйелері нақты МҚБЖ-нің құрамына кірмейтін автономдық жүйелер түрінде жеткізілімді қарастырады. Әдетте, олар ODBC интерфейсі арқылы мәліметтер қорының бірнеше форматын қолдайды.

Тәуелсіз жүйелер қатарына SDesigner (SPD), ERwin (Logic Works), Silverrun (SilverrunTechnologies) жатады. Орнатылған CASE жүйелері әдетте мәліметтер қорының форматын қолдайды. МҚБЖ Oracle құрамына кіретін орнатылған жүйелер мысалы болып, Designer/2000 табылады. Платформа компьютерді және операциялық жүйені анықтайды. Delphi көмегімен қолданбаны және мәліметтер қорын жасау кезінде қолданылатын CASE-құралдарын атап өтейік: ModelMaker – Delphi 7-мен бірге жеткізілетін өнім.

Delphi құрауыштарының дестелерін және кластарын жасауға қызмет етеді. Delphi-дің генерациялау кодына бағытталған CASE-құрал болып табылады. Класстар және олардың мүшелері арасындағы қатынасты сақтауға және қызмет көрсетуге, UML-диаграммаларды құруды қолдауға мүмкіндік береді. Басқа генераторлар кодымен салыстырғанда ModelMaker күрделі жобаларды жасауға мүмкіндік береді.

DataModule Designer – мәліметтер қорын Paradox форматындағы кестелермен жобалауға мүмкіндік береді. Программа ыңғайлы және көркем интерфейсті қамтамасыз етеді. Мәліметер қорының құрылымы, сонымен қатар кестелер арасындағы байланыстар графикалық түрде көрсетіледі.

Cadet – тәуелсіз өнім, dBase, Paradox және InterBase форматындағы кестелермен мәліметтер қорын жобалауға мүмкіндік береді. Көрсетілген форматтар Delphi үшін жақын болып табылған жағдайда, Cadet программасын ақпараттық жүйені жасау кезінде қолданған ыңғайлы.

Data Module Designer және Cadet мәліметтер құрылымын үлгілеу және мәліметтер қорын жобалауды автоматтандыруға арналған программалар.Осы құралдармен көрсетілетін мүмкіндіктер мысалы, Sdesigner сияқты қуатты жүйелердің мүмкіндіктеріне қарағанда аз.

Cadet программасы шартты тегін болып табылады, ал Data Module Designer Paradox 7.0 МҚБЖ құрамына кіреді. ModelMaker пайда болғаннан кейін, басқа CASE-құралдарын қолдану қажет болмауы мүмкін. Қазіргі заманғы

автоматтандырылған басқару жүйелерінің күрделілігінің жоғарлауы және оған қойылатын талаптардың өсуі өмірлік циклдің барлық уақытында ақпараттық жүйені құруда және сүйемелдеуде тиімді технологияларды қолдануға негізделеді. Ақпараттық жүйелерді дайындау әдістемесіна және сәйкес интегралданған инструменталдық құралдар кешеніне негізделген, сонымен қатар ақпараттық жүйелердің толық өмірлік циклін немесе оның негізгі кезеңдерін қолдауға бағытталған мұндай технологиялар, CASE-технологиялар және CASE-құралдар атына ие болды. Ақпараттық жүйенің жобасын жүзеге асыру үшін толық және қарама-қайшылықсыз функционалдық және басқару жүйелерінің ақпараттық үлгілері құрылуы тиіс. Атап өткен үлгілердің жинақталған тәжірибесі, бұл логикалық күрделі, қиын және ұзақмерзімдік жұмыс, жоғары біліктілікті мамандарды қажет ететіндігін көрсетеді. Әдетте, көп жағдайларда ақпараттық жүйені жобалау негізінде эксперттік бағаларға және тәжірибелік зерттеулерге негізделген қалыптастырылмаған әдістерді қолдану арқылы интуитивті деңгейде орындалады.

Сонымен қатар, ақпараттық жүйенің функционалдау және құру үрдісінде қолданушылардың ақпараттық қажеттіліктері өзгеруі немесе нақтылануы мүмкін, бұл автоматтандырылған басқару жүйелерін жасауды және сүйемелдеуді одан әрі қиындатады. Осы кемшіліктеріне байланысты, ақпараттық жүйе құру және сүйемелдеу CASE-технологияларын жүзеге асырушы арнайы CASE-құралдары класының программалы-техникалық құралдарына негізделген тұрғылар еркін болады.

CASE (Computer Aided Software Engineering) термині ретінде, ақпараттық жүйені құру және сүйемелдеу үрдістерін, сонымен қатар талдау және талаптарды қалыптастыру, қолданбалы бағдарламалық жасақтаманы және мәліметтер қорын жобалау, кодты генерациялау, тестілеу, құжаттандыру, сапаны қамтамасыз ету, конфигурациялық басқару және жобаны басқару, және т.б. үрдістерді қолдайтын бағдарламалық құралдар деп түсінеміз.

BPwin, сол модельді құру үшін керек – функциональдық модель (немесе процестер моделі). CASE технологиялары күрделі бағдарламалық жүйелерді талдау, жобалау, жасау және сүйемелдеу методологияларының жиынтығы болып табылады. Олар өзара біріктірілген автоматттандыру құралдар кешені көмегімен құрылымдық және обьектілік тұрғыларға негізделеді. Кез-келген CASE технологиясы негізінен методология/әдіс/нотация/құрал/парадигмасы алынады. Методология қандай да бір тұрғының негізінде жасалып, жұмыстың қадамдарын, қадамдардың орналасу тәртібін, сонымен қатар әдістер міндеттері мен үйлестірілу ережелерін анықтайды.

Әдіс қандай да бір мақсатқа жету, яғни жұмыстың белгілі бір қадамын орындау тәсілін анықтайды. Нотация деп модельдердің қандай да бір класын сипаттау үшін қолданылатын таңбалар жүйесін айтады. Нотацияның графикалық және текстілік түрлері болады. Графикалық нотациялар, диаграммалар, кестелер, схемалар түрінде сипаттайды, ал текстілік нотациялар модельдерді формальді және кәдімгі тілде сипаттайды. CASE технологиясын нотациялар жобаланатын жүйенің деректер элементтерінің жасау кезеңдерінің құрылымын сипаттау үшін қолданылады. Құралдар әдістерді жүзеге асыру

үшін қажетті құрал - жабдықтарды құрайды. Олар: графикалық жобаны жасау және жөндеу құралдары, жобаны абстракция деңгейі иерархиясы түрінде ұйымдастыру құралдары. Сонымен қатар түрлі деңгейлер компьютердің сәйкестігін тексеру құралдары.

CASE құралдарының мынадай түрлерін бөліп қарауға болады.

- талаптарды талдау, спецификациялары мен құрылымдарды жобалау, интерфейстерді жөндеудің CASE құралдары (CASE1 - бірінші ұрпағы).
- бағдарламалық қамсыздандыруды жасаудың толық өмірлік циклін қарастыратын біріктірілген орталардың жүзеге асыру және бастапқы текстерді генерациялаудың CASE құралдары (CASE2 - екінші ұрпағы).

CASE1 негізінен графикалық модельдерді спецификацияларды жобалаудың экрандық редакторлар мен берілген сөздікті қолдау құралдарынан тұрады.

CASE2 айталықтай үлкен мүмкіндіктерімен ерекшеленеді. Мұнда жүйелік ақпарат пен жобалық процесін басқару бойынша ақпаратты бақылау, талдау және байланыстыру, жүйенің модельдері мен прототиптерін жасау, генерацияланған программаларды тестілеу, мақұлдау және талдау қамсыздандырылады.

Көп еңбекті қажет ететін операцияларды автоматтандыра отырып, қазіргі заманғы CASE технологиялар бағдарламалаушы мамандардың еңбек өнімділігін айтарлықтай өсіріп, жасалатын бағдарламалық қамсыздандырудың сапасын көтерді. Олар: Жоба спецификацияларының сәйкестігінің автоматты түрде бақылануын қамсыздандырады. Жүйе прототипін жасау уақытын қысқартады. Жобалау және жасау процесін жеделдетеді. Өмірлік циклдің барлық кезеңдері үшін арналған жобалық құжаттаманың қазіргі заманғы стандарттарға сәйкес жасалуын автоматтандырады. Кейбір бағдарламалық кодтарды түрлі жасау платформалары үшін генерациялайды. Жүйе компонентін қайтара қолдану технологиясын қолдайды. Жобалық құжаттарды бастапқы кодтар бойынша қалпына келтіру мүмкіндігін қамсыздандырады.

2.3 Қазіргі заман талабына сай CASE-құралдарының сипаттамасы

CASE – технологиялары жүйеге талаптар қою және жобалау үдерістерін қысқарту үшін CASE жабдықтары қолданылады. XX ғасырдың 70 – 80 жылдары талдаудың құрылымдық технологиясы қолданыла бастады. Құрылымдық технология графикалық көрнекті техниканы қолдануға негізделген. Графиктік техника әртүрлі модельдерді сипаттауға арналған. Қазіргі кезде CASE құрделі құрылымды сүйемелдеу үдерістерін қамти алады. CASE технологиялары келесілерден тұрады:

- Ақпараттық жүйелерді жобалау әдістері;
- Нотация (жүйе элементтерін бейнелеу тәсілі);
- Инструменталды жабдықтар;

CASE жабдықтарының негізгі функциялары:

- Жобаның орталықтандырылған деректер қорында сақталынады. Орталықтандырылған ДҚ *репозиторий* деп аталады. Ол әртүрлі типті объектілерді сақтай алады: Диаграмма, Деректерді сипаттау, Программаның алғашқы коды, Бағдарламалық жабдықтау мен ДҚ жобалау.

CASE жабдықтарының қолдану реті:

- Жүйенің логикалық моделі құрылады
- Нақты бағдарламалау тілі немесе физикалық үлгіні құру үшін деректер қорын басқару жүйесі таңдалынады
- Физикалық модель әрі қарай өңделінеді
- Программаның мәтінін немесе дискідегі деректер құрылымын автоматты түрде генерациялау орындалады.
- Кері жобалау (реинженеринг). Бұл жағдайда CASE жабдықтарын қолдану кері бағытта болады, яғни программа мәтінін немесе дискідегі деректер құрылымын логикалық моделге ауыстырамыз
- Физикалық түрде іске асырумен жүйелер моделін синхронизациялау. Бұл жағдайда жүйенің физикалық моделіне қажетті өзгерістер енгізілуі мүмкін
- Сапаны автоматты түрде қамтамасыз ету және модельді қателерге тексеру
- Құжаттарды автоматты түрде генерациялау.

CASE технологияларын қолданудың *мақсаты*: жүйені жобалау мен талдау сатыларын максималды түрде автоматизациялау.

Заманауи CASE жабдықтары ақпараттық жүйелерді талдау мен жобалау кезінде объектіге бағытталған технологияларды қолданады.

CASE жобалауының бір – бірінен ерекшелігі жүйенің декомпозициясы (орындалатын жұмыстар), тәсілдерін талдау болып табылады. *Жобалаудың заманауи әдістері:*

2 кесте – Жобалаудың заманауи әдістері

Әдістеме	**Модель типтері**
SADT (Structured Analysis and Design Technique)	Функционалды модель
DFD (Data Flow Diagrams)	Функционалды, ақпараттық және компонентті
ERD (Entity-Relationship Diagrams) – мағына – мән диаграммасы	Ақпараттық
STD (State Transition Diagrams) – күй диаграммасы	Қалыптық модель

Flowcharts(блок - схема)	Қалыптық, ақпараттық және компонеттік

Функционалдық диаграммалардың қызметі бағдарламалық жабдықтар құрамындағы функциялардың өзара байланысуын, иерархиясын көрсетеді. Функционалдық диаграммаларды функционалдық модельдер деп те атайды. Функционалдық модельдің көп тараған түрінің бірі SADT (Structured Analysis and Design Technigue –технология структурного анализа и проектирования). Оны 1973 жылы Д. Росс ұсынған. Функционалдық диаграммаларды тұрғызу келесі қағидаларға негізделген :

- әрбір функция бір блок ретінде қарастырылады;
- әрбір блок үшін бастапқы мәлімет, басқарушы команда, функцияны орындаушы механизм (бағдарламалық жабдық немесе техникалық құрылғы) және нәтиже анықталады.

Бағдарламалық жасақтамаларды автоматтандырылған түрде әзірлеу, case-технологиясы — бағдарламалық жасақтама әзірлеу кезінде компьютерлерді қосалқы құрал ретінде пайдалану. Бағдарламалық жасақтаманы жобалау, құжаттамаларды туындату және қызметтің басқа да түрлерінде аспаптық бағдарламалық құралдарды пайдалану көзделген.

Қазіргі заманғы CASE - құралдары АЖ жобалау технологиясының көптеген аумағына ие: қарапайым анализ құралынан және құжаттық құрал автоматизациясында, барлық бағдарламалық жасақтаманың өмірлік циклін құрады.

АЖ еңбекті талап ететің кезең анализ және жобалау болып табылады, процесте CASE - құралдары жоба құжатын дайындайды. Және де ақпаратты визуалды тәсілі басты рөлді атқарады. Бұл құрылымдық немесе диаграммалардың масштабты уақытын білдіреді, көптеген палитра түстерін қолданады. Графикалық құрылым модельдеуі пәндік аймақта АЖ қойылған мақсатқа және шектеулерге қайта құру.

CASE - құралының разрядына дербес компьютерлердің шектелген мүмкіншіліктері және қымбат бағалы жүйе есептеу платфотмасының ОЖ үшін қажет.

CASE-құралының диаграммасы бір-бірімен 3-6 дейінгі блоктардан тұрады және модель құрылымының бірнеше түрлері бар. Бір диаграммасын айыра білу үшін С - номерлері қолданылады. Диаграммадағы блоктар жүйелік функцияларды көрсетеді, ал иін көптеген обьектілердің жүйесін көрсетеді және бірнеше бұтақтарға бөлінеді, әр түрлі күрделі әдістермен қосылады. Бірақта иіндер барлық бөлінген және қосылған жағдайда, олар өздері көрсетілген обьектілерді сақтау керек.

CASE - құралын келесі қасиеттермен жіктеу:

- ұйымдық өнімнің нақты бағасын анықтау;
- тапсырыс берушіні ұстап тұру үшін бағаны анықтау;

Бірінші кезекте маңызды жұмыстарды анықтау және бағасы қымбат жұмыстарды табу қажет.

Бағалық талдау Design/IDE (Meta Software), BPWin (Logic Works)) тараған

әдістеме болып табылады, ол халықаралық корпорациялармен және мемлекеттік мемлекеттік ұйымдармен((Vantage Team Builder ((Cayene), Designer/2000(ORACLE), Silverrun (CSA), PRO-IV (McDonnell Douglas), (МакроПроджект) ұйымдағы нақты шығындарды іздеу үшін қолданылады.

CASE – құралдарына байланысты ақпараттар біздің ойымызша,кәдімгі бір бағдарламалаумен тығыз байланысты зат сияқты. Америкада, конкуренцияның күшті болуына байлансыты, CASE – құралдары бағдарламалық қамтаманы өңдеуші фирмаларды басып отыру үшін қолданылады. CASE – құралдарының дамуы обьектілі – бағытталған ПҚ өндеу технологияларына байланысты. Осы кезде обьектілік модельдеу технологиялары да пайда бола бастады Booch, OMT, UML, олар өз алдына бағдарламалаумен байланыстыруға мүлдем келмейді. Бүгінде алдынғы қатардағы CASE-жүйе болып Rational Rose корпорациясымен шығарылған Rational Software алынады. Rational Rose жүйесі Unified Modeling Language (UML) тілін қолдана отырып модульдер құру үшін пайдаланылады. Айталық, UML обьектілі – бағытталған стандартты тілі болып қалыптасуы Rational Software – дің арқасында деп айтсақ та артық емес, ол UML-ді қолданатын бағдарламалық өнімдерді шығарып қана қоймай, сонымен қатар UML тілінің спецификациясын құратын және жаңартатын CORBA таратылған есептеулермен байланысты технологиялар Object Management Group (OMG) – пен де жұмыс жасайды, және Rational компаниясында үш, UML тілін және обьектілі – бағытталған өңдеуді құрушылар жұмыс жасайды. Бұлар Гради Буч, Айвар Джекобсон және Джим Рамбаух.

CASE-технологиялары мен CASE-құралдарының пайда болуына бағдарламалау методологиялары облысындағы зерттеулер әсер етті. CASE-технологиялардың пайда болуына мынадай факторлар әсер етті:

- Модельдік және құрылымдық бағдарламалау концепцияларын білетін аналитиктер мен бағдарламаларды дайындау;
- Жобалау сатыларының көбісін автоматтандыру және графикалық құралдарды тиімді қолдануға мүмкіндік беретін компьютерлер өнімділігінің ендірілуі мен тұрақты өсуі;
- Жоба туралы қажетті ақпарат сақталған таратылған деректер қорын пайдалана отырып, жеке орындаушыларды бір жобалау процесіне біріктіретін желілік технологияларды ендіру.

CASE-құралдарын сапалы енгізу үшін мекемеде келесідей қасиеттер болу керек:

- Технология. Жаңа технологияларды игере білу қабілеттілігі;
- Мәдениеттілік. Өндірушілер мен қолданушылар арасындағы өзара байланыстар мен жаңа процестерді ендіруге дайын болуы;
- Басқару. Енгізу процестері мен сатыларына байланысты нақты басқару және ұйымдастырушылық қабілетінің болуы.

CASE-құралдарын жасаушы фирмалардың арасында IBM Rational Software Corp. (2003 жылдың тамызына дейін - Rational Software Corp.) компаниясы бағдарламалық жүйелерді талдау және жобалаудың обьектіге бағытталған технологияларын дамытуды қарастырды. Бұл компания визуалды

модельдеу тілі унификациясының инициаторы болып табылады. Бұл UML тілінің алғашқы версияларының пайда болуына әкелді. Осылайша бұл компания алғашқы болып инструменталды обьектіге бағытталған CASE-құралын жасап шығарды, онда UML визуалды модельдеу тілінің базалық нотациясы жасалды.

IBM Rational Rose CASE-құралы пайда болғаннан бері қаншама өзгерістерден өтті, қазіргі кезде ол бағдарламалық жүйелерді құру, жобалау архитектурасы, талдау, модельдеу үшін интеграцияланған инструментарий болып табылады. IBM Rational Rose UML тілі, бағдарламалық жүйелерді визуалдау және құрудың базалық технологиясы болды, ол осы инструментарийдің танымалдылығын және стратегиялық перспективтілігін анықтады.

IBM Rational Rose құралының бірнеше варианттары бар, олар бір-бірінен берілетін мүмкіндіктердің диапазонымен ерекшеленеді. Қазіргі кезде базалық құрал IBM Rational Rose Enterprise Edition болып табылады, оның мүмкіндіктері жоғарырақ болып келеді. Бұл CASE-құралының соңғы версиясы IBM Rational Rose 2003 (release 2003.06.00) программасы болып табылады, онда ақпараттық технологиялар облысындағы барлық мүмкіндіктер қарастырылған, бұл программаның функционалды мүмкіндіктері мыналар:

- MS Visual Studio 6 ортасымен интеграциялануы, ATL (Microsoft Active Template Library), Web-Класстар, DHTML және әр түрлі деректер қорына енуге мүмкіндік беретін хаттамаларды қолдану арқылы кодтардың тікелей және кері генерациясын, Visual Basic және Visual C++ диаграммаларын қолдау;
- EXE, DLL, TLB, OCX форматтары кітапханаларымен және орындалатын модульдермен жұмыс жасау (инжиниринг және реинжиниринг);
- MTS (Microsoft Transaction Server) және ADO (ActiveX Data Objects) технологияларын қолдау;
- CORBA және J2EE компоненттерін толық қолдау, оған қоса CBD (Component-Based Development) қосымшаларын компоненттік құру, IDL (Interface Definition Language) интерфейсті анықтау тілі және DDL (Data Definition Language) деректерді анықтау тілі технологияларын жүзеге асыру;
- Java-қосымшаларын құру ортасын қолдау.

IBM Rational Rose программасының жұмыс интерфейсі түрлі элементтерден тұрады, олардың негізгілері төмендегілер болып табылады:

- Негізгі мазмұны;
- Стандартты құрал-саймандар панелі;
- Арнайы құрал-саймандар панелі;
- Жоба браузері терезесі;
- Диаграмма суреттерінің облысы немесе диаграмма терезесі;
- Құжаттандыру терезесі;
- Журнал терезесі.

Rational Software Rational Rose 98 компаниясының CASE-жүйелері коммерциялық ПҚ құру үшін барлық жерлерде және әйгілі бағдарламалау тілдер Java, Cu++, Смолток, Ада, Visual Basic, Power Builder жәнен Forte қолданылады. Сонымен қатар, Rose 98 пакеті CORBA и Data Definition Language (DDL) қосымшалары үшін, Interface Definition Language (IDL) тілдеріндегі сипаттамаларды генерациялауға, деректер базасына қатынау қосымшалары соның ішінде Oracle 8-ге қатынауға мүмкіндік береді. Айталық, соль немесе басқа бағдарламалау тілін қолдауы Rational Rose 98 пакетінің қай редакциясы екендігіне байланысты.

Мысалы, қарапайым - Rose 98 Modeler Edition пакетіне қатты талаптар қоюға болмайды. Ал оның орнына Rose 98 Enterprise Edition барлық талаптарға сай деп айтсақ болады

Айталық, Rose жүйесі - визуальды модельдеу жүйесінде жақсы орын алады, оны қолдана отырып құрылып отырған қосымшаның құрылымын құруға, оның текстін генерациялауға және паралельді өңделіп отырған жүйенің құжаттарымен жұмыс жасауға болады. Rational Rose көмегімен com модульдегі кері талдау базасы негізінде жаңа модельдер құруға немесе қолданбалы программа текстін және кітапханалар класстарын анықтауға болады.

Rational Rose 98-дің артықшылықтары:

1. Қосымшаның өңделу циклын қысқарту.
2. Бағдарламалаушы жұмысының өнімділігін арттыру.
3. Бизнес және пайдаланушыларға байланысты тапсырыс берушілердің бағдарлама құрудағы сапалық бағасын жақсарту.Үлкен жобалар және жобалар тобын құра алу мүмкіндіктері.
4. Бұрын құрылған ПҚ қолдана алу және олардың құрылымы мен компоненттерін өзгерте алу мүмкіндіктері.
5. UML тілі әртүрлі бөлімдер мен өндірушілер арасындағы әмбебап "көпір".

Жүйенің кемшілігі, басқа да бөліктеп жинау жүйелері сияқты мұнда да көп керек емес бөліктер көп. Яғни Rational Rose жобаның базалық құрамдас бөлігін жасайды.

CASE-құралдарын жасаушы фирмалардың арасында IBM Rational Software Corp. компаниясы бағдарламалық жүйелерді талдау және жобалаудың обьектіге бағытталған технологияларын дамыту перспективаларын қарастырды. Бұл компания визуалды модельдеу тілі унификациясының инициаторы болып табылады, бұл UML тілінің алғашқы версияларының пайда болуына әкелді. Осылайша бұл компания алғашқы болып инструменталды обьектіге бағытталған CASE-құралын жасап шығарды, онда UML визуалды модельдеу тілінің базалық нотациясы жасалды. CASE-құралдар (Computer Aided Software/System Engineering компаниясынан) компьютерде кез келген жүйені жобалауға мүмкіндік береді. CASE-құралдары бизнес-процестерді, деректер қорын, бағдарламалық қамсыздандыру компоненттерін, қызметті және мекеме құрылымын жобалауға мүмкіндік береді және барлық қызмет түрлерінде қолдануға болады. CASE-құралдарын қолдану тиімділігі – жүйелерді

оңтайландыру, шығындарды азайту, тиімділікті жоғарылату, қателер ықтималдығын төмендетеді. CASE-технологиялары мен CASE-құралдарының пайда болуына бағдарламалау әдістемелері облысындағы зерттеулер әсер етті.

CASE-технологияларының пайда болуына мынадай факторлар әсер етті:

- Модельдік және құрылымдық бағдарламалау концепцияларын білетін аналитиктер мен бағдарламашыларды дайындау;
- Жобалау сатыларының көбісін автоматтандыру және графикалық құралдарды тиімді қолдануға мүмкіндік беретін компьютерлер өнімділігінің енгізілуі мен тұрақты өсуі;
- Жоба туралы қажетті ақпарат сақталған таратылған деректер қорын пайдалана отырып, жеке орындаушыларды бір жобалау процесіне біріктіретін желілік технологияларды ендіру.

CASE-технологиясы АЖ жобалау әдістемесі болып табылады, сонымен қатар пәндік аймақты көркем түрде жобалауға, осы модельді құру кезеңтарын талдауға мүмкіндік беретін инструменталды құралдар жиынтығы. Қазіргі таңдағы CASE-құралдарының көбісі құрылымдық (негізгі) және обьектіге бағытталған талдау мен жобалау методологияларына негізделген, онда спецификациялар диаграмма немесе сыртқы талаптарды сипаттауға арналған мәтіндер, жүйе модельдерінің арасындағы байланыс, бағдарламалық құралдар архитектурасы мен жүйенің әрекет ету динамикасы түрінде қолданылады.

CASE-құралдарын сапалы енгізу үшін мекемеде келесідей қасиеттер болу керек:

Технология. Жаңа технологияларды игере білу қабілеттілігі;

Мәдениеттілік. Өндірушілер мен қолданушылар арасындағы өзара байланыстар мен жаңа процестерді ендіруге дайын болуы;

Басқару. Енгізу процестері мен сатыларына байланысты нақты басқару және ұйымдастырушылық қабілетінің болуы.

CASE-құралдарын сәтті сапалы енгізудің тиімділігі мынадай:

- Бағдарламалық құралдарды құру және әрі қарай жұмыс жүргізу процестерін технологиялық қолдаудың жоғарғы деңгейі;
- Келесі факторларға оң әсер етуі: өнімділік, өнім сапасы, стандарттарға сәйкестік, құжаттандыру.

IBM Rational Rose CASE-құралы пайда болғаннан бері қаншама өзгерістерден өтті, қазіргі кезде ол бағдарламалық жүйелерді құру, жобалау архитектурасы, талдау, модельдеу үшін интеграцияланған құрылғы болып табылады. IBM Rational Rose UML тілі, бағдарламалық жүйелерді визуалдау және құрудың базалық технологиясы болды, ол осы инструментарийдің танымалдылығын және стратегиялық перспективтілігін анықтады.

Қорытындылай келе IBM Rational Rose CASE-құралы пайда болғаннан бері қаншама өзгерістерден өтті, қазіргі кезде ол бағдарламалық жүйелерді құру, жобалау архитектурасы, талдау, модельдеу үшін интеграцияланған инструментарий болып табылады. CASE-құралдары (Computer Aided Software/System Engineering компаниясынан) компьютерде кез келген жүйені

жобалауға мүмкіндік береді. CASE-құралдары бизнес-процестерді, деректер қорын, бағдарламалық қамсыздандыру компоненттерін, қызметті және мекеме құрылымын жобалауға мүмкіндік береді. Барлық қызмет түрлерінде қолдануға болады, CASE-құралдарын қолдану тиімділігі – жүйелерді оңтайландыру, шығындарды азайту, тиімділікті жоғарылату, қателер ықтималдығын төмендету.

CASE - құралы Silverrun американдық фирмасы Computer Systems Advisers, inc. Компьютер үшін бағдарлама - бұл бағдарламалау тілінде құрылған немесе жазылған тәртіптер тізімі. Бағдарламалау тілі берілген есепті шешу үшін және тиімді, жылдам, сапалы, үнемді жобалау мақсатында таңдалады.

Жүйеде дайын әдістеме бар: DATARUN(негізгі методология, Silverrun), Gane/Sarson, Yourdon/De Marco, Merise, Ward/Mellor, Information Engineering. Тіл деңгейі - бағдарламалау деңгейі. Төменгі деңгейлі бағдарламалау - бұл микропроцессор деректер форматын және командаларын кодтық немесе мнемоникалық формада (ассемблер) қолданатын бағдарламалау.

DATARUN әдістемесі

Дүние жүзінде электрондық әдістемелердің ішіндегі кең таралған түрлерінің бірі DATARUN әдістемесі болып табылады. DATARUN әдістемесіна сәйкес БҚӨЦ ISO 12207 стандартымен анықталатын негізгі процестерді орындау нәтижелерімен байланысатын кезеңдерге бөлінеді. Әрбір кезең келесі кезеңнің жұмыс жоспарымен аяқталуы керек.

Талаптарды құру мен жоспарлау кезеңі жобаның құны мен бастапқы бағалар көлемін анықтайтын әрекеттерді қамтиды. АЖ – ні құруға арналған талаптар мен экономикалық негіздеуді құру қажет. Сонымен қатар, жобаның іске асырылуының техникалық бағалануына негіз болатын мәліметтердің бастапқы концептуалды моделі мен функционалдық модельдерді (мекеменің бизнес - процестер модельдерін) құру керек. Осы кезеңнің негізгі нәтижелері ретінде мекеме қызметінің(процестер мен мекеме мәліметтерінің бастапқы модельдері), жүйеге қойылатын талаптардың модельдері мен бастапқы бизнес – жоспар болуы керек.

Жобалаудың концептуалды кезеңі бастапқы мәліметтердің детальді талдануы мен мәліметтердің концептуалды моделін анықтаудан басталады. Содан кейін жүйенің архитектурасы жобаланады. Жүйе архитектурасы концептуалды модельдің көз жетерлік ішкі модельдерге бөлінуін қамтиды. АЖ – ні пайдаланудың мүмкіндігі бағаланады және оларды түрлендірудің сәйкес әдісі таңдалады. Жоба құрылғаннан кейін бастапқы бизнес – жоспар анықталады. Осы кезеңнің қорытынды компоненттері болып мәліметтердің концептуалды моделі, жүйе архитектурасының моделі және анықталған бизнес – жоспар табылады.

Қосымшаларды арнайыландыру кезеңінде жобаны құру мен бөлшектендіру процесі жалғасады. Мәліметтердің концептуалды моделі мәліметтердің реляциялық моделіне түрленеді. Қосымшаның құрылымы мен экрандар, есептер мен пакеттік процестер түрінде көрсетілетін қажетті

интерфейстері анықталады. Мәліметтердің моделі әр кесте үшін бизнес – ережелермен және әдістермен анықталады. Осы кезең аяқталған кезде қосымшаларды жүзеге асыру әдісі туралы тиянақты шешім қабылданады. Кезеңнің нәтижелері бойынша АЖ архитектурасының, мәліметтердің, функциялардың, интерфейстердің (сыртқы жүйелер мен пайдаланушылармен бірге), өңделетін қосымшаларға арналған талаптардың, қосымшаларды интеграциялау талаптарының модельдерін қамтитын АЖ жобасы құрылуы керек. АЖ – ні құрудың ақырғы жоспары жасалуы қажет.

Әзірлеу, интеграциялау және тестілеу кезеңінде мәліметтердің жеке және кешендік тестері, тестілік қоры жасалуы керек. Жобаға сәйкес МҚ мен қосымшалардың әзірленуі, тестіленуі мен прототиптелінуі өткізіледі. БҚ – ның ағымдағы нұсқасының конфигурациясы сипатталады. Тестілеу нәтижесі негізінде МҚ мен қосымшалардың оңтайландырылуы өткізіледі. Қосымшалар жүйеге интеграцияланғаннан кейін қосымшалардың тестіленуі өткізіледі. Кезеңнің негізгі нәтижелері ретінде жүйедегі кешендік тестілеуден өткен дайын қосымшалар, БҚ конфигурациясының ағымдағы сипатталуы, сынау нәтижелері бойынша түзетілген жүйенің нұсқасы мен жүйені қолданбалы құжаттау болып табылады.

Енгізу кезеңіне МҚ мен қосымшаларды енгізу мен орнату әрекеттері кіреді. Кезеңнің нәтижелері ретінде жұмысқа дайын және тапсырыс берушінің бағдарламалық – аппараттық платформасына ауыстырылған жүйенің нұсқасы, сүйемелдеу құжатталуы мен жұмыс нәтижелері бойынша қабылдау сынақтарының акті болуы қажет.

Сүйемелдеу мен даму кезеңдерінде қателерді тіркеуге, табу мен жоюға, өзгертулерді енгізу мен тестілеуге, БҚ – ның жаңа нұсқаларын айналымдау мен таратуға, қосымшаларды жаңа платформаға ауыстыру және жүйені көлемдеуге байланысты процестер мен операцияларды ұйымдастырады. Даму кезеңі әзірлеу кезеңінің қайталау итерациясы болып табылады.

DATARUN әдістемесі 2 модельге негізделген:

- Мекеме моделі;
- АЖ моделі.

DATARUN әдістемесі мекеме қызметін сипаттауды жүйелі түрде көрсетуге негізделген. Модельдерді жасау процестерді сипаттаудан басталады. Процестерден бастапқы мәліметтер алынады. Бастапқы мәліметтер мекеменің өнімдерін немесе қызметтерін, орындалатын операциялар (транзакциялар) мен пайдаланатын ресурстарын сипаттайды. Бастапқы мәліметтерге сыртқы және ішкі мәндер, яғни, клиенттер немесе агенттіктер жатады. Сонымен қатар, бастапқы мәліметтер ретінде шешімдерді қабылдау нәтижесінде алынған мәліметтер алынады. Мысалы, жұмыс графигі, тауарлар бағасы.

DATARUN әдістемесінің негізгі принципі - бастапқы мәліметтердің АЖ архитектурасын жобалау үшін негіз болуы.Бизнестің көрінісін анықтайтын негізгі операциялармен тығыз байланысқан бастапқы мәліметтерге негізделген болса.,АЖ архитектурасы тұрақты болады.

Кез келген АЖ процессорлармен орындалатын және МҚ – мен өзара

әрекеттесетін модульдер жиынын көрсетеді. МҚ мен процессорлар орталықтандырылған немесе үлестірілген болуы мүмкін. Барлық транзакциялар бір немесе бірнеше МҚ – мен өзара әрекеттесетін интерфейстер модулі немесе объектілер арқылы жүзеге асады.

DATARUN әдістемесінің 2 мақсаты болады:

- АЖ құрылатын тұрақты құрылымды анықтау. Мұндай құрылым ретінде мекеменің фундаментальді процестерін көрсететін бастапқы мәліметтерден алынатын **мәліметтер моделі** болып табылады;
- Мәліметтер моделі негізінде АЖ – ні жобалау.

Мәліметтер моделінің негізінде құрастырылған объектілер клиент-сервер ортасының серверлерінде орналасатын МҚ – ның объектілері болып табылады. Компьютерлік жүйе архитектурасында анықталған интерфейс объектілері, әдетте, клиенттік бөлігінде орналасады. МҚ – ның пайдаланатын объектілері мен интерфейстің әртүрлі объектілерін спецификациялау үшін негіз болып табылатын мәліметтер моделі АЖ – нің сүйемелденуін қамтамасыз етеді. Silverrun DATARUN әдістемесіна сәйкес жобалау жұмыстарының өткізілуінің автоматтандырылуын қамтамасыз етеді. Осы құралдармен көрсетілетін жобалау ортасы жоба басқарушысына жұмыс әрекетін бақылау, жұмыстың орындалуын қадағалау, жұмыс тәртібінен ауытқуларды дер кезінде байқау мүмкіндігін береді. Жобаның кез келген қатысушысы осы ортаға қосылғаннан кейін өзіне берілген тапсырма мен орындау мерзімін біле алады, тапсырманы орындау техникасын детальді зерттей алады.

АЖ бизнес-процестер моделдерінен басталып және осы процестерді автоматтандыратын бағдарламалау моделімен аяқталатын моделдер қатарынан тізбектеліп құрылады.

BPM (Business Process Model) – бизнес-процестер моделі

PDS (Primary Data Structure) – алғашқы мәліметтердің құрылымы

CDM (Conceptual Data Model) – мәліметтердің концептуалды моделі

SPM (System Process Model) – жүйе процестерінің моделі

ISA (Information System Architecture) – АЖ- нің архитектурасы

ADM (Application Data Model) –

IPM (Interface Presentation Model) – интерфейсті көрсету моделі

ISM (Interface Specification Model) – интерфейсті спецификациялау моделі.

Құрылып жатқан АЖ мекеме орындайтын фунуцияларға негізделуі керек. Сондықтан алғашқы құрылатын модель – бұл Silverrun BPM модулінде құрылымы іске асатын бизнес- процестерінің моделі. Бұл модель үшін арнайы BPM нотациясы қолданылады. Бизнес – фунуцияларды талдау мен спецификациялау процесінде мәліметтер құрылымы ретінде құжатталатын негізгі ақпараттық объектілер анықталады. Құрылымды құру үшін мекемеде пайдаланылатын құжаттар, инструкциялар, өндірістік операцияларды сипаттау пайдаланылады. Қажет емес ақпараттарды нормализациялау мен жою Silverrun ERX модулінде мәліметтердің концептуалды моделін құруда жүзеге асырылады. Бизнес-процестердің моделін құрғаннан кейін ақпарат жобаның

репозиториінде сақталады.

Мекеме жұмысын зерттеу барысында алғашқы мәліметтер құрылымы анықталады және құжатталады. Осы құрылымдар мекемедегі құжаттарды, хаттарды, мәліметтерді сипаттау арқылы BPM модулінің репозиторийіне енгізіледі. Бизнес-процестер моделінде мәліметтердің алғашқы құрылымдары ақпараттың ағыны мен сақталуымен байланысты болады.

Мәліметтердің алғашқы құрылымдарының негізінде Silverrun ERX модулінде мәліметтердің концептуалды моделі (ER - моделі) құрылады. Концептуалды модель мәліметтердің алғашқы құрылымдарынан қажет емес ақпаратты жоюмен, нормализация мен түсініктер атауларының стандартизациясымен ерекшеленеді. Бұл операциялар ERX модулінде кірістірілген зерттеуші жүйе арқылы жүзеге асады. Мәліметтердің концептуалды моделінің мақсаты – пайдаланылатын ақпаратты МҚ – да ешбір детальсіз сипаттау (нормализацияланған түрде).

Бизнес-процестер моделі мен мәліметтердің концептуалды моделінің негізінде АЖ – нің архитектурасы жобаланады. Жүйеге кіретін қосымшалар анықталады. АЖ архитектурасы арнайы ISA нотациясын қолдану арқылы Silverrun BPM модулінде құрылады. Осы модельдің негізгі мазмұны – жүйенің құрылымдық компоненттері мен олардың өзара новигациясы. Мәліметтердің концептуалды моделі қосымшаларға сәйкес болатын бөліктерге бөлінеді.

Қосымшаны құру алдында МҚ –ның корпоративті құрылымы жобалануы керек. DATARUN реляциялық модельге негізделген мәліметтер қорын пайдалануды ұсынады. Мәліметтердің концептуалды моделі нормализациядан кейін арнайы ERX- RDM арқылы Silverrun RDM реляциялық модельдеу модуліне ауыстырылады. Модельдің ERX форматынан RDM форматына түрленуі пайдаланушының әрекетінсіз автоматты түрде іске асады. Форматтардың түрленуінен кейін МҚ – ның реляциялық моделі пайда болады. Бұл модель Silverrun RDM модулінде физикалық іске асырылуының анықталуымен (МҚБЖ мәліметтері типтерінің, кілттердің, индекстердің, триггерлердің) нақтыланады. Мәліметтерді өңдеу ережелерін МҚБЖ бағдарламалау тілдерінде ғана емес, сонымен қатар, декларативті түрде де көрсетуге болады.

Жүйелік процестерінің модельдері арқылы әр қосымшаның әрекеті неғұрлым дәлірек құжатталады . BPM модулінде бизнес – процестердің қалай жүзеге асатынын анықтайтын жүйелік процестер моделі құрылады. Бұл модель әр қосымша үшін жеке құрылады және қосымшаның мәліметтер моделімен тығыз байланысты болады.

Қосымша интерфейстік объектілерден (экрандық формалардан, есептерден, ақпаратты өңдеу процедураларынан) тұрады. Жүйенің әр интерфейсі (экрандық форма, есеп, мәліметтерді өңдеу процедуралары) мәліметтер қорының ішкі жиынымен байланысты болады.

3 Жобалау әдістері және бағдарламалардың өмірлік циклы

Мақсаты: Жобалау технологиялары мен әдістерін қарастыру. Бағдарламалық қамтамасыз ету процессінің өмірлік циклдері.

Ақпараттық жүйені жобалау – ұзақ мерзімді және динамикалық үрдіс. Қазіргі уақытта қолданылатын жобалау технологиялары жүйені кезеңдік жасауды ұсынады. АЖӨЦ дегеніміз - ААЖ–ні (автоматтандырылған ақпараттық жүйе) жасау қажеттілігінен бастап оның келесі сатысына көшу уақытымен бітетін кезең.

АЖӨЦ (өмірлік цикл) келесі кезеңдерден тұрады:

1. Жоба алдындағы кезең;
2. Жобаны әзірлеу кезеңі;
3. Жобаны пайдалануға беру кезеңі;
4. Жаңа сатыға көшу кезеңі

1. Жоба алдындағы кезеңде екі құжат дайындалады:

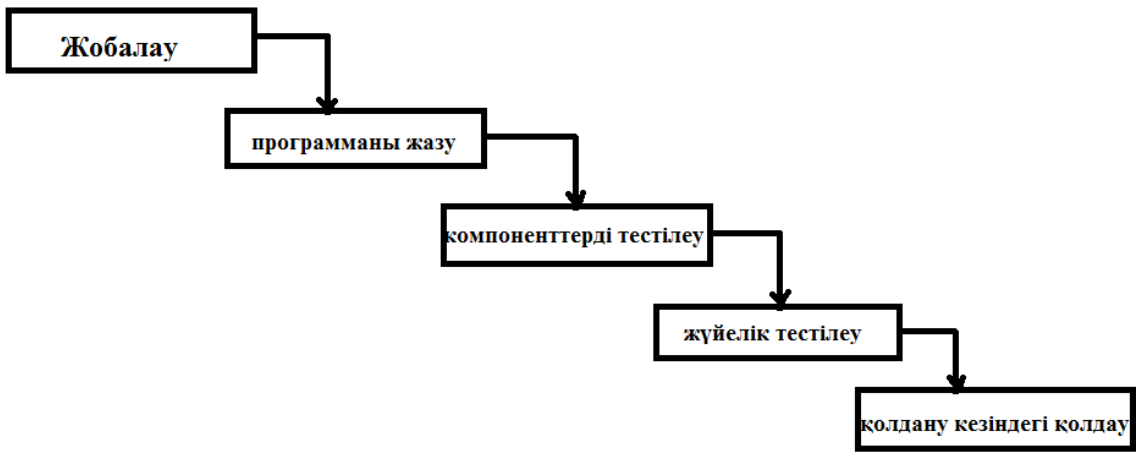
а) Техникалық - экономикалық тұжырымдама (ТЭТ)

б) Техникалық тапсырма (ТТ)

3.1 БҚ өмірлік циклының модельдері

АЖ өмірлік циклінің моделі – қолданбалы бағдарламалық қамсыздандыруды жасау, жұмыс жасау және сүйемелдеу үрдісінен, жұмысынан және тапсырмасынан тұратын, жүйе өмірін оған қойылатын талаптардан бастап оның қолданылуы тоқтағанға дейін қамтитын құрылым. Қазіргі кезде оның келесі модельдері белгілі және қолданылуда: каскадты модель, деңгейлі модель.

Каскадты модельдің негізгі сипаттамасы АЖ-ні әзірлеуді кезеңдерге бөлу, бір кезеңнен екінші кезеңге өту ағымдағы кезеңдегі жұмыс толығымен аяқталғанда ғана жүреді. Әрбір кезең АЖ-ні әзірлеу әзірлеушілердің басқа командасына тапсырылатындай құжаттардың толық жиынымен аяқталады.



1 сурет – АЖ өмірлік циклінің моделі

Каскадты модельде келесі деңгейге өту алдыңғы кезеңдегі барлық жұмыстардың аяқталуын білдіреді.

Әр кезең біткен кезде келесі кезеңде жұмысты жалғастыруға жетерліктей мөлшерде құжаттар жасалып шығарылады. Сонымен қатар жұмыстың кезеңдрі логикалық тізбекте орындалады, ал бұл жұмысқа кететін барлық уақытты және шығынды алдын ала жоспарлауға ыңғайлы. Бұл модельдің кемшілігі бар. Оның кемшілігі – бір кезең толық орындалып біткеннен кейін, келесі кезеңді орындау барысында қателіктер, жетіспеушіліктер пайда болуы мүмкін. Яғни алдыңғы кезеңдерге қайтадан баруға тура келетін жағдайлар көп кездеседі. Нәтижесінде жұмыстың орындалу мерзімі ұзаққа созылып кетуі мүмкін.

Деңгейлі моделі. Бұл модельде алдыңғы кезеңдерге қайтадан барып, қажет болса түзетулер, толықтырулар енгізуге болады. Каскадты модельмен салыстырғанда деңгейлі түзетулер бағдарламалық қамтаманы құру процесінің жұмыс сыйымдылығын азайтуға мүмкіндік береді. Әр кезеңнің өмір ұзақтығы бүкіл жұмыстың орындалу уақытына теңестіріледі.

2 сурет – Деңгейлі модель

Өмірлік циклдің қайта келу (Спиральді) моделі

Қайта айналып келу моделінде өмірлік циклдың бастапқы кезеңдерінде талдау және жобалау жүзеге асырылады.

3 сурет – Спиральді модель

Бұл модельде әсіресе бағдарламалық өнімді жасаудың бастапқы кезеңдеріне көп көңіл бөлінеді. Олар – стратегияны құрастыру, талдау және жобалау. Онда қандай да бір техникалық шешімдердің жүзеге асырылатындығы немесе жүзеге асырылмайтындығы прототиптер жасау арқылы тексеріледі. Спиральдің әр бөлігі бағдарламалық өнімнің бір компонентін немесе бір версиясын жасауды білдіреді. Сол арқылы бағдарламалық өнімнің сапасы тексеріліп, келесі бөліктегі жұмыстар жоспарланады.

Спиралді модель бір бөліктен екінші бөлікке жұмысты толық аяқтамай-ақ өте беруге мүмкіндік береді. Бұл жерде туындайтын мәселе –келесі бөлікке қашан өту керек деген сұрақ. Сол үшін әр бөлікке уақытқа шектеу қойылады. Мысалы, спиральдің өңдеу деп аталатын әр бөлігіне 2 күннен уақыт беріледі немесе әр тестілеу бөлігіне 1 күннен беріледі. Осылайша спиралдің әр бөлігінде жұмыстың кішкене бір бөлігі жасалып отырады да, соңында дайын өнім шығады. Бұл модель жұмыстың нәтижесіндегі қателіктерді айтарлықтай азайтады. Спиральді модель көбінесе үлкен, күрделі әрі қымбат болып келетін жобаларға арналған.

АЖ жобасының негізін құраушылары:

- Жобалауд әдістері.
- Жобалау технологиялары.
- Жобалаудың аспаптық құралдары (CASE құралдары)
- АЖ-ні жобалаудың әдістері және технологиялары.

Жобалау әдістері мен инструменталды құралдары (CASE-құралдары) АЖ-нің жобасын орындаудың формальданған пәнінің орталық бөлігін құрайды.

АЖ-ні жобалаудың әдісі дегеніміз нақты анықталған нотацияларды қолданып әзірленетін жүйенің әртүрлі аспектілерін сипаттайтын модельдерді жасау процестерінің жиынтығы.

Әдіс мынандай құраушылардың жиынтығы ретінде анықталады.

1) Концепциялар және теориялық негіздер. Бұндай негіздер ретінде құрылымды немесе объектіге бағытталған тәсіл қарастырылады;

2) Жобаланатын жүйе тәртібі динамикасының және статикалық құрылымының модельдерін тұрғызуда қалданылатын нотациялар; әдетте нотациялар ретінде графиктік диаграммалар қолданылады, себебі олар неғырлым көрнекті және қабылдауға қарапайым.

(Мысалы: мәліметтер орнының диаграммалары, “мән-байланыс” диаграммасы-құрылымдық тәсілде, ал 0-ге бағытталған тәсілде қолдану қолдану нұсқаларының диаграммалары, кластар диаграммасы).

3) Әдістің практикалық қолданысын анықтайтын процедуралар: модельдерді тұрғызу тізбектілігі және ережелері, нәтижелерді бағалауда қолданылатын критерийлер.

Жобалау технологиясы төмендегі құраушылардың жиынтығы ретінде анықталады:

- Жобалаудың технологиялық амалдарының тізбегін анықтайтын қадамдық процедуралар;
- Технологиялық амалдарыдың орындалу нәтижесін бағалауда

қолданылатын критерийлер мен ережелер;
- Жобаланатын жүйені сипаттау үшін қолданылатын графикалық және мәтіндік құралдар;
- Ақпараттық жүйелерді жобалаудың модельдері мен әдістері

Қолданбалы бағдарлама пакеті базасында компьютерлік ақпаратты және жүйені басқарушыны құрастыру. БАӨКЖ–ні құрудың ірі тобырлар: жоба алдындағы тексеру, ТЭН,есептеу, техникалық тапсырманы өңдеу; жобаларды өңдеу – техникалық жобалау, жұмыс жобалауы; қолдануға енгізу – жұмыстың жобасын өңдеу, тәжірибелік қолдану, өндірістік қолдану. БАӨЖК–нің күтілген экономикалық тиімділігін есептеу.

БАӨЖК–ні өңдеудің индустиальды әдістері: объектілік, элементті, ішкіжүйелік және модельді.

ОЦ-нің моделі дегеніміз ОЦ барысында орындалатын міндеттер мен әрекеттердің, процестердің орындалуын және өзара байланысының тізбегін анықтайтын құрылым. ОЦ-нің моделі АЖ-нің ерекшелігіне байланысты.

Қазіргі кезде ОЦ-нің екі моделі кеңінен тараған:

1) каскадты модель(70-85ж)

2) спиральді модель(86-90ж)

Каскадты модельдің негізгі сипаттамасы АЖ-ні әзірлеуді кезеңдерге бөлу, бір кезеңнен екінші кезеңге өту ағымдағы кезеңдегі жұмыс толығымен аяқталғанда ғана жүреді. Әрбір кезең АЖ-ні әзірлеу әзірлеушілердің басқа командасына тапсырылатындай құжаттардың толық жинағымен аяқталады.

Каскадты модельдің жағымды жақтары: әрбір кезеңде аяқталған жобалау құжаттарының пакеті қалыптасады.

Каскадты тәсіл АЖ-ні әзірленудің басында әзірленушілерге барлық талапларды толық және дәл қоятындай АЖ-ні тұрғызуда өте ыңғайлы.

Негізгі кемшілігі:
- Нәтижелерді алудың көп кешіктірілуі
- Корпоративті ақпараттық жүйелерді жобалаудың технологиясы
- Мультикомпонентті ақпараттық және басқарушы жүйелерді жобалау әдістері.
- «Дуализм» және көпкомпонентті принциптері.

Өндірістік кәсіпорындардың корпоративті ақпараттық жүйелері. Жаңа автоматтандырылған ақпараттық жүйелерді жобалау кезеңдері: Бизнес–модель жасау және талдау. Бизнес–модельде формализациялау, бизнес–процестердің логикалық моделін жасау. Лингвистикалық қамтаманы таңдау, бағдарламалық қамтама жасау. ААЖ–ні тестілеу және іске асыру. Жүйелерді қолдану, іске қосу және бақылау.

Соңғы уақыттарда мекемелерде корпоротивті ақпараттық жүйелерді тұрғызу маңыздылығы қазіргі таңдағы шарттарда бизнесті басқарудың қажетті құралы ретінде қарастырылып отыр. Корпоративті ақпараттық жүйелер – бағдарламалық қамтамасыздандыру орнатылған және бапталған арнайы

бағдарламалық өнім және есептегіш аппараты платформаның жиынтығы.

Корпоративті ақпараттық жүйелерге маңызды әсер ететін 3 факторды ерекше атауға болады;

- мекемені (кәсіпорынды) басқарудың әдістемесін дамыту;
- компьютерлік жүйелердің жалпы мүмкіндіктері мен өнімділігін дамыту;
- ақпараттық жүйелер элементтерінің техникалық және бағдарламалық жүзеге асырылу тәсілдерін дамыту;

Әр факторды жекелей қарастырайық:

1) Мекемені басқарудың теориясы өте кең пән ретінде қарастырылады, сондықтан үнемі оқып үйрену мен дамытуды талап етеді. Бәсекелестіктің өсуіне байланысты компания жетекшілеріне нарықты өздерінің бар болуын сақтап қалатын жаңа әдістерді іздеуге мәжбүрлейді. Сол себептен қазіргі заманғы ақпараттық жүйелер менеджмент теориясы мен практикасындағы жаңа кірістірулермен қатар жүреді;

2) Компьютерлік жүйелердің қуаттылығы мен өнімділігі прогресі, желілік технологиялар мен мәліметтерді жіберу жүйесінің дамуы, компютерлік техниканың әртүрлі жабдықтармен интеграциясының кең мүмкіндіктері қпараттық жүйелердің өнімділігі мен қызметін үнемі дамытып отыруды талап етеді.

3) Ақпараттық жүйелердің аппараттық бөлігінің дамуы мен бірге соңғы жылдары ақпараттық жүйелердің бағдарламалық – технологиялық жүзеге асырылуының жаңа, неғұрлым ыңғайлы және әмбебап әдістерін іздеу тұрақты жүріп жатыр.

Соңғы уақытта ақпараттық жүйелердің дамуына үлкен әсер еткен маңызды жаңалықтар мыналар болды:

1) Бағдарламалаудың жаңа тәсілі, яғни объектілі – бағдарлы бағдарламалау тілінің пайда болуы.

2) Желілік технологиялардың дамуына байланысты локальді ақпараттық жүйелер клиент – серверлік және көп деңгейлі ақпараттық жүйелермен ауыстырылуда.

3) Интернеттің дамуы қашықтықта орналасқан бөлімшелермен жұмыс жасау мүмкіндігін берді. Интернет арқылы сатып алушыларға қызмет көрсете отырып, электронды комерцияның келешегін ашып берді. Мекеменің ішкі желісінде интернет технологияларында, яғни интернет - технологияларды қолдану жетістіктерін берді.

Корпоративті ақпараттық жүйелердің негізгі құраушылары:

Мекеменің компьютерлік инфрақұрылымы. Ол желілік, телекомуникациялық, бағдарламалық, ақпараттық және ұйымдастырушылық инфрақұрылымдар жиынтығы.

Мекеменің компьютерлік инфрақұрылымын корпоративті желі деп те атайды.

Ақпараттық жүйені жобалауды тиімді әдістер мен жабдықтарды қолдану сапалы жүйені құруға мүмкіндік береді және шығындарды аз жұмсауға болады.

Ақпараттық жүйені жобалау әдістері бұл жобалаудың сәйкес жабдығын қолдану арқылы ақпараттық жүйені құрудың әдістері.

Жобалау жабдықтарына мыналар жатады:

1) типтік жобалау;

2) қолданбалы бағдарламалар пакеті;

3) ақпараттық жүйенің типтік жобасы;

4) жобалаудың құрал'жабдықтары.

АЖ жобалаудың құрылымдық әдістемелері: Апараттық жүйелердің жобалаудың құрылымдық әдістемелері – АЖ мемлекеттік стандартпен қарастырылған жобасы, құжаттаманы сипаттау мен жинақтау ережесінің белгілері аясындағы құру процесіндегі түсінігінде, яғни архитектураның құрылатын жүйенің функционалдық құрылымы мен алгоритмінің бір мағыналық түсінігінің сипаттамасы. Осы тұрғыдан бастап жобалау АЖ тіршілік циклінің түрлі кезеңдерінде жобалық шешімдерінің дәйекті нысандалуына алып келеді: техникалық және жұмыс жобалауының талаптарын жоспарлау мен талдау, АЖ енгізу және пайдалану. Ақпараттық жүйелерді жобалаудың құрылымдық әдістері – бұл жобалаудың сәйкес жабдығын қолдану арқылы ақпараттық жүйелерді құру әдістері. Ақпараттық жүйелерді жобалауда тиімді әдістер мен жабдықтарды қолдану сапалы жүйені құруға мүмкіндік береді. Жобалау әдістері 3-ке бөлінеді:

1. Ерекше

2. Типтік

3. Автоматтандырылған.

Ақпараттық жүйелерді жобалау обьектілері жеке элементтер немесе олардың функционалдық және қамтамасыз ететін бөліктерінің кешені болып табылады.Сонымен дәстүрлі декомпозицияға сәйкес міндеттер, міндеттердің кешенімен басқару функциясы функционалдық элементтері болады. АЖ қамтамасыз ету бөлігінің элементтері мен олардың ақпараттық бағдарламалық жүйені техникалық қамтамасыз ету кешендері жобалау обьектісі болады. Қазіргі заманғы ақпараттық технологиялар әдетте әзірлеу процестерінде өзгертілетін, іріктеуі болжалды пайдаланушылар тарапынан қойылатын талаптардың негізінде іске асырылатын АЖ-ны жүзеге асыру әдістерінің ауқымды жиынтығын білдіреді. АЖ жобасын нақты бағдарламалық-техникалық ортада АЖ-ды құру мен пайдалану жөнінде жобалау шешімдерінің сипаттамасы берілген жобалық-конструкциялық және технологиялық құжаттама деп түсінеміз.

Жобалаудың мақсаты болып АЖ-ны тиімді жобалау мен өз мамандық есептерін және орындау мен басқарушылық шешімдерді қабылдау үшін ЭЕМ-нің нақты экономикалық обьектісі мен комуникация құралдарын дамыту ортасында қолданылатын мамандармен автоматтандырылған ақпараттық технологиялар арасында өзара тығыз қызметті қамтамасыз ету болып табылады. Ақпараттық жүйенің құрылымы – бұл ақпараттық жүйенің элементтері мен ішкі жүйелерінің өзара байланыстарының ішкі кеңістіктегі уақытша тұрақты бір тәртіппен құрылуының арқасында сол ішкі жүйелердің

функционалдық міндетінің анықтала түсуі және олардың сыртқы ортамен байланысы. Ақпараттық жүйелер құрылымы бойынша функционалды және жабдықтаушы бөлімнен тұрады.

Жобалаудың логикалық фазасы – бағдарламалық өнімнің алдын-ала өңдеу фазасы. «Менде алты құл бар. Олар ақылды, талпынғыш. Менің білетінімнің бәрі, солардың арқасы. Олардың аттары: қалай және неге, кім, не, қашан және қайда» - логикалық модел құру үшін керекті сұрақтар құрамының парадигмасы. Пәндік аймақтың талдануының және сипаттамасының - кезеңдері, алгоритмдер мен деректер моделінің логикалық құрылуы, олардың байланысы және пайдаланушыға берілуі. Стандарттар. Техникалық тапсырма (ТТ) – өңдеуге берілетін талаптардың нәтижесі. ТТ құрамы. Өңдеудің ашылған әдістемесі, және олардың қазіргі заманғы құралдарда қолданылуы мен элементтері. Өңдеу кезеңдерін алдын – ала спецификациялау және оның ТТ әсер етуі. Аспаптар – мүмкіндіктері, технологиялары, жұмыс әдістемесі. Құрылымдық, модульдік және обьектілік ұстанымдар – айырмашылықтары мен ұқсастықтары. Итерация, ұйымдастыру және жоғарыдан – төмен немесе төменнен – жоғары жобалау.

Каскадты моделдің проблемаларын шешу мақсатында АЖ-нің ОЦ-нің спиральді моделі ұсынады. Мұнда талдау және жобалау кезеңдеріне басты назар аударылған. Бұл кезеңдерде техникалық шешімдердің жүзеге асырылуы прототиптерді жасау жолымен тексеріледі. Спиральдің әрбір ұшы АЖ-ні жасаудың үзіндісіне немесе нұсқасына сәйкес келеді, сонда жобаның мақсаттары мен мінездемелері нақтыланады жіне сапасын анықтай отырып, спиральдің келесі ұшының жұмысы жоспарланады. Осылайша жобаның детальдары тереңдетіліп, нақтыланады, нәтижесінде жүзеге асырылатын негіздемесі бар нұсқа таңдалынады.

Итерациялық әзірлеу жүйені жасаудың объективті спиральді циклды бейнелейді.

Әрбір кезеңнің жұмысының толық аяқталмауы оны күтпей-ақ келесі кезеңге өтуге мүмкіндік береді, яғни итерациялық тәсілде жетпей тұрған жұмысты келесі итерацияда орындауға болады. Басты міндеті – пайдаланушыларға жұмыс жасауға қабілетті өнімді тезірек көрсету, ұсыну болып табылады.

Спиральді циклдің негізгі проблемасы – келесі кезеңге өту сәтін анықтау. Оны шешу үшін ОЦ кезеңдерінің әрбіреуіне уақытша шектеу енгізу керек. Кезеңнен кезеңге өту жоспарға сәйкес жүзеге асырылады, тіпті жоспарланған барлық жұмыс аяқталмаса да жоспар алдындағы жобалардан және әзірлеушілердің жеке тәжірибесінен алынған статистикалық мәліметтер негізінде құрылады.

Логикалық жобалау – ER-диаграмма негізінде реляциялық жүйелердің мүмкіндігі ескерілген мәліметтердің логикалық моделі, мәліметтер қоры кестелері арасындағы байланыс түрлерін анықтау. ER-диаграмманы тұрғызу үшін заттық аумақтың объектілерін, атрибуттық құрамын анықтау керек. Ғылыми–зерттеу бөлімінің деректер қорының логикалық моделі ERwin-да ортасында жасалып араларына байланыстар жүргізілді .

Физикалық жобалау кезеңі ДҚ логикалық құрылымының және деректерді аса тиімді орналастыру мақсатымен сақтаудың физикалық ортасының байланысымен қорытындыланады, яғни сақтау құрылымында ДҚ логикалық құрылымның бейнесі.Жады кеңістігінде сақталатын деректерді орналастыру, «физикалық» ДҚ әртүрлі компоненттеріне қатынаудың тиімді әдістерін таңдау мәселелері шешіледі. Осы деңгейдің нәтижелері деректерді анықтау тілінде сақтау сызбасы түрінде құжатталады. Осы деңгейге қабылдаған шешім жүйенің өнімділігіне анықталған ықпал көрсетеді.

RAD әдіснамасы

ОЦ – нің спиральді моделінің шеңберінде АЖ-ні әзірлеудің мүмкін тәсілдерінің бірі болып соңғы уақыттарда кеңінен таралған.

RAD (Rapid Application Development) – қосымшаларды жылдам жасаудың әдіснамасы болып табылады.

Бұл термин бойынша 3 элементті құрайтын АЖ – ні әзірлеудің процесі түсіндіріледі:

- бағдарламалаушылардың шағын командасы (8-10 адам);
- қысқа, бірақ тиянақты атқарылатын ойында график (3 ай);
- қайталанатын цикл.

RAD әдіснамасы бойынша АЖ-нің ОЦ 4 кезеңнен тұрады:

1) талаптарды талдау және жоспарлау кезеңі;

2) жобалау кезеңі;

3) жобаны жүзеге асыру кезеңі;

4) жобаны енгізу кезеңі.

1-кезең талаптарды талдау және жоспарлау кезеңінде жүйені пайдаланушылар жүйені орындайтын функцияларды анықтайды, олардың ішінен алғашқы болып өңдеуді талап ететін неғұрлым арнайы функциаларды ерекшелейді, ақпараттық қажеттіліктерді сипаттайды. Жоба масштабы шектеледі, келесі кезеңдердің әрбіреуі үшін уақыт шектеулі анықталады. Осы кезеңнің нәтижесінде болашақ АЖ функцияларының тізімі мен приоритеттілігі, қызметтік және ақпараттық модельдері нақтыланады.

«Жобалау» кезеңінде пайдаланушылардың бөлігі әзірлеуші-мамандардың жетекшілігімен жүйенің техникалық жобалауына қатысады. Қосымшалардың (бағдарламалардың) жұмыс істейтін прототиптерін жылдам алу үшін сәйкес инструменталды құралдар (CASE-құралдар) қолданылады. Пайдаланушылар әзірлеушілермен тікелей байланыс жасай отырып, алдыңғы кезеңде анықталмаған талаптарды (жүйеге қатысады) нақтылайды, толықтырады. Осы кезеңде мынандай әрекеттер орындалады:

1) жүйенің процестері нақты қарастырылады;

2) қажетті болса әр қарапайым процесс үшін жекелеме прототиптер жасалады: экрандық форма, диалог, анық емес немесе бір мәнді еместерді жоятын есептер.

3) мәліметтердің шектеу талаптары анықталады.

Осыдан кейін әзірленетін жүйенің функционалды нүктелерінің (function point) саны бағаланады және АЖ-ні ішкі жүйелерге бөлу шешімі қабылданады (RAD-жоба уақытында (3 айға дейін) әзірлеуші топтың жүзеге асыра алатын

ішкі жүйелеріне) функционалды нүкте деп әзірленетін жүйенің мына әлементтерінің түсіндіріледі:

1. қосымшаның кіріс элементі (кіріс құжаты немесе экрандық форма)
2. қосымшаның шығыс элементі (шығыс құжаты немесе экрандық форма)
3. сұраныс (”сұрақ/жауап”жұбы)
4. логикалық файл, яғни қосымшаның ішінде қолданылатын мәліметтер жазуларының жиынтығы.
6. Қосымшаның интерфейсі, яғни басқа қосымшаға берілетін немесе одан алынатын мәліметтер жазбаларының жиынтығы.

3.2 CASE – құралдарын енгізу

CASE – құралдары. Жалпы сипаттамасы және жіктелуі.

CASE – құралдарына АЖ-нің өмірлік циклінің процестерінің жиынтығын автоматтандыратын бағдарламалық құрал жатады және мынандай негізгі мінездемелік ерекшеліктерімен сипатталады:

1. Әзірлеушімен ыңғайлы интерфейсті қамтамасыз ететін және оның шығармашылық мүмкіндіктерін дамытатын АЖ-ні сипаттау және құжаттау үшін қуатты график құралдары;
2. АЖ-ні әзірлеудің үрдісін қамтамасыз ететін CASE-құралдарының жекелеме компоненттерінің интеграциясы.
3. Жобалық метамәліметтердің (репозитория) ұйымдастырған қоймасын арнайы қолдану.

Интеграцияланған CASE-құралы мынадай компоненттерден тұрады:

1. CASE-құралдына негіз болатын репозитория. Ол жоба версияларын сақтауды және оның жекелеме компоненттерін, топтық әзірлеуде әртүрлі әзірленушіден түскен ақпаратты синхронизациялауды және қарама-қайшы еместігіне бақылауды қамтамасыз етуі керек;

2. АЖ-нің моделін құрайтын иерархиялық байланысқан диаграммаларды (абциссаны DFD, DRD...) жасауды және редакторлауды қамтамасыз ететін талдау мен жобалаудың графиктік құралдары;

3. 4GL тілдерін және кодтар генераторларын қамтитын қосымшаларды әзірлеудің құралдары;

4. Конфигурациялық басқару құралдары;

5. Құжаттау құралдары;

6. Тестілеу құралдары;

7. Жобаны басқару құралдары;

8. Реинжиниринг құралдары;

Қазіргі барлық CASE құралдар типі және категориясы бойынша жіктелуі.

Типі бойынша жіктелу CASE-құралдардың ОЦ-дің әрбір үрдістерінің функционалдық бағытын көрсетеді.

Категориясы бойынша жіктелу орындалатын функциялардың интеграциялану дәрежесін анықтайды және үлкен емес автономды есептерді

шешетін жекелеме локальді құралдарды, АЖ-нің ОЦ-інің кезеңдерінің мейлінше көбін қамтитын жартылай интеграцияланған құралдардың жиынын қамтиды.

CASE-құралдары мынадай белгілері бойынша да жіктеуге болады:

1) қолданылатын әдіснамасы және модельдері және МҚ бойынша;

2) МҚБЖ-мен интеграциясы дәрежесімен;

3) қолданылатын платформасы бойынша;

Қазіргі әдістемелер және оларды іске асыратын технологиялар CASE құралдарымен бірге электрондық түрде көрсетіледі және методология бағытталған БҚ-ның жүйелерін құруға арналған процестер кітапханасынан, шаблонардан, әдістерден, модельдерден және басқа да компоненттерден тұрады. Электрондық методологиялар нақты пайдаланушылар үшін бейімделуді және нақты объекттерді орындау нәтижелері бойынша методологияның дамуын қамтамасыз ететін құралдарды қамтиды.

Бейімделу процесі ӨЦ-нің қажет емес процестерін, әрекеттері мен әдістеменің басқа да компоненттерін жою, сәйкес келмейтін процестер мен әрекеттерді, сонымен қатар, әдістерді, модельдерді, стандарттарды өзгерту болып табылады. әдістемені реттеу келесі аспектілер бойынша іске асырылады: ӨЦ – нің кезеңдері мен операциялары, жоба әзірлеушілері, ӨЦ – нің қолданылатын модельдері, концепциялар және т.б.

Электрондық методологиялар мен технологиялар және CASE құралдары АЖ – ні құру ортасының ұйымдастырылған аспаптық құралдар комплексінің ядросын жасайды.

4 Унифицирленген модельдеу тілі (UML)

Мақсаты: Унифицирленген модельдеу тілін (UML) қарастыру. UML диаграммасы мен құрылымы. Диаграммаларды қолдану үшін құрылымдық модельдеу және модельдеу ерекшелігін қарастыру. UML пайдалану үшін әр түрлі модельдеу проблемаларын шешу.

Унифицирленген модельдеу тілі (Unified Modeling Language -UML) - бұл бағдарламалық жүйелерді ерекшелендіру, бұрыштама қою, конструкциялау және құжаттамалау, сондай-ақ модельдер бизнесі мен өзге де бағдарламалық емес жүйелердің тілі. UML бұдан бұрын да үлкен және күрделі жүйелерді модельдеу кезінде ойдағыдай қолданылып жүрген инженерлік әдіс-тәсілдердің бірлестігін көрсетеді. UML-дің құрамдас бөлігі болып OCL табылады (Object Constraint Language - объектілерді шектеу тілі).

UML-ды өңдеу 1994 жылғы қазан айында басталды, бұл кезде Rational Software Corporation-нан шыққан Гради Буч (Grady Booch) және Джим Рамбег (Jim Rumbaugh), OMT (Object Modeling Technique - объектілік модельдендіру техникасы) әдістемесін бірыңғайландыру бойынша жұмыстарды бастаған болатын. 1995 жылғы қазан айында бірыңғайландыру әдісінің алдын-ала шамаланған болжамы ұсынылды. 1995 жылғы экономикалық құлдырау кезінде Иве Иакобсон (Ivar Jacobson) және оның Objectory компаниясы Rational-мен бірікті. Бірлесу қорытындысы болып OOSE (Object-Oriented Software Engineering) әдісімен бірыңғайландыру әдісінің қосылуы табылды.

Модельдендірудің әмбебап тілін құру кезінде Гради Буч, Джим Рамбег және Иве Иакобсон өздеріне келесідегі мақсаттарды қойды:

- ОБ әдістемесін (тек қана БҚ ғана пайдаланбастан) пайдалана отырып, модельдендіру жүйесін қамтамасыз ету;
- тілдің анық тұжырымдамасын жасау;
- күрделі жүйеде туындайтын үлкен мәселені шешу;
- адам ғана пайланып қоймайтын, сондай-ақ машина пайдалана алатын модельдендіру тілін жасау.

Бучтің, Рамбег және Иакобсонның әрекет жасауы 1996 жылдың қазан айында UML болжамында құжаттарды жасаумен аяқталды. 1996 жылдың ішінде Rational бірлесіп қызмет атқарушылардың UML консорциумын құрады. Консорциум DEC, HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI және Unisys-тан тұрады. Олардың ынтымақтастығының нәтижесі 1997 жылдың қаңтар айында 1.0 болжамындағы UML ерекшелігін құру болып табылады.

1997 жылдың қаңтар айында бірлесіп қызмет атқарушыларға Object Time, Platinum Technology, Ptech, Taskon&Reich Technologies, Softeam қосылып, 1997 жылдың 1 қыркүйегінде жарияланды.

Ақпараттандыру жүйесін дамытуда 1990-шы жылдар объектілік технологиялардың қалыптаса бастауы кезеңі болып танылды. Объектілік БҚБЖ нарығы қалыптасып, белсенді дами бастады. Деректер базасының жүйесін бағдарламалық қамтамасыз ету нарығында басымдылықпен келе жатқан Oracle, Informix және IBM компанияларының деректер базаларының объектілік-

реляциялық серверлерінің заманауи үлгілерінің шығуына байланысты, 1996-1997 жылдары реляциялық ортадан объектілік ортаға көшіп-қонуының процесі өтеді.

Аталған процестер, өз кезегінде талдаудың объектілік технологияларын және жүйелерді жобалаудың дамуын ынталандырды. Коммерциялық бағдарламалық өнімдерді іске асыратын түрлі әдістердің айтарлықтай саны пайда болды.

Көптеген компаниялар үшін бағдарламалық қамтамасыз етудің стратегиялық маңызы өсіп отырғандығына байланысты, индустрия бағдарламалық қамтамасыз етудің өндірісін автоматтандыру әдісін, оның сапасын көтеру, сондай-ақ оны нарыққа шығарудағы құны мен шығару уақытын төмендету әдістерін іздестіру үстінде. Бұл әдістер құрауыш технологияларға, көрсетушілік бағдарламаландыру, үлгілерді (pattern) және құрал-жабдықтық ортаны (framework) пайдалануға негізделеді.

UML – бұл бағдарламалық жүйелердің артефактілерін көрсету, арнайылау, құрылым және құжаттандыру тілі.

UML – бірыңғай модельдендіру тілі. Оны құруға индустрияның қандай да бір дәрежесінде болмасын барлық салалары қатысты болғандықтан, "UML – бұл бағдарламалық жүйелердің артефактілерін көрсету, спецификациялау, конструкциялау және құжаттандыру, сондай-ақ бизнес-процестердің және бағдарламалық емес жүйелердің тілі.

UML «міндеттерінің» тізбесіне басты назарды аудара кетейік. Спецификациялау, көрсету, конструкциялау және құжаттандыру - бұлардың барлығы да жоғары деңгейлі жобалауға тікелей қатысты болып отыр. Ал жоғары деңгейлі «құрал-саймандар» қолданудың бір-бірегей аспектісі болуы мүмкін емес, сондықтан, UML анықтамасын келесідегі көпаспектілі интерпретациямен толықтыруға болады:

- UML әдіс есебінде жүйелердің қозғалысын тану үшін пайдаланылады;
- UML тіл есебінде пәндік аймақ туралы білімді «есептен шығару» үшін пайдаланылады;
- UML модельдендіру тілі ретінде жүйелердің байланысу заңдылықтарын түсіну үшін пайдаланылады;
- UML бірыңғайландыру түрі ретінде құрастырушылардың қызметін үйлестіру үшін пайдаланылады.

Көптеген бағдарламалаушылардың көзқарасы көрсетіп отырғандай, жобаны іске асыру мәселесі бойынша ой-толғаулар шамамен оған код жазу үшін балама болып отыр. Кейбір заттар бағдарламаландырудың қандай да бір болмасын кодында тікелей алғанда өте жақсы мәнерленеді, себебі бағдарламаның мәтіні - бұл алгоритмдерді және өрнектерді жазу үшін өте қысқа және қарапайым жол.

Бірақ мұндай жағдайлардың өзінде программист бейресми болса да модельдендірумен айналысады. Ол өз ойының жобасын тақтада немесе майлықта жазды делік. Бірақ мұндай жақындасу жағымсыз лаң туғызуы мүмкін. Біріншіден, концепциялық модельге қатысты сылтаулар бойынша ой алмасуларға қатысатын пікірталастың қатысушылары бір тілде сөйлесетін кезде

ғана мүмкін болады. Жобаларда жасау кезінде компаниялар сөздерінің тілін жаңалық есебінде ашуы тиіс болатындығы ереже сияқты болып кеткен.

Бағдарламалық құралдарды шығаратын компания, орындалатын кодпен қатар басқа да артефактілерді шығарады, соның ішінде келесілер:

- жүйеге қойылатын талаптар;
- архитектура;
- жоба;
- бастапқы код;
- жобалау жоспарлары;
- тестер;
- прототип;
- болжам нұсқамалары, және т.б.

Өңдеу жөнінде қабылданған әдістемеге байланыссыз бір жұмысты орындау басқаларға қарағанда ресми жүргізіледі. Айтылған артефактілер -бұл жобаның құрамалы бөлігінің жеткізушілері ғана емес, олар басқару үшін, нәтижені бағалау үшін, сондай-ақ жүйені жасау және оған өрбіту жасап болған соңғы уақытта ұжым мүшелері арасындағы қарым-қатынас құралы есебінде қажет.

UML жүйелік архитектураны және барлық оның бөлшектерін құжаттандыру жөніндегі проблеманы шешуге мүмкіндік береді. Жүйеге қойылатын талаптарды тұжырымдау және тестерді анықтау үшін тілді ұсынады, және соңында жобаны жоспарлау,болжамдарды басқару кезеңінде жұмыстарды модельдендіру үшін құралдарды ұсынады.

UML тілі біріншіден бағдарламалық жүйелерді жасауға арналған. Оны пайдалану келесідегі салаларда әсіресе тиімді болмақ:

- кәсіпорын көлеміндегі ақпараттық жүйеде;
- банктік және қаржы қызметтерінде;
- телекоммуникацияда;
- транспортта;
- қорғаныс өнеркәсібінде, авиацияда және космонавтикада;
- бөлшектеп сату саудасында;
- медициналық электроникада;
- ғылымда;
- үлестірілген Web-жүйелерде.

UML қолдану аясы бағдарламалық қамтамасыз етуді модельдендірумен шектелмейді. Оның мәнерлілігі заң жүйесіндегі құжаттандыру айналымын, ауруханалардағы емделушілерге қызмет көрсету жүйесінің құрылымын және қызмет көрсетуін, аппарат құралдарына жобалау жасауды модельдендіруге мүмкіндік береді.

UML жасауда басты болып келесі мақсаттар танылған:

- пайдаланушыларға мағыналы модельдерді жасауға және олармен алмасуға мүмкіндік беретін, модельдендіруді көрсетушіліктің мәнерлі

тілін қолдануға дайын етіп ұсыну;

- базалық тұжырымдаманы кеңейту үшін кеңейту және мамандандыру механизмдерін қарастыру;
- бағдарламаландырудың нақты тілдерінен және жасау процестерінен тәуелсіз болуын қамтамасыз ету;
- модельдендірудің осы тілін түсіну үшін формальды негізін қамтамасыз ету.

UML патенттелген құрал болып табылмайды және барлығы үшін ашық. Ол өзі негізделіп жасалған әдістерді пайдаланудың тәжірибе жүзінде расталған ғылыми қоғамдастықтары мен тұтынушыларының қажеттілігін қанағаттандыру үшін тағайындалған.

Әдістеме бойынша көптеген мамандар, ұйымдар және инструменталды құрал-жабдықтарды тасымалдаушылар бұл тілді пайдаланамыз деп міндеттеме алған. UML Бучтың, OMT, OOSE әдістемелерінде және басқа да озық әдістерінде пайдаланылуы сияқты, сондай-ақ UML және кең қауымдастық серіктестерінің ұсыныстарын енгізетіндіктен, семантиканың және нотацияның негізінде құрылғандықтан бұл тілді мойындап, қадірлеу кең, жаратынды түрде болуы тиіс.

UML атауындағы «бірыңғайлау» эпитетінің екі аспектісі бар. Біріншіден, ол модельдендірудің ертеректегі әдістерінің тілдері арасындағы маңызды емес көптеген ерекшеліктерді іс жүзінде жоққа шығарады. Екіншіден, мүмкін аса ерекше шығар, ол жүйелердің көптеген түрлі түрлерінің болашағын, жасау фазаларын (талаптарды талдау, жобалау және іске асыру) және ішкі тұжырымдарды бірыңғайластырады (олармен байланысты бағдарламалық қамтамасыз етуді емес, бизнесті).

Дегенмен, UML нақты тілді анықтаса да, бұл модельдендірудің тұжырымдарын болашақта жетілдіру үшін тосқауыл бола алмайды. UML жасау кезінде көптеген озық әдістер назарға алынған болатын, бірақ UML келешектегі болжамына өзге де әдістер өз ықпалын тигізетін болады.

Сонымен қатар, UML негізінде жаңа перспективалық (болашақ) әдістер анықталуы мүмкін. UML оның ұйытқысын қайта анықтамастан-ақ,кеңейтілуі мүмкін.

UML оның ағымдағы түрінде көптеген инструменталды құрал-жабдықтардың, соның ішінде көрсетушілік модельдендірудің, ұқсатқыштық модельдендірудің, сондай-ақ құру ортасының негізі болып табылады.

Объектілік технологияларды дамытуда 1989 жылы құрылған Object Management Group (OMG) консорциумының құрылуы басты звено болып отыр, оның мақсаты - интероперальды біртекті емес бөлінген объектілік орталарды құру үшін индустриалды стандарттарды жасау болып табылады. OMG қабылдау, 1991 жылдан бастап, CORBA индустриалды стандартының және оның инфрақұрылымына байланысты стандарттардың бірқатар болжамдары, сондай-ақ тәжірибе жүзінде CORBA технологиясын белсенді түрде енгізу бұл стандарттардың объектілік талдау мен жобалау технологияларына, модельдендіру тілінің стандарттарын өндіру қажеттілігін сезінуге, CORBA архитектурасына сүйенетін жүйелерді құруға қолдау көрсететін бұл тектес

технологиялардың негізі болып табылатын, объектілік-бағдарланған инструменталды құрал-жабдықтардың технологиясын пайдаланатын интероперабельділікті қамтамасыз ететін және осы саладағы көптеген ұжымдар жинақтаған тәжірибелермен жинақталса да құрастырушылар мен пайдаланушылардың қызығушылығын тудыра алмады.

Осы мақсатта OMG қабылдаған көрсеткіштердің нәтижесінде 1997 жылдың қыркүйек айында қабылданған модельдендірудің бірыңғайландырылған тілі (UML) деп аталған тіл стандартының қабылдануы болды.

UML стандартының негізін объектілік талдау және жобалау жөніндегі аты әйгілі технологияларының үш ойларымен жинақталған Г.Буч, OOSE И.Якобсона және OMT Д.Рэмбо технологияларының Rational Software компанияларының ұсыныстарына бірқатар ірі корпорациялардың қолдау көрсеткендігі табылды.

UML кез келген жүйені модельдеу үшін жарамды: кәсіпорын масштабындағы ақпараттық жүйелерден таралған Web – қосымшаларға дейін және нақты уақыттың кіріктірілген жүйелері үшін де жарамды болады. Бұл өте мәнерлі тіл, ол жүйені жасау мен кейінгі жолды кеңінен ашып көрсетуге қатысы бар барлық көзқарас жағынан қарап шығуға мүмкіндік береді. Мәнерлі мүмкіндіктерінің молдығына қарамастан, бұл тіл түсіну және пайдалану үшін өте оңай. UML-ды зерттеуді оның концептуалды моделінен бастаған жөн, оның үш негізгі элементі болады: базалық құрылыс блоктары, осы блоктардың өзара үйлесімділігін анықтаушы ережелер және тілдің кейбір жалпы механизмдері.

Өзінің артықшылықтарына қарамастан UML — бұл тек тіл ғана; ол тек бағдарламалық қамтамасыз ету процесін құраушылардың бірі ғана. UML модельденетін нақтылыққа байланысты болмаса да, модельдеу процесі итерактивті және қадамдық болған жағдайда қолданған ыңғайлы. Ал жүйенің өзінің анық көрсетілген архитектурасы бар.

UML модельдеу тілі бағдарламалық қамтамасыз ету «сызбасын» құрастыру үшін стандартты құрал болып табылады. UML тілінің сөздігі мен ережесі жақсы анықталған модельді қалай құрып және қалай оқуды түсіндіреді, бірақ та қандай жағдайда қандай модельдерді құру керектігі туралы хабарламайды.

Жүйенің кейбір ерекшеліктерін бәрінен бұрын мәтін түрінде модельдеуге болады, екінші біреулерін – графикалық модельдеуге болады. Шын мәнісінде барлық қызықты жүйелерде бір ғана бағдарламалау тілінің көмегімен беру мүмкін емес құрылымдар бар болады. Мұндайда, UML – екінші белгіленген мәселелерді шешуге мүмкіндік беретін графикалық тіл.

UML – бұл жай ғана графикалық символдар жиыны емес. Олардың әрқайсысының артында жақсы анықталған семантика тұр. Бұл бір дайындаушы жазған модельді екінші біреу бірмәнді түсіндіруі мүмкін, тіпті аспаптық бағдарламамен интерпретациялануы мүмкін. UML талдау, жобалау және жүзеге асыруға қатысты барлық мәнді шешімдердің өзгешеліктерін анықтауға мүмкіндік береді, ол бағдарламалық қамтамасыз ету жүйесін жасау мен кеңінен тарату процесінде де қабылданылуы тиіс. UML-дың қолдану саласы

бағдарламалық қамтамасыз етуді модельдеумен шектелмейді.

UML стандартты модельдердің мынандай диаграммаларының жиынын ұсынады:

1. Қолдану нұсқаларының диаграммалары (Use case diagrams) – ұйымның бизнес-үрдістерін және жасалатын жүйенің талаптарын модельдеуде қолданылады.

2. Кластар диаграммалары (class diagrams) - жүйенің кластарының статикалық құрылымын және олардың арасындағы байланыстарды модельдеу үшін арналған.

3. Жүйенің тәртібі диаграммалары (behavior diagrams)

-Өзара байланыстар (interaction diagrams), тізбектілік диаграммалары (sequence diagrams) және кооперативтік диаграммалар (collaboration diagrams) – объектілер арасында хабарламалармен алмасу үрдісін модельдеуге арналған.

-Жағдайлар (қалыптар) диаграммалары (statechart diagrams) – бір қалыптан екінші қалыпқа өтуде жүйенің объектілерін модельдеуге арналған.

- Қызметтер диаграммалары (activity diagrams) қолданудың нұсқаларының әртүрлі шеңберінде жүйенің тәртібін модельдеуге арналған қызметтерді модельдейді.

4. Жүзеге асыру диаграммалары (implementation diagrams)

Компоненттер диаграммалары (component diagrams) – жүйе (ішкі жүйенің) компоненттерінің иерархиясын модельдеуге арналған.

Орналастыру диаграммалары (deployment diagrams) – жүйенің физикалық архитектурасын модльдеу үшін арналған.

Қолдану нұсқаларының диаграммалары.

Қандай да бір сыртқы объектімен, яғни қатысушы кейіпкердің (тұлғаның) әсерімен орындалатын оқиғаға жауап ретінде жүйеде орындалатын әрекеттердің (транзакциялардың) тізбегін қолдану нұсқасы деп аталады.

Қолдану нұсқасы пайдаланушы мен жүйе арасындағы типтік өзара әрекеттестікті сипаттайды. Қарапайым жағдайда қолдану нұсқасы пайдаланушының жүзеге асырғысы келген функцияларын талқылау үрдісінде анықталады.

Қатысушы кейіпкер, яғни актер (actor) – жүйеге қатысты пайдаланушының атқаратын ролі. Қатысушы кейіпкерлер нақты адамдарды немесе жұмыстардың атауын емес белгілі бір рөлді көрсетеді. Мысалы: менеджер, сатушы мен пайдаланушы есебінің жүйесі, клиент және т.б.

Қолдану нұсқаларының диаграммаларында адам фигурасымен бейнеленгенімен, ол берілген жүйеден ақпарат қажет болатын сыртқы жүйе де болуы мүмкін, яғни адам болуы міндетті емес.

Диаграммада қатысушы кейіпкерлерді көрсету оларға қандай да бір қолдану нұсқалары қажет болған жағдайда ыңғайлы.

Актерлер негізгі 3 типке бөлінеді:

- жүйенің пайдаланушылары;
- берілген жүйемен өзара әрекеттесетін басқа жүйелер;
- уақыт.

Жүйеде қандай да бір оқиғалардың болуы уақытқа тәуелді болса, онда

уақыт қатысушы кейіпкер бола алады.

UML тілінде қолдану нұсқалары мен қатысушы кейіпкерлер үшін байланыстардың бірнеше типтері қолданылады. Бұл байланыстар:

1) коммуникациялық (communication);

2) іске қосу (include);

3) кеңейтілу (extend);

4) жалпылау (generalization).

Коммуникация байланысы қолдану нұсқасы мен қатысушы кейіпкер арасындағы байланыс. UML тілінде коммуникация байланысын бір бағыттағы ассоциация, яғни бағытталған сызық арқылы көрсетеді. Сызық бағыты кімнің коммуникацияны бастағанын түсінуге мүмкіндік береді.

Іске қосу байланысы жүйенің тәртібін қандай да бір үзіндісі қолданудың нұсқасында бірден көп қайталанатын болған жағдайда қолданылады. Осындай байланыс көмегімен бірнеше рет қолданылатын функционалдылықты модельдейді.

Мысалы: Банктік жүйенің «Шоттан ақша алу» және «Салым жасау» сияқты қолдану нұсқалары транзакцияны жүзеге асыру алдында алдымен клиентті және PIN-кодты аутентификациялауы керек. Әрбіреуі үшін аутентификация үрдісін нақты сипаттау үшін осы қызметті «клиентті аутентификациялау» атауымен қолдану нұсқасының өзіне орналастыру керек.

Кеңейту байланысы жүйенің қалыпты тәртібінің өзгерісі сипаттауда қолданылады. Ол қолдану нұсқасына қажет болған жағдайда басқаның функционалдық мүмкіндіктерін қолдануға мүмкіндік береді. UML тілінде іске қосу және кеңейту байланыстары сәйкес стереотиптердің тәуелділіктері түрінде көрсетіледі оқиғалардың болуы уақытқа тәуелді болса, онда уақыт қатысушы кейіпкер бола алады.

Өзара әрекеттестіктер диаграммалары (interaction diagrams) объектілер тобының өзара әрекеттестіктерінің тәртібін сипаттайды. Ол қолданудың тек бір ғана нұсқасы шеңберіндегі объектілердің тәртібін қамтиды. Осы диаграммада объектілер қатары және олар өзара алмасатын хабарламалары бейнеленеді.

Хабарламалар (message) – объект-жіберуші объект-қабылдаушыдан өзінің амалдарының біреуін орындауды сұрайтын құрал.

Ақпараттық (informative) хабарлама – объект-қабылдаушыны жағдайын жаңалау үшін ақпаратпен жабдықтайтын хабарлама.

Хабарлама-сұраныс (interrogative) – объект-қабылдаушы туралы ақпарат беруді сұрайтын хабарлама.

Императивті (imperative) хабарлама – объект-қабылдаушыдан әрекеттерді орындауды сұрайтын хабарлама.

Өзара әрекеттестік диаграммасының 2 түрі бар:

- тізбектілік диаграммасы;
- кооперативтік диаграммалар.

Тізбектілік диаграммалары қолдану нұсқасы шеңберінде болатын оқиғалардың ағынын бейнелейді. Мысалы: «Шоттан ақша алу» қолдану нұсқасы бірнеше мүмкін тізбектілікті қарастырады:

1) ақша алу;

2) шотта жеткіліксіз болса да, ақшаны алу;

3) дұрыс емес PIN-код бойынша ақша алу және т.б.

Қандай да бір соманы алудың қалыпты сценарийін құрастыруға болады. Сценарий дегеніміз оқиғалар ағынының нақты экземпляры.

Бұл тізбектілік диаграммасы «Шоттан ақша алу» қолдану нұсқасы шеңберіндегі оқиғалардың ағынын бейнелейді. Қатысушы кейіпкерлер диаграмманың жоғары бөлігінде көрсетіледі. Мысалы: Клиент жүйеге осы қолдану нұсқасын орындауды талап ететін объектілер де жоғары бөлікте көрсетіледі. Бағыттар қатысушы кейіпкер мен объект арасында берілген немесе қажет функцияларды орындау үшін объектілер арасындағы хабарламаларға сәйкес келеді.

Тізбектілік диаграммасында объект пунктирлі тік сызықтың төбесінде тік төртбұрыш түрінде бейнеленеді. Тік сызық объектінің өмір сызығы (lifeline) деп аталады. Ол өзара әрекеттестік үрдісінде объектінің өмірлік циклінің үзіндісін көрсетеді. Әрбір хабарлама 2 объектінің өмір сызықтарының арасында бағыт түрінде бейнеленеді. Хабарламалар үстіден астыға қарай пайда болады. Әрбір хабарлама оның атымен белгіленеді, аргументтер мен қандай да бір басқару ақпаратын қосуға болады және тағы да өзін-өзі беруді көрсетуге болады.

Өзін-өзі беру – объект өзіне-өзі жіберетін хабарлама. Мұнда хабарлама бағыты өзінің өмір сызығына көрсетіледі.

Тізбектілік диаграммалары уақыт бойынша реттелген.

Кооперативті диаграммалаы қолдану нұсқасының нақты сценарийі арқылы оқиғалардың ағынын бейнелейді. Бұл диаграммалар объектілер арасында байланыстарға қатты көңіл бөледі. Бұл диаграммадан объектілер арасындағы байланысты жеңіл түсінуге болады, бірақ оқиғалардың тізбектілігін түсіндіру қиынырақ. Осындай себептерге байланысты сәйкес қандай да бір сценарийлер үшін диаграммалардың 2 типі де жасалады. Олар бір мақсатты көздесе де, бір ақпаратты құраса да, оларды әртүрлі көзқарастан көрсетеді.

Кооперативті диаграммаларда да бағыттар хабарламаларды бейнелейді. Олардың уақыт тізбектері хабарламаларды нөмірлеу жолымен көрсетіледі.

Кластар диаграммасы жүйенің кластар типтерін және олардың арасындағы әртүрлі текті статикалық байланыстарды анықтайды. Кластар диаграммасында кластар атрибуттары, кластар амалдары және кластар арасындағы байланыстарға таңылатын шектеулер кескінделеді. Диаграммада әрбір класс тік төртбұрыш түрінде бейнеленеді. Ол үш бөлікке бөлінеді,бірінші бөлігінде класс аты, екінші бөлігінде оның атрибуттары, үшінші бөлігінде кластың тәртібін бейнелейтін класс амалдары, яғни класпен орындалатын әрекеттер беріледі.

Кластарды байланыстыратын сызықтар кластар арасындағы өзара әрекеттестікті білдіреді.

Кластар стереотиптері. Стереотиптер кластарды категорияларға бөлуге мүмкіндік беретін механизм. UML тілінде негізгі стереотиптерге Boundary (Шекара), Entity (Мән), Control (Басқару) жатады.

Шекаралық кластар (Boundary Classes) – жүйенің және қоршаған

ортаның шекарасында орналасқан кластар. Олар барлық формаларды, есептерді, аппаратура интерфейстерін (принтер, сканер) және де басқа жүйелер интерфейстерін қамтиды.

Кластар шекарасын табу үшін қолдану нұсқаларының диаграммаларын зерттеу керек. Қатысушы кейіпкер мен қолдану нұсқасы арасындағы әрбір өзара әрекеттестікке бір шекаралық класс сәйкес келеді. Осындай класс қатысушы кейіпкерге жүйемен өзара байланыс жасауға мүмкіндік береді.

Мәндер-кластары (Entity Classes) пәндік аймақтың негізгі ұғымдарын бейнелейді және де сақталған ақпараттардан тұрады. Әдетте әрбір мән-класс үшін мәліметтер қорында кесте жасайды.

Басқарушы кластар (Control Classes) – басқа кластардың әрекеттерінің үйлестіктеріне жауап береді. Әдетте қолданудың әрбір нұсқасында осы нұсқаның оқиғаларының тізбектілігін бақылайтын бір басқарушы класс болады. Басқарушы класс басқа кластарға жауапкершілік береді, сол себептен оны класс-менеджер деп те атайды. Жүйеде қолданудың бірнеше нұсқаларына ортақ басқа да басқарушы кластар болуы мүмкін. Мысалы: қауіпсіздікке байланысты оқиғаларды бақылауға жауап беретін Security Manager – қауіпсіздік класы. Transaction Manager – транзакциялар менеджері класы мәліметтер қорымен транзакцияларға қатысты хабарламалардың үйлестігімен айналысады.

Жүйенің қызметінің басқа да элементтерімен жұмыс үшін басқа да менеджерлер болуы мүмкін. Мысалы: ресурстарды бөлісу, мәліметтерді таратылған өңдеу немесе қателерді өңдеу.

Пакеттер механизмі. Пакеттер қандай да бір жалпылыққа ие болатын кластарды топтауда қолданылады. Топтаудың мейлінше кең таралған тәсілдері:

- Стереотипі бойынша топтау. Бұл тәсіл дайын жүйені орналастыру тұрғысында пайдалы.
- Кластардың қызметі бойынша топтастыру. Жетістігі – қайталап қолдану мүмкіндігінің болуы.

Пакеттер механизмі кластарға ғана емес, модельдің кез келген элементіне қолдануға болады. Кез келген екі кластың арасындағы пакетте кез келген тәуелділік бар болса, онда екі пакеттің арасында тәуелділік болады. Сонымен пакеттер диаграммасы кластар пакетінен және олардың арасындағы тәуелділіктен тұратын диаграмманы көрсетеді .

Пакеттер мен тәуелділіктер кластар диаграммаларының элементтері болып табылады, яғни пакеттер диаграммасы кластар диаграммасының формасы.

Екі элемент арасындағы тәуелділік мына жағдайда орын алады: егер бір элементті анықтаудағы өзгерістер екінші элементке де ықпалы тисе. Кластар арасындағы тәуелділіктер әртүрлі болуы мүмкін: бір класс екінші класқа хабарлама жібереді; бір класс екінші класты амалдар параметрі ретінде қолданатын болса. Егер класс өзінің интерфейсін өзгертетін болса, онда оның жіберетін кез келген хабарламасы өзінің күшін жоюы мүмкін. Пакеттер диаграммасын жалпы құрылымдық жүйені басқарудың негізгі құралы деп қарастыруға болады.

Пакеттер үлкен жобалар үшін өмірлік қажетті құрал болып табылады.

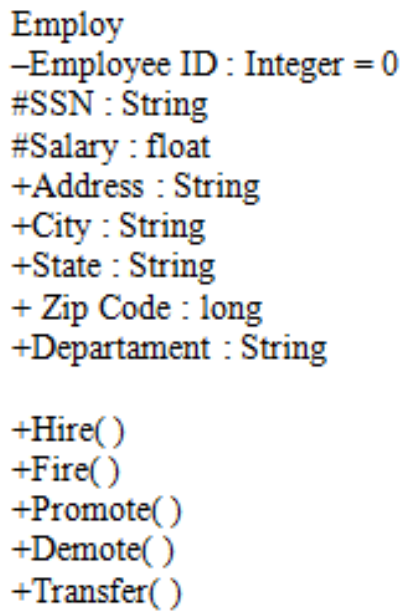
Атрибут дегеніміз класпен байланысты ақпарат элементі. Мысалы: Компания класында атауы (Name), мекен-жайы (Address) және қызметкерлерінің саны (Number of Employees) болуы мүмкін.

Атрибуттар кластың ішінде тұрады, сондықтан олар басқа кластар үшін жасырын болады. Осыған байланысты кей кездері қандай кластардың атрибуттарды оқуға және өзгертуге құқығы бар екендігін көрсету керектігін талап етеді. Бұл қасиет атрибуттың көрінуі (attribute visibility) деп аталады. Атрибутта осы параметрдің 4 мүмкін мәнін анықтауға болады. Олардың әрқайсысын мысал арқылы қарастырайық (4-сурет).

Employee класы Address атрибутымен және Company класы берілсін.

Атрибуттың 4 мүмкін мәні:

1. Жалпы, ашық (Public) – барлық кластарға көрініп тұрады. Кез келген класс оны қарауына және мәнін өзгертуіне болады. Ендеше, Company класы Employee класының Address атрибутының мәнін өзгерте алады. UML нотациясына сәйкес жалпы атрибутқа «+» таңбасы қойылады.
2. Жабық, құпия (Private) – ешбір класқа көрінбейді. Employee класы Address атрибутының мәнін біледі және оны өзгерте алады, ал Company класы оны көре де алмайды, өзгерте де алмайды. Егер қажет болса, ол Employee класынан атрибуттың мәнін қарауға рұқсат сұрайды немесе жалпы амалдардың көмегімен осы атрибуттың мәнін өзгертуді ұсынады. Жабық атрибут сәйкесінше «–» таңбасымен белгіленеді.
3. Қорғалған (Protected) – атрибут кластың өзіне ғана және оның ішкі кластарына ғана ашық. Бұл атрибут «#» таңбасымен белгіленеді.
4. Пакетті (Package or Implemention) – атрибут пакет шеңберінде ғана жалпы болып табылады. Бұл атрибут ешқандай таңбамен белгіленбейді.



4 сурет – Атрибуттардың көрінуі

Амалдар класпен байланысқан тәртіпті жүзеге асырады. Амалдар үш бөлікті қамтиды: аты, параметрі және қайтарылатын мәннің типі.

Парметрлер – «кірісте» амал арқылы алынған аргументтер. Қайтарылатын мәннің типі амал әрекеті нәтижесіне қатысты.

Кластар диаграммасында амалдар аттарын және олардың параметрлері мен қайтарылатын мәнінің типімен амалдар аттарын көрсетуге болады.

UML тілінде амалдар мынандай нотациямен беріледі. Амал аты (аргумент1: аргумент1-дің мәліметтер типі, аргумент2: аргумент2-нің мәліметтер типі, ...): қайтарылатын мәннің типі.

Амалдардың әртүрлі төрт типін қарастырайық:

1. Жүзеге асыру амалдары (implementor operations) – қандай да бір бизнес-функцияларды жүзеге асырады. Өзара әрекеттестік диаграммаларын зерттеу арқылы осындай амалдарды табуға болады. Осы типтің диаграммалары бизнес-функцияларда қолданылады, диаграмманың әрбір хабарламасын жүзеге асыру амалымен сәйкес келтіруге болады.

2. Басқару амалдары (manager operations) – объектілерді жасау және жоюды басқарады. Бұл категорияға кластардың конструкторлары мен деструкторлары жатады.

3. Қатынау амалдары (access operations) – басқа кластардың атрибуттарының мәндерін қарау немесе өзгерту үшін қажет. Мысалы: Employee класында Salary атрибуты болсын. Басқа кластар бұл атрибутты өзгерткенін қаламаймыз. Оның орнына Employee класына GetSalary және SetSalary сияқты қатынаудың 2 амалын қосамыз. Біріншісі GetSalary-ға басқа да кластар қатынай алады, себебі, ол жалпы болып табылады. Ол Salary атрибутының мәнін алады және оны шақырған класқа осы мәнді қайтарады. SetSalary амалы да ортақ болып табылады, ол оны шақырған класқа Salary атрибутының жаңа мәнін беруге көмектеседі. Бұл амалда жалақы өзгеруі үшін орындалуға қажетті тексерудің кез келген ережелері мен шарттары болуы мүмкін.

Осындай тәсіл класс ішінде атрибуттарды қауіпсіз инкапсуляциялауға мүмкіндік береді, басқа кластардан қорғайды, оған бақыланатын қатынау жасауға мүмкіндік береді. Әрбір кластың атрибуты үшін Get және Set амалдарын (мәнді алу және өзгерту) жасау стандарт болып табылады.

4. Көмекші амалдар (helper operations) – класс үшін оның жауапкершілігін орындауға қажетті кластың амалдары, бірақ бұл туралы басқа кластар ештеңе білмеуі қажет. Бұл кластың жабық және қорғалған амалдары. Амалдарды идентификациялау үшін келесі әрекеттерді орындау керек:

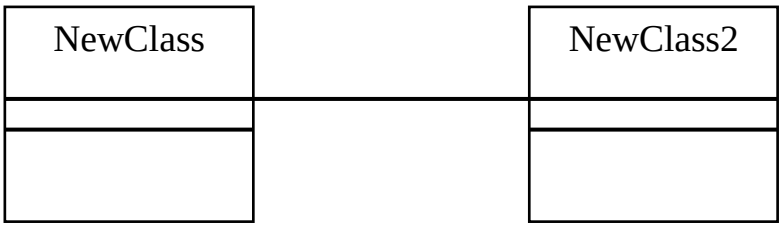
- тізбектілік және кооперативті диаграммаларды толықтай оқып-үйрену керек. Бұл диаграммалардағы хабарламалардың үлкен бөлігі жүзеге асыру амалдары болып табылады. Рефлексивті хабарламалар көмекші амалдар болып табылады.
- басқару амалдарын қарастырыңыз. Конструкторлар және деструкторларды қосу қажеттігі туындайды.
- қатынау амалдарын қарастырыңыз. Кластың әрбір атрибуты үшін (олармен жұмыс жасайтын басқа кластар үшін) Get және Set амалдарын жасау керек.

Кластар арасындағы семантикалық өзара байланысты білдіреді. Ол класқа басқа кластың атрибуттары, операциялары және байланыстары туралы білуге мүмкіндік береді. Басқа сөзбен айтқанда бір класс басқа класқа хабарлама жібергенде тізбектілік диаграммасында немесе кооперативті диаграммада олардың арасында байланыс болу керек.

Кластар арасында қойылатын байланыстың 4 типі бар:

- ассоциациялар (association);
- тәуелділіктер (dependency);
- агрегациялар (aggregations);
- жалпылау (generalization).

Ассоциация – кластар арасындағы семантикалық байланыс. Оны кластар диаграммасында кәдімгі сызықпен көрсетеді (5-сурет).

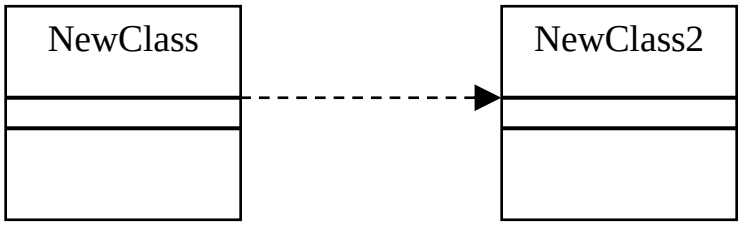


5 сурет – Ассоциация

Ассоциациялар бір бағытты немесе екі бағытты болады. UML тілінде екі бағытты ассоциацияларды бағыттаушысыз қарапайым сызықпен немесе екі жағынан да бағыттаушысы бар сызықпен көрсетеді.

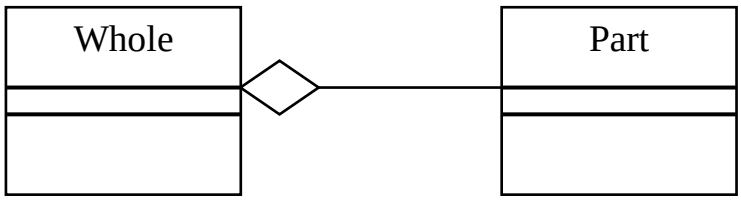
Ассоциациялар бағытын тізбектілік диаграммалары мен кооперативті диаграмманы оқып-үйренуде анықтауға болады. Егер барлық хабарламалар оларға бір ғана класпен жіберіліп, басқа класпен қабылданатын болса және керісінше, онда осы кластар арасында бір бағытты байланыс бар. Егер ең болмағанда бір хабарлама кері жаққа жіберілетін болса, онда ассоциация екі бағытты болғаны. Ассоциациялар рефлекситі болуы да мүмкін. Кластың бір экземпляры осы кластың басқа да экземплярымен өзара әрекеттескен жағдайда рефлексивті ассоциация пайда болады.

Тәуелділіктер де кластар арасындағы байланысты бейнелейді, бірақ олар әрқашан бір бағытты және бір класс басқа класпен жасалған анықтамаларға тәуелді екендігін көрсетеді. Тәуелділіктер үзік сызықтармен жүргізілген бағыттауыштармен кескінделеді (6-сурет).



6 сурет – Тәуелділік

Осы кластар үшін кодты генерациялауда олара жаңа атрибуттар қосылмайды. Агрегациялар ассоциацияның неғұрлым тығыз формасын білдіреді. Агрегация дегеніміз бүтін мен оның бөліктері арасындағы байланыс. Мысалы: Компьютер класы, сонымен бірге Жүйелік блок класы, Монитор класы және т.б. Агрегациялар бүтін болып табылатын класта кішкентай ромбысы бар сызықпен бейнеленеді (7-сурет). (Whole - бүтін, Part – бөлік).



7 сурет – Агрегация

Қарапайым агрегацияға толықтыру үшін UML агрегациясының неғұрлым күшті түрі – декомпозицияны енгізеді.

Жалпылау екі класс арасындағы мұралау байланысын көрсетеді. Объектіге бағытталған тілдер мұралау концепциясын қолдайды. Ол бір класқа басқа кластың барлық атрибуттарын, операцияларын және байланыстарын мұралауға мүмкіндік береді. UML тілінде мұралау байланысын жалпылау деп атайды және оны ішкі кластан түпкі класқа бағытталған бағыттауыштар түрінде кескіндейді

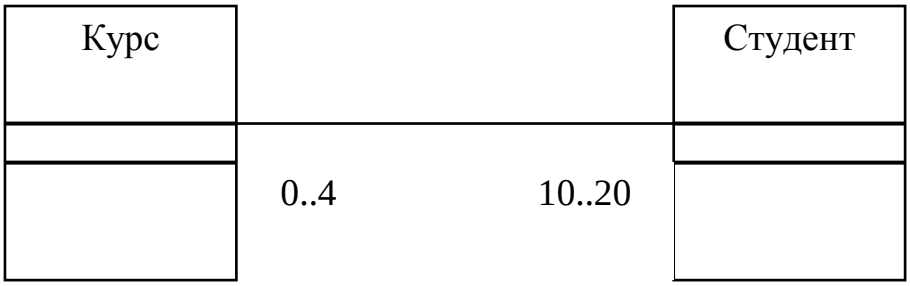
Әрбір ішкі кластың мұраланғаннан басқа өзінің меншікті бірегей атрибуттары, операциялары және байланыстары болады.

Бір кластың қанша данасы осы байланыс көмегімен уақыттың осы сәтінде басқа кластың бір экземплярымен өзара әрекеттесуін жиын (multiplicity) деп атайды. Мысалы: Университеттегі курстарды тіркеудің жүйесін жасауда мынандай кластарды анықтауға болады: Курс, Студент. Олардың арасында мынандай байланыс қойылған: курстарда студенттер бар, студенттерде курс.

Жиынның параметрі жауап беретін сұрақтар мынандай болуы мүмкін:

1. Осы сәтте студент қанша курсқа қатыса алады?
2. Бір уақытта бір курсқа қанша студент қатыса алады?

Жиын осы 2 сұраққа жауап бере алатындықтан оның индикаторлары байланыс сызығының екі ұшына да орналастырылады. Мысалы: Бір студент 0-ден 4 курсқа дейін қатыса алады, ал бір курсты 10-нан 20-ға дейін студенттер таңдай алсын. Осы берілгендерді диаграммада былайша көрсетеміз (8-сурет).



8 сурет – Жиын

UML тілінде жиынды белгілеу үшін келесі нотациялар қабылданған (3-кесте):

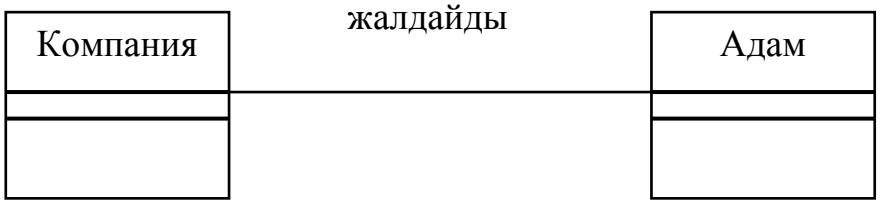
3 кесте – Нотациялар

Жиын	**Мәні**
*	Көп
0	Нөл
1	Бір
0 .. *	Нөл немесе одан артық
1 .. *	Бір немесе одан артық
0 .. 1	Нөл немесе бір
1 .. 1	Тек біреу

Байланыстарды оның аттарымен немесе рөлдік аттарымен нақтылауға болады.

Байланыс аты - әдетте етістік болады немесе оның не үшін қажет екендігін сипаттайтын етістікті фраза.

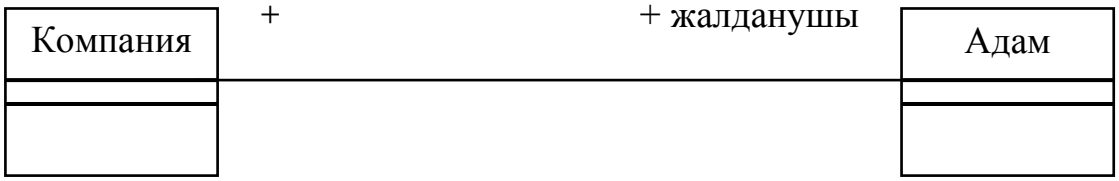
Мысалы: Адам (Person) және Компания (Company) кластарының арасында ассоциация болуы мүмкін. Осымен байланысты Адам класының объектісі компанияның клиенті, оның қызметкері немесе иесі болып табылады ма? деген сұрақ туындайды. Осыны анықтау үшін ассоциацияны «employs» (жалдайды) деп атауға болады (9-сурет).



9 сурет – Байланыс аты

Байланыстардың аттарын анықтау міндетті емес. Әдетте оны байланысты жасау себебі көрінбеген болса ғана жасайды. Атты байланыс сызығында көрсетеді.

Рөлдер. Рөлдік аттар ассоциация немесе агрегация байланыстарында қолданылады. Адам класы Компания класының қызметкерінің рөлін атқарады. Рөлдік аттар – зат есімдер немесе олардың фразаларын қамтушылар. Оларды диаграммада класпен бірге көрсетеді (10-сурет).

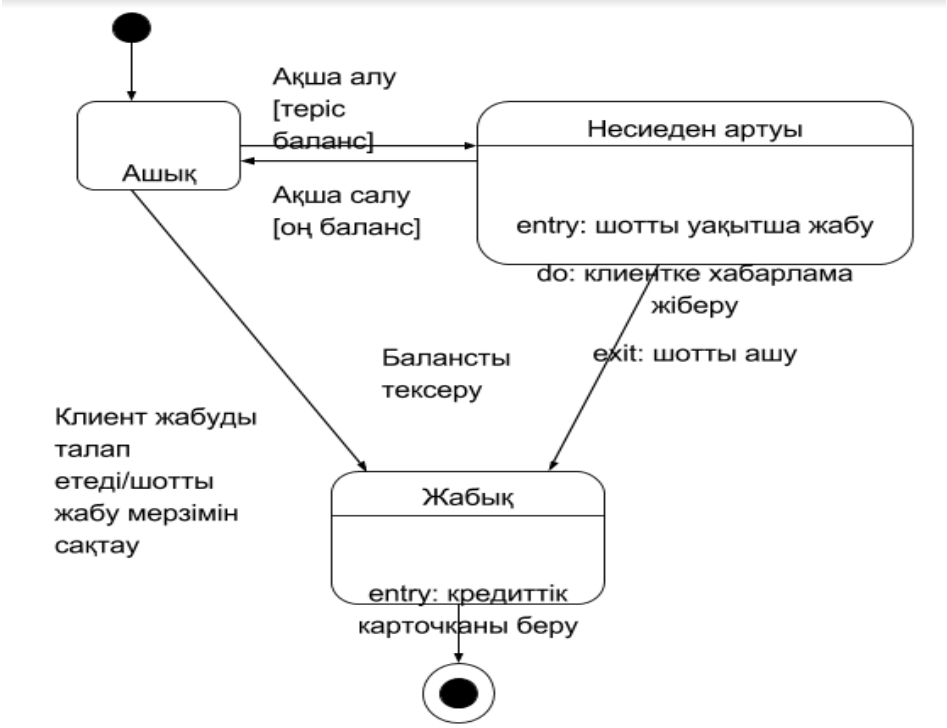


10 сурет – Рөлдік аттар

Жағдайлар диаграммасы.

Жағдайлар диаграммалары нақты объект болатын барлық мүмкін жағдайларды анықтайды, қандай да бір оқиғалардың түсу нәтижесінде объектінің жағдайының ауысу үрдісін анықтайды. Бір-бірінен керемет айырмашылығы болмайтын жағдайлар диаграммаларының көптеген формалары бар.

Банктік есеп-шот үшін жағдайлар диаграммасын көрсетейік (11-сурет). Есеп-шоттың бір жағдайдан екінші жағдайға көшу үрдісін де осы диаграммадан бақылауға болады. Мысалы: Егер клиент есеп-шотты жабуды талап етсе, онда ол «Жабық» жағдайына көшеді. Клиенттің талабы оқиға (event) деп аталады, осындай оқиғалар объектінің бір жағдайдан екіншісіне көшуді шақырады.



11 сурет – Есеп-шот класы үшін жағдайлар диаграммасы

Егер клиент есеп-шоттан ақша алса, ол «Несиенің артуы» жағдайына көшеді. Егер есеп-шот бойынша баланс 0-ден кем болса ғана осы орындалады, яғни диаграммадағы [теріс баланс] шартымен сәйкес [ ] жақшасындағы шарттар объектінің бір жағдайдан екінші жағдайына көшуін анықтайды.

Жағдайлар диаграммасында екі арнайы жағдай болады: бастапқы жағдай

(start) қара нүктемен белгіленеді, ол объектінің жойылуының алдындағы жағдайына сәйкес келеді. Жағдайлар диаграммасында тек бір ғана бастапқы жағдай, ал бірнеше соңғы жағдайлар болуы мүмкін. Объект қандай да бір нақты жағдайда болғанда, әртүрлі үрдістер орындалуы мүмкін. Мысалдағы диаграммада несиенің артуы барысында клиентке сәйкес хабарлама жіберіледі. Объектінің нақты бір жағдайда болатын сәтінде жүретін үрдістер әрекеттер (actions) деп аталады.

Жағдайлармен байланысатын мынандай мәліметтер болады:

- қызмет (activity);
- кіріс әрекеті (entry action);
- шығыс әрекеті (exit action);
- оқиға (event).

Осы мәліметтерді банк жүйесінің Есеп-шот класының жағдайлар диаграммасында қарастырайық (жоғарыдағы 11-суретке сәйкес).

Қызмет – осы жағдайда объектімен жүзеге асатын тәртіп. Мысалы: Есеп-шот «Жабық» жағдайында болғанда пайдаланушыға кредиттік карточкасы қайтарылады.

Қызмет – үзуге болатын тәртіп, яғни осы жағдайда объектінің болғанында аяқталғанға дейін орындалуы мүмкін немесе объектінің басқа жағдайға көшуіне байланысты үзілуі мүмкін. Қызметті жағдайдың ішінде бейнелейді, оның белгіленуі do (жасау керек) қызметші сөзінен кейін жазылады.

Кіріс әрекеті – берілген жағдайға объектінің көшуінде орындалатын тәртіп. Есеп-шот «Несиенің артуы» жағдайына көшуінде объектінің осы жағдайға қайдан көшкеніне тәуелсіз «Уақытша есеп-шотты жабу» әрекеті орындалады. Кіріс әрекеті үзуге болмайтын әрекет ретінде қарастырылады. Кіріс әрекеті жағдайлар ішінде көрсетіледі, entry: (енгізу) сөзінен кейін орындалады.

Шығыс әрекеті кіріс әрекетіне ұқсас. Нақты жағдайдан шығу үрдісінің құрамдас бөлігі ретінде орындалады. Жоғарыдағы мысалдағы «Несиенің артуы» жағдайынан «Есепті ашу» әрекеті орындалады. Жағдайлар ішінде бейнеленеді, exit: (шығу) сөзінен кейін орындалады, үзуге болмайтын әрекет.

Қызметі уақытында, кіріс және шығыс әрекеттері барысында объектінің тәртібі басқа объектіге оқиғаларды жіберуді қамтуы мүмкін. Мысалы, Есеп-шот объектісі . Карточканы оқу құрылғысы объектісіне хабарлама жіберуі мүмкін? Осы жағдайда қызметтің сипаттамасына, кіріс немесе шығыс әрекеттеріне «^» таңбасы қойылуы мүмкін. Диаграммада мынандай жол болуы мүмкін:

Do: ^ Мақсат. Оқиға (Аргументтер).

Мұндағы: Мақсат – оқиғаны алатын объект, оқиға – жіберілетін хабарлама, ал аргументтер жіберілетін хабарламалардың параметрлері.

Қызмет объектінің қандай да бір оқиғаны алуы нәтижесінде орындалуы мүмкін. Мысалы: Есеп-шот объектісі «Ашық» жағдайында болуы мүмкін. Объектінің бір жағдайдан екінші жағдайға ауысуы көшу (transition) деп аталады. Диаграммада аудармасы стрелкалармен бейнеленеді. Көшулер рефлексивті болуы мүмкін.

Көшулерде бірнеше спецификациялар болады: оқиғалар, дәлел,

қоршалған шарттар, әрекеттер және жіберілетін оқиғалар.

Оқиға – бір жағдайдан екінші жағдайға көшіуді шақырушы. Мысалы: «Клиенттің есеп-шотты жабуды талап етуі».

Диаграммада оқиғаларды бейнелеу операция аты ретінде немесе әдеттегі фраза ретінде қолданылуы мүмкін.

Оқиғалардың аргументтері болады. «Салым жасау» оқиғасы «Есептеу артылды» жағдайынан «Ашық» жағдайына көшуді шақырады, сонда ол саны (депозит сомасын сипаттайтын) аргументіне ие болуы мүмкін.

Көшудің жүзеге асырылуы мүмкін немесе мүмкін емес екендігін анықтаушыны қоршалған шарттар деп атайды. Мысалы: «Салым жасау» оқиғасы баланстың 0-ден артық болған жағдайында «Есептің артылуы» жағдайынан «Ашық» жағдайына көшіреді, қарсы жағдайда көшу жүзеге асырылмайды.

Қызметтер диаграммасы параллель үрдістердің саны көп болғанда тәртіпті сипаттауда ыңғайлы.

Компоненттер диаграммасы физикалық деңгейде модельдің қалай көрінетінін көрсетеді.

Орналастыру диаграммалары жүйенің бағдарламалық және аппараттық құраушыларының физикалық өзара байланысын бейнелейді.

Қорыта келе, UML-дың мынадай мүмкіндіктерін атауға болады.

- UML объектілі-бағдарланған болып келеді, соның нәтижесінде талдау және жобалау нәтижелерін сипаттау әдістері семантикалық жағынан қазіргі заманғы ОО-тілдерде бағдарламалау әдістеріне жақын болады;
- UML жүйе сипатының түрлі аспектілері мен барлық мүмкін болатын іс жүзіндегі көзқарастарының жүйесін сипаттауға мүмкіндік береді;
- UML диаграммасы модельді оқу үшін оның синтаксисімен жедел танысу үшін өте қарапайым болып келеді;
- UML дербес мәтіндік және графикалық стереотиптерді кеңейтуге және ендіруге мүмкіндік береді, бұл оның тек бағдарламалық инженерияда ғана емес, басқа да қолданысына ықпал етеді;
- UML кеңінен таралып және жедел қарқынмен дамып келеді.

UML тілі әртүрлі бөлімдер мен өндірушілер арасындағы әмбебап “көпір”.

4.1 UML тілінде, Rational Rose, Aris, Microsoft Office Visio 2007 ортасын пайдала отырып жүйелерді жобалау

Rational Rose UML тіліне негізделіп жобалау және обьектілі бағытталған талдау әдістерін қолданады. Rational Rose жаңа жобаларда бағдарламалық компоненттерінің қайта қолдануын қамтамасыз ететін бағдарламалар мен мәліметтер қорының реверстік инжинирингтің құралдарынан тұрады.

Rational Rose көмегімен модульдегі кері талдау базасы негізінде жаңа модельдер құруға немесе қолданбалы бағдарлама текстін және кітапханалар

класстарын анықтауға болады. Rose жүйесі – визуальды модельдеу жүйесінде жақсы орын алады, оны қолдана отырып құрылып отырған қосымшаның құрылымын құруға, оның мүмкін орындау текстін генерациялауға және параллельді өңделіп отырған жүйенің құжаттарымен жұмыс жасауға болады.

Rational Rose модельдеу ортасындағы көшбасшы болып табылады. Оның ағымдағы түрінде көптеген инструменталды құрал-жабдықтардың, соның ішінде көрсетушілік модельдендірудің, ұқсатқыштық модельдендірудің, сондай-ақ құру ортасының негізі болып табылады.

Rational Rose UML тіліне негізделіп жобалау және обьектілі бағытталған талдау әдістерін қолданады. Rational Rose осы болжамасы C++, Visual C++, Visual Basic, Java, PowerBuilder, CORBA Interface Definition Language (IDL) бағдарламалар үшін кодтар генерациясын және ANSI SQL, Oracle, MS SQL Server, IBM DB2, Sybase үшін мәліметтер қорының генерация бейнеленуін, сонымен қатар диаграмма түріндегі жобалау құжаттарын және егжей-тегжейлерін іске асырады. Rational Rose жаңа жобаларда бағдарламалық компоненттерінің қайта қолдануын қамтамасыз ететін бағдарламалар мен мәліметтер қорының реверстік инжинирингтің құралдарынан тұрады.

Құрылымдар және функциялар. Rational Rose - да жұмыс істеудің негізі жүйе архитектурасының статикалық және динамикалық аспектілерін анықтайтын UML егжей- тегжейі мен диаграммаларды құру болып табылады. Rational Rose құрамындағы келесі алты негізгі құрылымдық компоненттерді белгілеуге болады: репозиторий, қолданушының графикалық интерфейсі, жобаны қарау құралдары, жобаны бақылау құралдары, құжаттардың статистикалық және генераторлық құралдарын жинау. Оларға сонымен қатар кодтар генераторлары (әрбір тіл үшін жеке) және реверстік инжинирингті қамтамасыз ететін C++ үшін анализатор кіреді.

Репозиторий жобаның мәліметтер қоры болып табылады. Браузер иерархия кластары бойынша орын ауыстыру, диаграммалардың бір түрінен екінші түріне ауысуды жоба бойынша «навигацияны» қамтамасыз етеді.

C++ тіліндегі автоматы түрдегі генерация кодтар бағдарламасының құралдары компоненттер мен диаграммалар кластарында болатын бағдарламаны қолданады да тақырыптар және класстардың файлдары және объектілерін қалыптастырады. Осындай түрмен бағдарламаның «қанқасын» жасайды да ол C++ тілінде тура бағдарламалауда анықталады. C++ кодтар анализаторы жеке бағдарламалық модуль тәрізді негізделген. Оның тағайындалуы C++- тегі қолданушы анықтайтын мәтінде болатын бағдарламаның негізінде Rational Rose- дағы проекттер модулін құру. Жұмыс барысында анализатор мәтіндердің дұрыстылығын және қателердің болдырмауын іске асырады. Оның жұмыс қорытындысында алынған модель бүтіндей немесе бөлшектей әртүрлі жобада қолданылуы мүмкін. Анализатордың кіріс және шығыс күші бойынша кең мүмкіншіліктері бар. Мысалы файлдар типін, компилятор қорларын, қандай анықтама модельге кіруі және қандай модель элементтері экранға шығуын анықтауы мүмкін. Осындай мүмкіншіліктермен Rational Rose/C++ бағдарламалық компоненттердің қайта қолднылуын қамтамасыз етеді. Жобаны құрудың қорытындысында Rational

Rose CASE құралдарының көмегімен келесі құжаттар құралады:

- UML диаграммалары;
- Кластар, обьектілер, атрибуттар және операциялар жүйелері;
- Бағдарламалар мәтінін дайындау нұсқалары;

Бағдарлама мәтінін бағдарламашының келесі жұмыстары үшін дайындау нұсқалары болып табылады. Келесі құралдармен әрекеттестік және топпен жұмыс жасау ұйымы.

Rational Suite келесі нұсқалары бар:

Rational Suite AnalystStudio - анықтау және толық жиындар талаптарын басқаратын құралдар жүйесі;

Rational Suite Development Studio - БҚ жобалау және орындауға арналған;

Rational Suite Test Studio - қосымшалардың автоматты түрде тестілеу үшін арналған нәрселер жиыны;

Rational Suite Enterprise - БҚ толық өмірлік циклін қамтамасыз етеді;

Rational Suite жиынына Rational Rose - дан басқа келесі компоненттер кіреді:

Rational Requisite Pro - өндірушілер тобының бірігіп жұмыс істеу ұйымы үшін арналған талаптармен басқару құралы;

Rational ClearCase – БҚ конфигурациясымен басқару құралы;

Rational SoDA – жобалы құжаттаманың автоматты генерациясының құралы;

Rational Clear Quest - e-mail және web құралдар негізінде өзгерістерді басқару және жобада ақауларды қадағалау құралы;

Rational TeamTest – бағдарламаның орындалуы кезіндегі қателерді автоматты түрде табу және регресивті тестілеу өткізу үшін сценарийлер генерациясының құралы;

Rational Robot – тестілердің автоматты түрде жіберілу және модификацияны құру құралы;

Rational Purify - Бағдарламаның орындалу кезіндегі қиын табылатын қателерді локализациялау құралы;

Rational PureCoverage – тестілеу кезінде өткізілген кодтың идентификациялау учаскісінің құралы;

Rational Quantify - Бағдарламаның толық жұмыс жасау нәтижелілігіне байланысты болатын тар орындар көлемін анықтау құралы;

Rational Suite Performancestudio – «клиент-сервер» қосымшасы мен Web-қосымшасының жүктемелік тест құралы. Топтық жұмыстың ұйымы үшін Rational Rose-та модельді басқаратын ішкі модельдерге бөлуі мүмкін. Олардың әрбіреуі дискіде тәуелсіз сақталады немесе модельге жүктеледі, R сапасында ішкі модельдер қаттама немесе ішкі жүйелерге қатысуға болады.

Класс түсінігі. Класс (Class) ортақ қасиетттері (атрибуттары), тәртібі (функциялары), семантикасы және басқа объектермен байланысы бар объектілер тобын анықтайды. Класты объектіні құруға арналған шаблон ретінде қарауға болады. Әрбір объект қандайда бір ғана кластың нұсқасы мысал ретінде келесідей сипаттамалары бар ‘курсты ұсыну’ класс анықтамасын

қарастырайық: – Атрибуттар – «сабақты өткізетін орын», «сабақты өткізетін уақыт», «курстың аталуы», «курс нөмірі», «ұсыныс нөмірі»; – Функциялары – «сабақты өткізетін орынды анықтау», « сабақты өткізетін уақытты анықтау», «студентті тіркеу», «студентті тіркеуді тоқтату». «Курстық ұсыну» класының нұсқасының орнына ‘алгебра 101, 1- бөлім’ және ‘алгебра 101, 2- бөлім’ объектілерінде алуға болатын еді, себебі олардың да әрқайсысының нақты атрибуттурының жиынтығы және ортақ функционалдық сипаттамалары бар. Дұрыс құрылған класс тек бір ғана абстракцияны бере алады. Мысалы, студент туралы мәлімет сақталған, сонымен қатар барлық оқу барлығында студент өткен курстар тізімі функциясы көрсетілген класты сәтті құрылған деп айта алмаймыз, өйткені ол екі әртүрлі операциялар тобын қамтиды. Сондықтан мұндай класты екіге – «студент» және «студент өткен курстар» бөлсе жақсы болар еді. Кластың атауы үшін пән облысына сәйкестендіріліп қабылданған терминдерді қолданған дұрыс. Класс аты ретінде жобаланатын түсініктерде толығымен сипаттай алатын зат есіміңіздің жекеше түрі қолданылады. Кейде қысқартылған атауларда қолданылып, класты құжаттандырғанда міндетті түрде мағынасын ашып көрсету керек. Егер аббревиатура бірдей мағыналы емес интерпретация жіберсе, онда сәйкесінше айтылудың толық түрін қолданамыз. Объектілер мен кластар арасындағы ерекшелікті табу қиын. UML-да класс зоналарға бөлінген тіктөртбұрыш түрінде кескінделеді. Жоғары зонада класс аты, ортасында оның құрылымы (атрибуттар тізімі) оның астындағысында тәртіп сипаттамаларын анықтайтын функциялар беріледі.

Кластар диаграммасын алу тәртібі. Функционалдық және актерлермен функциялардың бірлесе қолданылуы. Ақпараттық элементтер және құрылымдар (АО) арасындағы қарым – қатынас және байланыс. Инкапсуляция, мұрагерлеу және АО полиморфизмі. Бірлесуі (уақыттық және жазықтықтық) және көрінуі (ішінде және сыртында). Декомпозициялық және құрылымдық. ҚВ, тізбекті, бірлескен (кооперативті) диаграммалары – орындалу тізбегі.

Класстар арасындағы қарым – қатынасты орнату. Агрегаттық қатынастар және мұрагер қатынастар. Қолдану қатынастары. Ассоциативті байланыстар. Бағыттық. Уақыттық және қалыпты байланыстар. Графикалық көрсетілуі.

Класстардың көрсетілуінің графикалық моделі. Класстар типі және жобадағы олардың көрсетілуі. Негізгі кластар, басқарушы кластар, интерфейстік кластар. Графикалық көрсетілімі. Әртүрлі нотациялар және олардың аспаптарда қолданылуы.

5 Программалық қамтамаларды өңдеудегі өмірлік циклды қолдаудың аспаптық құралдары

Мақсаты: Интерфейс құру әдістері және аспаптық құралдарын білу. Интерфейс элементтерінің активациясы және фокустың қолданылуы. Форманың модальдығы – артықшылықтары және кемшіліктері.

Аспаптық бағдарламалық қамтама құрудың теориялық негізі. Аспапты өңдеу мақсатын анықтау. Ұқсас прототиптер және олардың атқаратын қызметін анықтау. Прототиптерді таратудың артықшылықтары мен кемшіліктері. Өңдеу бағытын анықтау. Тарату құралдарын анықтау. Жұмыс топтарын және оларға жауаптыларды (менеджерлерді) құру. Өңдеу администраторын тағайындау. Логикалық модельдердің көрсетілуін құру және ақпаратты өңдеу. Жобаны тарату құралын таңдау. Өңдеу және құжаттау. Жұмысты параллельдеу – өңдеуді басқару. Тестерді өңдеу және өнімді тестілеу. ПӨ сертификациясы және таратылуы. ПӨ енгізу.

Аспаптардың бағдарламалық құрамы. Аспапты тарату программасы; көмек, лицензия және сатып алу шарты, үйрену жүйесі мысалмен бірге, инсталлятор, тестер және тестілеу жүйесі, бағдарлама пакетінің құжаттары.

Аспаптарды жобалау автоматизациясы. Визуальды құраушының ішкі қызметтерін қолдану. Контейнерлерді қолдану. Шаблондарды қолдану. Интерфейстерді қолдану және интерфейстік технологиялар.

Аспаптарды пакеттеу. Пакет құру үшін аспаптарды таңдау. Жаңа пакет құру және бар пакетке қосу. Пакеттеу аспаптары.

5.1 Help құру

Контекстік help. Контекстті Help құру. Контекстік Help`s құрушылар. Көмек құрамын анықтау. Құрылымдық көмек ұйымдастыруды анықтау. Визуализациялау және көмек файлымен жұмыс істеудің контекстік жұмысын қамтамасыздандыру.

Helper қосымшасы – бөлек бағдарлама, онсыз функцияларға толық немесе аралық қатынауға болмайды. Helper қосымшасы операциялық жүйенің немесе бағдарлама құрушының қолдауында болады. Мысалы helper қосымшасы - Server Manager бағдарламасының InterBase утилитасы, ол деректер базасын, пайдаланушыларды және қауіпсіздікті басқарады. Мұндай бағдарлама бағдарламалық өніммен бірге таратылады.

Көмек файлдарын құрушылар – олардың саны өте көп. MS- Help & Manual - аспабы, көмек файлдарын құруды жеңілдетеді және құжаттарын баспаға шығарады. Текстік редактор және ағаш бейнесі мен редакторлаудың жұмысын бірге орындайды, олар негізінен әртүрлі деңгейдегі енгізулерді тескстерден фрагменттер құруға мүмкіндік береді. Тұрып қалу және интуитивтік интерфейс құрылған файлмен жеңіл манипуляциялауға және қоюларды оңай орындауға мүмкіндік береді. WYSIWYG (What You See Is What You Get) интерфейсі сізді leaves қойылатын фрагмент типін көрсету

проблемасынан босатады және файл құрылымын құруға тоқталуға мүмкіндік береді. Яғни қою элементі басқарушы және бейнеленетін ақпараттарды көрсете алады. Редактор барлық стандарты Windows көмегінің форматтарын құра алады - HTML Help and Classic Winhelp, Word documents and PDF files. Сонымен, пайдаланушыға арналған көмекші құралды қосымшаның өзінен ақ құра аласыз. Барлық құжаттар Help & Manual жобасында құрылады, ол тақырыптар жайындағы ақпаратты, іздеу кілтерін, барлық жобалық шарттардың мазмұнын бір орындалатын файлда сақтауды орындайды.

Көмек файлын құру үшін: 1)Жаңа файл құру керек; 2) Көмек файлының тақырыбын жазу керек; 3) Құрылған тақырыпты егер керек болса басқа тақырыптармен гиперсілтеме арқылы байланыстыру керек; 4) Графикалық қоюларды және көрсеткіштерді қосу; 5) Макростар және OLE objects қолдану; 6) Аудио және бейне файлдарды көшіру немесе құру; 7) Қосымшаға контекстуальды байланыстыруды қолдану, ол үшін орындалатын файл құжатталады және көмек құрылады; 8) Құрылған жобаның тестіленуі.

Әрекеттер клиенттеріне таратылатын қасиеттер Егер бірнеше басқыштардың немесе меню пунктерінің бір ғана жалпы өңдегіші болса, онда олардың басқа қасиеттерінің де бірдей болуын талап еткен дұрыс. Delphi –де ол солай тартылған.

TActionManager компоненті (әрі қарай мәтінде — әрекеттер менеджері). Ол басқа да көптеген қосымша мүмкіндіктер береді.

TActionManager редакторының бірінші бетінде (екілік мауста басу арқылы немесе контексті менюдегі Customize командасы арқылы шақырылады) барлық панельдер тізімі тұрады, олар берілген әрекеттер менеджерімен байланысқан. Сіз жаңа компонент қосып немесе бұрынғы компонентті TActionToolBar басқышындағы New және Delete басу арқылы орындай аласыз.

Қолданушы интерфейсі - бұл жүйенің «беті» және пайдаланушының жүйемен жұмыс істеу тиімділігі оның ойлануына байланысты. Жұмыстың тиімділігіне әсер ететін факторлар жекелеген элементтер үшін нақты талаптар түрінде нашар ресімделеді, бірақ бағдарламалық қамтамасыз етудің пайдаланушылық интерфейсін құру бойынша жалпы ұсынымдар мен принциптер түрінде ескерілуі керек. Адам-машиналық өзара әрекеттесудің интерфейсін тестілеуді қолдануға ыңғайлылық деп атайды (орыс тіліндегі әдебиетте «ыңғайлылық» сөзі жиі қолданылатын терминнің аудармасы ретінде пайдаланылады).

Жүйенің стандартты интерфейсі. Интерфейсті өңдеу мысалы – MS Visual Systems, Borland Visual Systems, Oracle Visual Systems және т.б. Шешімі, артықшылықтары, кемшіліктері, мобильдігі, тереңдігі, түсі, өлшемі, анимациясы, масштабталуы, сенімділігі, қатынау жылдамдығы және т.б. Жүйенің стандартты интерфейсінің элементтері және олардың компоненттік таратылу.

Кірісу минимизациясы. Автоматизация деңгейі және компетенттік емес жағдайлардың сақтау. Әрекеттер тізбегі және сеанстары. Шаблондарды қолдану. Көптүрлілік және мобильдік көрсеткіштері. Жүйелерді қолдану, – аспаптарды, істік графиканы қолдану болып табылады.

Мұндай жағдай көп кездеседі. Мұндағы негізгі идея, ол қосылатын қосымша қосылар алдында ресурсты алып алу керек, ал қалғандары да оны орындағысы келеді бірақ ол бос болмағандықтан өз жұмыстарын аяқтайды да оның басқа қосымшалары жасалмайды.

Мұндай ресурстың мысалы ретінде жабдығы бейнеленетін файлдағы жалпы блокты алуға болады. Бұл ресурстың аты болғандықтан, оны өз қосымшыңызға уникальды етіп жасап алуыңызға да болады.

5.2 Программа инсталляциясы және қорғау талаптары

Инсталляциялау (орнату) - соңғы пайдаланушының компьютерінде бағдарламалық қамтамасыз етуді орнату процесі. Оны операциялық жүйеде (мысалы, GPU / Linux жүйесінде RPM және APT, Microsoft Windows жүйесіндегі Windows Installer) немесе бағдарламалық жасақтаманың өзі орнату құралы бөлігі болып табылатын арнайы бағдарлама (пакет менеджері) жүзеге асырады. GNU операциялық жүйесінде GNU құралдар тақтасының жүйесін және оның ұқсастығын орнату алдында бағдарламаны құрастыру үшін өте кең тараған.

Инсталляциялау, әдетте, бағдарламаның қажетті файлдарды файлдық жүйенің тиісті жерлеріне орналастыруды, сондай-ақ конфигурациялық файлдарды өзгертуді және жасауды қамтиды. Пакеттік менеджерлері тәуелділікті тексеруді орнатады, бағдарламаны іске қосу үшін жүйеде қандай да бір пакеттер бар ма, егер тексеру сәтті болса, қолжетімді пакеттер тізімінде жаңа пакетті тіркейді. Бұл үдеріс әр бағдарлама мен компьютер үшін әртүрлі болғандықтан, көптеген бағдарламалар (операциялық жүйелерді қоса) әмбебап немесе арнайы орнатқышпен - оларды орнатуға қажетті ең көп жұмысын автоматтандыратын бағдарламамен бірге жеткізіледі.

Инсталляциялық пакет құру құралдары – Install Shield. Бірнеше файлдардан тұратын программалық және ақпараттық қосымшалар, олар компьютерге инсталляциялық процедуралар көмегімен орнатылады. Инсталляция тіркелу, ақпараттық және орындалатын файлдарды орналастырудан тұрады. Олар енгізілетін пакеттің қосымша өңдеу аспаптарынның денесіне құрылатын ПӨ дұрыс функциялануын қамтамасыздандырады. Өңдеуші асапаптарына әдетте инсталляциялық пакет құру үшін арналған құралдар кіреді. Сондай – ақ Delphi – де де InstallShield Express аспаптық пакеті бар, ол Delphi – дің басқа қосымша аспаптарымен жұмыс жасау үшін сертификацияланған, соның ішінде BDE сияқты. Ол Windows Installer (MSI) technology негізінде жұмыс жасайды. Пакет инсталляциялық программалар құру үшін қолдан инсталлияцияланған болуы керек. GNU, Linux және BSD-ге негізделген амалдық жүйелердің көп бөліктері амалдық жүйенің қажетті құрамдас бөліктері мен үшінші тараптың бағдарламалық жасақтамасын орнатуға болатын жиі-жиі басқарылатын пакеттік басқару жүйелеріне ие, көбінесе тіпті егер ол өз орнатушысын пайдаланбаса да, ол қарастырылмайды.

Орнату, әдетте, бағдарламаның қажетті файлдарын файлдық жүйенің тиісті жерлеріне орналастыруды, сондай-ақ конфигурациялық файлдарды өзгертуді және жасауды қамтиды. Бума менеджерлері тәуелділікті тексеруді орнатады, бағдарламаны іске қосу үшін жүйеде қандай да бір пакеттің бар-жоқтығын тексереді, ал егер орнату сәтті болса, жаңа буманы бар пакеттер тізімінде тіркейді.

Mac OS X-де пакетті басқару жүйесі қолданылады. Mac OS X үшін кейбір коммерциялық қолданбалар бөлек орнатқышты пайдаланады, мысалы, Орнатушы VISE немесе Stuffit InstallerMaker. Қосымша жүйелік компоненттерді орнатудың қажеті жоқ қолданбалар жай қосымшалардың файлдарын қатты дискідегі қажетті орынға көшіру арқылы орнатылуы мүмкін. Mac OS X-де Бағдарламалық жасақтаманы жаңартуға арналған жеке бағдарлама (бағдарламалық жасақтаманы жаңарту шлюзі ретінде де белгілі), бірақ ол тек Apple бағдарламасының өнімдерін қолдайды.

Кейбір компьютерлік бағдарламалар өздерінің файлдарын дұрыс жерге көшіру арқылы орнатылады. Мұндай бағдарламалар орнатуды талап етпейді. Атап айтсақ, бұл Mac OS X, DOS және Microsoft Windows бағдарламалары арасында кең таралған. Орнатуды қажет етпейтін операциялық жүйелер бар, сондықтан пайдаланушы компьютерінде орнатылған басқа операциялық жүйелерге әсер етпей, жүктеу ықшам дискісінен, DVD немесе USB құрылғысынан тікелей іске қосуға болады. Осындай ОЖ-ның мысалы - Knoppix немесе Mac OS 1-9. Кейбір компьютерлік бағдарламалар өздерінің файлдарын дұрыс жерге көшіру арқылы орнатылады және орнату процесі осында жоқ. Қолмен орнату – орнатушы орнатпастан немесе пайдаланушы қолмен орындалатын операциялардың елеулі санынсыз орындалады.

Silent installation - хабарламалар немесе терезелер көрсетілмейтін орнату. «Silent installation» сөзі автоматты түрде «автоматты түрде орнатылуымен» синоним емес, дегенмен ол жиі қате қолданылады.

Автоматты орнату - бұл, әрине, оны іске қосу процедурасын қоспағанда, пайдаланушы араласуынсыз орындалатын орнату. Орнату процесі кейде таңдау жасау арқылы орнату процесін басқаратын пайдаланушымен өзара әрекеттесуді талап етеді: пайдаланушы келісімін қабылдау, параметрлерді орнату, құпия сөздерді анықтау және т.б. Графикалық орталарда орнату шебері деп аталатын орнатушылар пайдаланылуы мүмкін, бірақ олар автоматты түрде автоматты түрде орнатуға мүмкіндік беретін пәрмен жолы параметрлерін ұсынады.

Өзін-өзі орнату - бұл процесті бастапқы іске қосуды талап етпейтін орнату. Мысалы, Vodafone Mobile Connect USB модемі, ол қолмен іске қосуды қажет етпей оған қосылған кезде компьютердің USB портынан орнатылады.

Қашықтан орнату - бұл пайдаланушы компьютеріне қосылған мониторды пайдаланбай орындалатын орнату (атап айтқанда, бейне шығыссыз компьютерде жұмыс істеу). Бұл жергілікті желі арқылы немесе сериялық кабель арқылы қосылған басқа машинадан басқарылатын орнату болуы мүмкін. Автоматты және қашықтағы қондырғылар - жүйелік әкімшілер орындаған қарапайым операциялар.

«Таза» орнату - бағдарламадан бағдарламаға қарай өзгеруі мүмкін

осындай факторлар болмаған кезде орындалған орнату. Әдеттегі орнатудың күрделілігін ескере отырып, оның табысты нәтижесіне әсер ететін көптеген факторлар бар. Атап айтқанда, сол бағдарламаның бұрынғы орнатымынан қалған файлдар немесе амалдық жүйенің тұрақсыз жағдайы бағдарламаның дұрыс орнатылуына және жұмыс істеуіне әкелуі мүмкін.

Тікелей орнату - бұл түпнұсқалық медиадан (әдетте CD немесе DVD) емес, қатты дискіге (тегіс көшірме деп аталады) көшірілген бағдарламадан орнату. Бұл мақсатты машина бағдарламаларды орнатқан кезде үлкен CPU жүктемесін тудыратын тапсырмаларды орындау кезінде оптикалық дискілерден оқуға арналған кездейсоқ қол жеткізуге қол жеткізе алмайтын жағдайларда пайдалы болуы мүмкін.

Бағдарламалық жасақтама қайта жіберіліп, қалпына келтірілді. Деректерді пайдаланып деректерді жіберу және жіберу үшін бақылау нүктесін жасаңыз. Нысан ағашындағы бұта нысандарын шығарыңыз.

Бағдарламалық жасақтама әзірлеушілеріне ресурстарды (оның ішінде уақытша) жетіспеуіне байланысты проблема бар. Олар өздерінің тұрақты қорғауын жүзеге асыру үшін жеткілікті уақыт, қаржы немесе біліктілікке ие болмауы мүмкін. Олар бағдарламалық қамтамасыз етуді үшінші тараптың автоматты қорғау құралдарын пайдалануға мәжбүр. Бұл құралдар қауіпсіздік модулін құрастырылған бағдарламаға орнатады. Осындай қорғаудың артықшылығы - кез-келген бағдарламаға орнатылуы мүмкін (бағдарламаның бастапқы кодынсыз да). Өз тұрғысында кемшіліктер «үлгі» әдісі болып табылады. Стандартты қорғаныстың бұзылуы ықтимал, себебі олар бірнеше бағдарламаларда орнатылып, осылайша хакерлік нарықта сұранысты қамтамасыз етеді.

Алайда, автоматты құралдар бағдарламаны бұзуды қиындатады. Кейде оларды ешқандай қорғаныс болмаған кезде немесе өздерінің бірегей қорғанысын іске асырумен бірге пайдалану ұсынылады.

6 Программа интерфейсін құру

Мақсаты: Юзабилити мүмкіншіліктерін талдау және бағдарламаны басқаруды жеңілдету. Юзабилитиге қойылатын негізгі талаптарды және мәселелерді талдау. Қолдану ыңғайлылығының факторларын қарастыру.

Жүйенің стандартты интерфейсі. Аралас минимизация. Экранға орналастыру. Бір және көпбеттік интерфейс. Модальды терезелер және фокус. Интерфейсті визуальды жобалауды қамтамасыздандыру технологиялары. Бояу. Терезелердің ауысуы және олардың орналасу реті. Сілтемелер ұйымдастыру. Интерфейске инженерлік психология және эргономика талаптары. Визуальды дизайнерлердің интерфейстік объектілері және олардың интерфейс құрудағы қолданылуы. Қасиет редакторын құру. Компонент редакторлары. Қасиеттер категориясы. Windows қабықшасының кеңейтілуі –COM объектілер мастері, контексті менюді ауыстыру өңдегіштері, пиктограммалар. Open Tools API интерфейстері. Мастерлер құру.

TActionList компоненті, бұл — орталықтандырылған сақтау, мұнда падйланушы жағынан әрекеттер олармен оындалатын реакциялармен байланысады.

Әрекет (Action) деп операцияны атаймыз, оны пайдаланушы интерфейс элементтеріне әрекеттесіп жасайды. Әрекет орындалу керек компонент әрекет (Action target) мақсаты деп аталады. Әрекет инициализацияланған компонент (батырма, пункт меню), — әрекет клиенті (Action client) деп аталады.

Ескерту Барлық әрекеттер бір компоненттер TActionList немесе TActionManager тізіміне біріккен кезде ғана жұмыс істей алады. Бұл компоненттердің сыртынан ешбір әрекет орындалмайды.

Бірінші жұмыс формаға TActionList компонентін орнатудан басталады. Ол бірінші Standard бетіндегі компоненттер палитрасында орналасқан. Бұдан кейін әрекеттер тізімінің редакторын маустың басқышын екі рет компонент немесе контекстік менюді басу арқылы жіберу керек. Әрекеттермен байланысты уақиғалар TAction компоненті үш уақиғаға көңіл бөледі: OnExecute, OnUpdate , OnHint.

Бірінші — және бұл негізгі — берілген әрекетке берілетін реакция. Бұл уақиға басқышты басу кезінде орындалады. Мұнда — тәртіпке байланысты орындалады. Тәртіпке байланысты деп отырғанымыз, ол сигналды өңдеу сұлбасы бойынша орындалады, оның 4 - кезеңі бар:

1. Бірінші әрекет тізімінен OnExecute уақиға өңдегіші шақырылады

TActionList:

Property OnExecute: TActionEvent; TActionEvent = procedure (Action: TBasicAction; var Handled: Boolean)

of object;

Егер бұл уақиғаның өңдегіші сізбен қаралмаса онда келесі уақиға генерациясы орындалады — қадам 2.

2. Глобальды Application обьектісінің onActionExecute уақиғалар өңдегіші шақырылады (уақиға типі алдындағыдай — TActionEvent). Егер әрекет сигналын орындамаса онда келесі қадамға өтеміз.

3. Әрекеттің өзінен onExecute уақиға өңдегіші шақырылады (объект типі TAction немесе мұраға алынады).

4. Егер бірінші үш қадам жағдайды өңдемесе (False болса), онда, мүмкін, бұл дұрыс қойылмаған мақсатқа байланысты болған шығар (Target). Соңғы шанс ретінде қосымшаға CM_ACTIONEXECUTE хабарламасы жіберіледі. Бұл жағдайда бұл үшін басқа мақсат ізделеді.

onupdate уақиғаларды енгізу. onupdate уақиғаларды қолдануға мысал:

procedure TForml.PasteActionUpdate(Sender: TObject); begin

TAction(Sender).Checked := Clipboard.HasFormat(CFJTEXT);

End.

Ескерту onupdate уақиғаларды шақыру кезінде де 4 –кезеңдік әрекеттер тізбегі орындалады, ол алдыңғы OnExecute –ге ұқсас.

Үшінші уақиға типі келесідегідей:

THintEvent = procedure (var HintStr: string; var CanShow: Boolean) of object;

Ол басқару элементінен берілген әрекетке байланысты жауаптар керек болған кезде шақырылады. Уақиғалар өңделгенше бірнәрсе шығатын шықпайтындығын көрсетуге болады (CanShow параметрі) және, егер шықса онда не шығатындығы (Hintstr параметрі).

Бұлар Taction компонентіне байланысты уақиғалар болған. Ал TActionList компонентінің өзінің үш уақиғасы бар: OnExecute, OnUpdate жәнеOnChange. Бірінші екеуін біз қарастырдық; ал үшінші тізімді өзгерту кезінде орындалады (әрекеттерді қосу немесе жою).

Әрекеттер клиенттеріне таратылатын қасиеттер Егер бірнеше басқыштардың немесе меню пунктерінің бір ғана жалпы өңдегіші болса, онда олардың басқа қасиеттерінің де бірдей болуын талап еткен дұрыс. Delphi –де ол солай тартылған.

TActionManager компоненті (ары қарай тексте — әрекеттер менеджері). Ол басқа да көптеген қосымша мүмкіндіктер береді.

TActionManager редакторының бірінші бетінде (екілік мауста басу арқылы немесе контексті менюдегі Customize командасы арқылы шақырылады) барлық панельдер тізімі тұрады, олар берілген әрекеттер менеджерімен байланысқан. Сіз жаңа компонент қосып немесе бұрынғы компонентті TActionToolBar басқышындағы New және Delete басу арқылы орындай аласыз.

Қолданушы интерфейсі - бұл жүйенің «беті» және пайдаланушының жүйемен жұмыс істеу тиімділігі оның ойлануына байланысты. Жұмыстың тиімділігіне әсер ететін факторлар жекелеген элементтер үшін нақты талаптар түрінде нашар ресімделеді, бірақ бағдарламалық қамтамасыз етудің пайдаланушылық интерфейсін құру бойынша жалпы ұсынымдар мен принциптер түрінде ескерілуі керек. Адам-машиналық өзара әрекеттесудің интерфейсін тестілеуді қолдануға ыңғайлылық деп атайды (орыс тіліндегі әдебиетте «ыңғайлылық» сөзі жиі қолданылатын терминнің аудармасы ретінде пайдаланылады).

Жүйенің стандартты интерфейсі. Интерфейсті өңдеу мысалы – MS Visual Systems, Borland Visual Systems, Oracle Visual Systems және т.б. Шешімі, артықшылықтары, кемшіліктері, мобильдігі, тереңдігі, түсі, өлшемі,

анимациясы, масштабталуы, сенімділігі, қатынау жылдамдығы және т.б. Жүйенің стандартты интерфейсінің элементтері және олардың компоненттік таратылуы.

Кірісу минимизациясы. Автоматтандыру деңгейі және компетенттік емес жағдайларды сақтау. Әрекеттер тізбегі және сеанстары. Шаблондарды қолдану. Көптүрлілік және мобильдік көрсеткіштері. Жүйелерді қолдану, – аспаптарды, істік графиканы.

Мұндай жағдай көп кездеседі. Мұндағы негізгі идея, ол қосылатын қосымша қосылар алдында ресурсты алу керек, ал қалғандары да оны орындағысы келеді бірақ ол бос болмағандықтан өз жұмыстарын аяқтайды да оның басқа қосымшалары жасалмайды.

Мұндай ресурстың мысалы — жадыға бейнеленетін файлдағы жалпы блок. Бұл ресурстың аты болғандықтан, оны өз қосымшыңызға уникальды етіп жасауыңызға болады:

```
var UniqueMapping : THandle;
FirstWindow : THandle
; begin
UniqueMapping := CreateFileMapping($ffffffff,
nil, PAGE_READONLY, 0, 32,'MyMap');
 if UniqueMapping = 0 then
 begin ShowMessage(SysErrorMessage(GetLastError)); Halt; end
else if GetLastError = ERROR_ALREADY_EXISTS then
  begin  FirstWindow := FindWindowEx(0, 0, TfmMain.ClassName, nil);
    if FirstWindowoO then  SetForegroundWindow(FirstWindow}; Halt;
end;
// Application.Initialize қосмышасының басқа көшірмесі жоқ;
```

Бұндай жолдарды жоба формасын құрудан бұрын текстің басын орныластыру керек.

Жадымен бірге қолданылатын блок жүйелілік бет файлында беріледі (бұл жайында бірінші параметр айтып тұр, ол -1 тең, функция сипаттамасын қара.

CreateFileMapping). Оның аты — муМар. Егер блокты құру кезінде қате коды алынса ERROR_ALREADY__EXISTS, онда ол қосымшаның жұмыс істеп тұрған көшірмесінің бар екендігін білдіреді.

Экранға орналастыру. Бір- және көпбеттік интерфейс. Модальды терезелер және фокус. Панельдер, фреймдер, ағаштар, графтар, желілер – уақиғалық парадигмалардың визуальды бейнеленуінің ақпараттық моделі. Концентрация және интерфейстің таратылу жалпыламасы. Интерфейстің көпбетілігі және көппроцессорлығы. Бейнелерді өзгерту – программадағы әрекеттер. Интерфейс элементтерінің активациясы және фокустың қолданылуы. Форманың модальдығы – артықшылықтары және кемшіліктері.

Интерфейсті визуальды жобалауды қамтамасыздандыратын технологиялар. Бояу. Операциялық жүйенің графикалық интерфейсінің кітапханасы.Визуальды программа құрушылардың графикалық кітапханасы. Графикалық бағытталған жүйелер. Графикалық файлдардың форматы – графикалық ақпарат жеткізгіштер. Экспорт – графикалық форматтардың

импорты. Микропроцессордың графикалық мүмкіндіктері – деректер типі, операциялар, видеожүйелер.

Юзабилити, сондай-ақ қолдану ыңғайлылығы, эргономикасы - пайдаланушыға белгілі бір жағдайларда түсінуге, зерделеуге, қолдануға және тартымды етуге қабілеттілігі; жүйенің, өнімнің немесе қызметтің сипаты, белгілі бір пайдаланушысы белгілі бір жағдайларда белгілі бір мақсаттарға қол жеткізу үшін жүйені қажетті нәтижемен, тиімділікпен және қанағаттандырумен басқаруы мүмкін.

Жүйені пайдаланудың ыңғайлылығы (жарамдылығы) тек қана жұмыс істеу оңайлылығымен шектелмейді. ISO 9241 сериясының стандарттарына сәйкес, бұл сипаттаманы пайдаланушының жеке мақсаттарын, жүйенің қабылдауына байланысты эмоцияларын және сезімдерін, сондай-ақ жұмысқа қанағаттандыруды ескере отырып, неғұрлым кең түсіну керек. Қолданудың ыңғайлылығын қамтамасыз ету үшін қажетті қасиеттер сондай-ақ тапсырмаға және қоршаған ортаға байланысты. Қолданудың жарамдылығы абсолютті тұжырымдама емес, белгілі бір пайдалану шарттарында әртүрлі тәсілдермен көрінуі мүмкін.

6.1 Юзабилитиге қойылатын негізгі талаптар және мәселелер

Жүйелік статустың көрінуі

Пайдаланушы қолайлы уақытта кері байланыс ала отырып, не болып жатқанын білуі керек.

Жүйе мен нақты әлем арасындағы сәйкестік.

Жүйе «жүйеге бағдарланған» тілге қарағанда түсінікті терминология мен тұжырымдамаларды қолдана отырып, «пайдаланушының тілінде сөйлеуі» керек.

Қолданушы үшін басқарушылық және еркіндік

Пайдаланушы жүйелік функцияларды жиі қателесіп таңдайды және күрделі диалогты қажет етпейтін қажетсіз жүйе күйінен анық көрінетін «апаттық шығу» болуы керек. Болдырмау(undo) және қайталау(redo) функцияларын қолдау керек.

Сәйкестік және стандарттар

Пайдаланушылар сол сөздерді, жағдайларды немесе әрекеттердің бірдей, бірдей еместігін болжамау керек. Сондай-ақ, осы платформаға қабылданған келісімдерді ұстану қажет.

Қателерді болдырмау.

Кез-келген қиындықты туғызбайтын ойластырылған дизайн. Қателердің өз жағдайларын жою керек, немесе оны анықтау және пайдаланушыға алдағы проблема туралы ескерту қажет.

Икемділік және пайдаланудың тиімділігі.

Тәжірибелі қолданушы үшін тезірек өзара әрекеттесуді тездететін жеделдеткіштер,яғни Акселаторлар (командалардың жылдам орындалу құралдары),тәжірбиесіз қолданушылар байқамайды. Сондықтан жүйе

тәжірибесіз де, тәжірибелі пайдаланушыларды да қанағаттандыруы керек. Жиі пайдаланылатын әрекеттерді теңеу мүмкіндігі болуы керек.

Эстетикалық және минималистік дизайн

Интерфейсте пайдаланушының көп жағдайларда қажет емес немесе қажет етпейтін ақпарат болмауы керек. Әрбір артық диалог элементі қажетті элементтерге назар аударуға кедергі келтіреді.

Қатені түсінуге және түзетуге көмектесу

Қате туралы хабарлар қарапайым тілде жазылған, кодсыз, проблеманы нақты анықтап, сындарлы шешімді ұсынуы керек.

6.2 Қолдану ыңғайлығының факторлары

Сайттың ыңғайлылығы - интерфейспен пайдаланушының жұмысының ыңғайлығын бағалау. Сол тұжырымдама жобалау сатысында ресурсты жетілдіруге бағытталған әдістер деп аталады.

Дұрыс салынған және түсінікті бетті дүкен терезесімен салыстыруға болады. Осылайша, интерфейсті түсіну және дамыту қажет, ол келушінің түсініктемесіне, қайда және ол не істеу керек екенін түсіндіреді. Бұдан басқа, ешкім де қызығушылығын жоймайды, себебі адам жай ғана кетіп қалады. Мұның негізгі себебі - ережелерді елемеу. Сайып келгенде, іздеу көп уақыт алады және ол тиімсіз болады.

Сайттың ыңғайлылығын дұрыс ұйымдастыруға көмектесетін негіздер үшін жақсы критерийлер бар. Оларға мыналар жатады:

Қарапайым оқыту. Яғни, келушінің сайтты игеру мүмкіндігі қаншалықты жылдам болады. Есте сақтау қабілеті. Егер сіз қайтадан барсаңыз, сайтыңызды есте сақтау оңай. Жұмыс кезінде қателер. Пайдаланушылар жиі қателіктер жібереді және осы кемшіліктердің қаншалықты маңызды екендігін біледі.

Қанағаттану. Келуші Сіздің жобаңызбен жұмыс істегенді ұнатады.

Жоғарыда келтірілген кеңестер мен принциптерді ескере отырып біз сіздермен біртіндеп ыңғайлылықтың негізгі ережелеріне жеттік. Жүйеге әсер ететін критерийлерді егжей-тегжейлі қарастырайық.

Веб-дизайн жағымды және әдемі болуы керек, сондықтан оны пайдаланушы ұнатады. Мазмұн оңай оқылып, қабылдануы мүмкін, ал пайдаланушылар ұзақ уақыт бойы олардан шаршамауы керек. Мен қараңғылықты жеңілдетуді және қара әріптерді ұсынуды ұсынамын, бұл, менің ойымша, ақпаратты қабылдауды жақсартады. Ең дұрысы, сайтты басқалардан ерекшелендіріңіз.

Пайдаланудың сапасы жүктеу жылдамдығынан үлкен маңызға ие, өйткені адамдар ұзақ уақыт бойы күтуге алмайды, керісінше басқа ресурсқа барады. Келушілерді сақтау үшін жылдамдықты сақтау керек. Жүктеу орналасу, оңтайландыру, хостинг және басқа да факторларға әсер етеді. Өзімнен жоғары сапалы және сенімді хостингті таңдауды ұсынамын, менде бұл туралы жеке бап бар.

Ұялы телефондар мен планшеттерді қоспағанда, сайт түрлі браузерлерде

тез және тиімді жұмыс істеуі керек. Мұны істеу үшін әртүрлі браузерлерді орнатып, олардың өз ресурстарын тексеріңіз. Егер бәрі жақсы көрінсе, онда бұл жақсы.

Ақпараттық жоспардағы мазмұнның сапасы ең жоғары деңгейде болуы керек. Мақалалар тиісті және пайдалы болуы керек, мәтінде қателіктер мен кемшіліктерді толығымен жояды. Бұл жағдайда сайт көбінесе жаңартылып, көбірек жаңа материалдар жарияланатын болады, демек, жоғары қатысу және соғұрлым дұрыс seo.

Навигация және пайдаланушыға ыңғайлы интерфейс, әдетте, бөлек тақырып болып табылады. Менің ойымша, менімен келісесіз, қарапайым және түсінікті веб-сайттарда қажетті ақпараттың сағаттары бойынша іздеуден әлдеқайда ыңғайлы. Келуші қай жерде басу керектігін көруі керек. Мен жоғарғы жағында көлденең менюді, торап жолағын және төменгі мәзірді ұсынамын. Ең жеңіл шарлау үшін «up» түймесі пайдалы. Барлық бөлімшелері мен мақалалары бар сайт картасы әсіресе пайдалы.

Көптеген жарнама келешектегі келушілерге жағымсыз әсер етуі мүмкін. Сондықтан ол ақылға қонымсыз болуы керек. Сондай-ақ, қалқымалы қалқалар да келушілерді қуантып, қорқытпайды.

Сіз қай сайтқа жеке есептелінетін боласыз? Әрине, байланыс туралы ақпарат болған жағдайда, «Компания туралы» қосындысы, әртүрлі куәліктер, шолулар мен деректемелер (интернет-дүкен болса). Міндетті түрде бұл туралы қамқорлық жасаңыз, сонда келушілер сізге сене бастайды.

Бұл критерий негізгі белгіден кейінгі ең маңыздысы. Іздестірудің арқасында, ол келіп отырған ақпараттың барлығын бірден табады және сайтта қалады.

Келушілердің болуын мүмкіндігінше қарапайым етіп, яғни 3 рет басу ережесін, 1 секундтық ережені, дұрыс пирамиданы, элементтердің жақындығын және басқа идеяларды пайдаланыңыз.

Сайттың мобильді нұсқасы бар болса, оны одан да тартымды етеді. Қазір пайдаланушылардың көпшілігі ұялы телефон көмегімен интернетте отырады, сондықтан олар кез-келген ыңғайлы жерде бұл ақпаратты жылдам таба алады.

Тақырыптар нақты сипатта болуы керек, сонымен қатар тақырыптың мазмұнын және мазмұн сипаттамасын толығымен көрсетуі қажет. Сілтеме бойынша келген келуші не нәрсеге байланысты екенін бірден түсінуге тиіс. Іздеу механизмдері беттердің өзектілігін және мазмұнын сканерлеуде әлдеқайда жылдам, бұл тақырыптардың орындылығын анықтайды.

Түрлендіруді көтеріңіз. Шарлау пайдаланушыларға кез келген әрекеттерді орындауға мәжбүр етеді, мысалы, тапсырыс беру түймешігін басыңыз. Пайдаланушы бір минутқа созылса, оны жоғалтуға болады.

Сабаққа қатысу үлесін арттыру. Бірде-бір құрылымды мүлдем жоқ блокқа қайта жібергіңіз келе ме, түсініксіз навигациямен және мәзірлерсіз шатастырып жатырсыз ба? Әрине, жоқ. Мен сізбен келісемін. Егер сіз сайтта ыңғайлы, өзіңізді жайлы сезінсеңіз және тек оң эмоцияларға барсаңыз, онда сіз осында қайта оралғыңыз келеді.

Сайттың ыңғайлығын талдау өте маңызды, себебі бұл интерфейспен

әртүрлі ықтимал мәселелерді тексеруге мүмкіндік беретін процесс, сондай-ақ оны пайдалануды жақсартады. Қолдану аясын бағалау сандық және сапалы болып табылады.

1-ші тақырып бойынша тест сұрақтары

1. AllFusionERwin Data Modeler (ERwin) мүмкіндігі:
A) Қиын құрылым деректерін көрнекі бейнелеу
B) Кластар және объектілерді көрнекі бейнелеу
C) Кластарды бірнеше рет қолдану
D) Модельді өңдеуді жеңілдету
E) Кластар және объектілерді өңдеуді жеңілдету
****

2. Жөндегіш:
A) Вирустық шабуылдан жүйені қорғауға арналған бағдарлама
B) Файлдық құрылымдарды басқару және құру үшін арналған орындамалық қабықша
C) Графикалық файлдарды жөндеу және құру бағдарламасы
D) Берілген нүктелерде тоқтауға, ағымдағы айнымалылардың мәндерін қарауға және олардың мәндерін өзгертуге мүмкіндік беретін бағдарлама
E) Орындамалық жүйені баптауға арналған жүйелік бағдарламалық қамтамасыздандыру
****

3. Бағдарламаны өңдеудің аспаптық құралдары:
A) Жаңа бардарламаларды құру құралдары
B) БҚ жүктеу құралдары
C) БҚ тестілеу құралдары
D) БҚ өңдеудің техникалық аспаптық құралдары
E) БҚ өңдеудің аналитикалық құралдары
****

4. Модельдеу:
A) Графикалық объектілер жиынтығы
B) Мәтіндік объектілер жиынтығы
C) Модельдеу тілінің семантикасы
D) БҚ өңдеудің аналитикалық құралдары
E) Объектілер тәртіптерінің жиынтығы
****

5. UML мәндерінің түрлері:
A) Объектілі-бағытталған
B) Бағдарламалық
C) Аннотациялық, топтастырушы, құрылымдық
D) Графикалық
E) Модульдік
****

6.UML қолданылатын қатынас түрлері:
A) Құрылымдық
B) Жалпылау
C) Тиянақтау
D) Жинап алу

E) Семантикалық
****
7. UML құрылыс блогы:
A) Диаграммалар
B) Кластар
C) Түйіндер
D) Компоненттер
E) Прецедент
****
8. Бағдарламалық қамтамасыз ету келесі кластарға бөлінеді:
A) Жүйелік БҚ және өңдеудің аспаптық құралдары
B) Жүйелік БҚ, қолданбалы БҚ және бағдарламаны өңдеудің аспаптық құралдары
C) Жүйелік БҚ және қолданбалы БҚ
D) Орындамалық жүйелер, қолданбалы БҚ, утилиттер және драйверлер
E) Жүйелік БҚ, қолданбалы БҚ және жүйелік бағдарламалау
****
9. BPwin 4.0 есеп берулер таратылған келесі форматтарға экспорттала алады:
A) IBM Rational
B) Мәтіндік
C) Internet Explorer
D) Acrobat
E) Символдық
****
10. BPwin DFD диаграммасында келесі түрдегі шекаралық бағдаршаларды қолдануға мүмкіндік береді:
A) Қарапайым шекаралық бағдарша
B) Механизм бағдаршасы
C) Бақылау бағдаршасы
D) Компоненттер
E) Прецедент
****
11. ModelMart бағдарламалық өнімі келесі міндеттерді шешуге мүмкіндік береді:
A) Модельдерді басқару
B) Кітапханаларды бірлесіп пайдалану
C) Жаңа кластар құру
D) Шешімдер кітапханасын құру
E) Бағдарламаны модельдеу
****
12. Компилятор:
A) Бағдарламаның бір жолын машиналық кодқа ауыстырып бірден оларды орындаушы
B) Қолданбалы бағдарламалық қамтамасыз ету
C) Жоғары деңгейдегі бағдарламалау тілінде жазылған бағдарламаларды

машиналық тілдегі бағдарламаларға оның орындалуын қатыспай ауыстыратын бағдарлама
D) Баспа орталықтарында қолданылатын
E) Жүйелік БҚ арнайы утилитасы
****
13. ERwin диаграммалар қандай блоктардан тұрады:
A) Класс
B) Объект, атрибут, байланыс
C) Компоненттер
D) Қатынастар
E) Қасиеттер
****
14. БҚ өмірлік циклінің модельдері:
A) Баспалдақтық модель
B) Аралық бақылауы бар қадамдық модель
C) Итерационды модель
D) Спиральды модель
E) Кезеңдік модель
****
15. Архитектураны жобалау нәтижесі болып табылады:
A) Кластар моделі
B) Административтік интерфейс моделі
C) Пайдаланушы интерфейсінің моделі
D) Компоненттер моделі
E) Ағындар моделі
****
16. RAD әдістемесі бойынша БҚ өмірлік циклі келесі фразалардан тұрады:
A) Талаптарды жоспарлау және талдау фразасы
B) Тексеру фразасы
C) Тексеру фразасы және ендіру фразасы
D) Өц фразасы
E) Жоспарлау фразасы
****
17. UML нотациясы тізбек диаграммасы хабарламасының стереотиптері:
A) “create process” (процесс құру)
B) “paste”(қою), “copy”(көшіру)
C) “destroy”(құрту), “send”(жіберу)
D) “delete” (жою)
E) “enter” (кіру)
****
18. UML нотациясындағы кластың негізгі стереотиптері:
A) component (компонент)
B) class (класс)
C) entity (мән)
D) Деректер жинағышы (жинағыш)

E) Сыртқы мәндер (ақпаратты өңдеуге қатысатын обьектілер)
****
19. Обьектілі-бағытталған талдау және жобалау аспаптары:
A) Erwin
B) Designer
C) BPwin
D) JAM
E) Rational Rose
****
20. CASE–құралдарының функционалды мінездемелері:
A) Технологиялық орта
B) БҚ техникалық құралдар
C) Жобаны басқару
D) Жалпы функциялар
E) Модельдеу құралдары
****
21. CASE–құралдарды класификациялау түрлері:
A) ДҚЖБ шығу түрі бойынша
B) БҚЕ бойынша
C) Қолданылатын әдістемелер және жүйе моделі әрі ДҚ бойынша
D) Қолжетімді әдістемелер бойынша
E) Модельдеу әдістемесі бойынша
****
22. Имитациондық модельдердің класификациясы:
A) Үздікті немесе дискретті емес
B) Анықталмаған немесе тізбектелмеген
C) Тармақталған және анықталған
D) Статикалық немесе динамикалық
E) Дискретті және тізбектелген
****
23. БҚ ӨЦ ұйымдастырушы процесстері
A) Құжаттау, конфигурациясын басқару
B) Сапамен қамтамасыздандыру, верификация, аттестация
C) Модельдерді басқару
D) Қолдау
E) ӨЦ анықтау, бағалау және жақсарту
****
24. Rational Rose көмегімен өнделетін жобалар келесі құжатты қалыптастырады:
A) Деректер жинағышының спецификациялары
B) Өңделетін ақпараттық жүйенің моделін көрсететін ER диаграммалар
C) Деректер қорының спецификациялары
D) Объектілер, атрибуттар, кластар және операцияның спецификациялары
E) Бағдарлама мәтінінің дайындау құралдары
****

25. Use Case бейнелейді
A) Жүйе кластарына арналған тәртіптер түрлері
B) Объектілермен алынатын кейбір ақпараттар
C) Түйіндермен алынатын кейбір ақпараттар
D) Жүйенің жеке модуліне арналған тәртіптер түрлері
E) Прецеденттер жайында кейбір ақпараттар
****

26. Құрастырушы:
A) Бағдарламадағы синтакстік және семантикалық қателерді іздеу үшін арналған бағдарлама
B) Мәтіндік құжаттарды безендіру және құрастыру үшін арналған бағдарлама
C) Бөліп жасалған компиляция нәтижесіде алынған автоматты түрде іздеуі бар және қосылған кітапханалық ішкі бағдарламалар мен процедуралардан тұратын объектілік модульдерден жүктемелік модуль жинау бағдарламасы
D) Презентациялар құру үшін арналған бағдарламалық қамтамасыздандыру
E) Деректер қорын енгізу және құру үшін арналған бағдарламалар кешені
****

27. AllFusion ERwin Data Modeler (ERwin) мүмкіндігі
A) Кластарды бірнеше рет қолдану
B) Кластар және объектілерді өндеуді жеңілдеті
C) Деректер қорын өндеуді жеңілдету
D) Моделді өндеуді жеңілдету
E) Кластар және объектілерді көрнекі бейнелеу
****

28. Құрастырушы:
A) Мәтіндік құжаттарды безендіру және құрастыру үшін арналған бағдарлама
B) Кітапханалық бағдарламалар және стандартты ішкі бағдарламалармен бірге қоса алғанда бір немесе бірнеше объектілік модулдермен жүктемелік модул қалыптастырушы бағдарлама
C) Презентациялар құру үшін арналған бағдарламалық қамтамсыздандыру
D) Деректер қорын енгізу және құру үшін арналған бағдарламалар кешені
E) Бағдарлама
****

29. UML міндерінің түрі
A) Объектілік-бағытталған
B) Аппаратық
C) Бағдарламалық
D) Концептуалды
E) Аннотациялық және топтастырушы
****

30. UML қолданылатын қатынас түрлері
A) Бағдарламалық
B) Құрылымдық
C) Жинап алу
D) Тарату және ассоциация

E) Семантикалық
****
31. UML құрылыс блоктары:
A) Компоненттер
B) Қатынастар, мән, диаграммалар
C) Пакеттер
D) Түйіндер
E) Кластар
****
32. Жөндегіш
A) Трассировкамен қамтамасыз ете отырып жүктелетін бағдарлама тәртібін талдауға көмектеседі
B) Вирустық шабуылдардан жүйені қорғауға арналған бағдарлама
C) Графикалық файларды жөндеу және бағдарлама
D) Орындамалық жүйені баптауға арналған жүйелік бағдарламалық қамтамасыздандыру
E) Файлдық құрылымдары басқару және құру үшін арналған орындамалық қабықша
****
33. Бағдарламаларды өндеудің аспаптық құралдары :
A) БҚ тестілеу құралдары
B) Орындамалық жүйені баптауға арналған жүейлік бағдарламалық қамтамасыздандыру
C) Жаңа бағдарламаларды құру құралдары
D) БҚ өндеудің техникалық аспаптық құралдары
E) БҚ өндеудің аналитикалық құралдары
****
34. Функциональды модельдерді көрсететін аспаптық құралдарды атаңыз
A) BPwin
B) IDEFO
C) Rational Rose
D) JAM
E) MS visio
****
35. BPwin келесі түрдегі нотациялар арасындағы ауысулар қарастырылған
A) DFD->IDEFO
B) IDEFO->DFD
C) IDEF3->DFD
D) IDEF3->IDEFO
E) IDEF3->IDEF3
****
36. IDEF3 нотациясындағы модель келесі диаграммалардан тұрады:
A) Контексттік диаграмма
B) Қолдану варианттар диаграммасы
C) Экспозиция үшін арналған диаграмма

D) Процестердің орындалуының сценарийлер диаграммасы
E) Ағаштар диаграммасы
****
37. IDEF3 арнайы спецификациясы сілтеме объектілерінің стильдерін ерекшелейді – олар:
A) Синхронды (Synchronous)
B) Қатынастар (Relational Link)
C) Сілтеме объектілері - (Referent)
D) Объектілер процесі (Object Process)
E) Объектілер ағымы (Object Flow)
****
38. Интерпретатор:
A) Ағаштар диаграммасы
B) Мультимедия файлдарын құру бағдарламасы
C) Бүкіл бағдарманы кодқа бірден ауыстырып тәуелсіз орындамалық файл құрушы
D) Транслятор түрлерінің бірі
E) Электронды кестелерді жөндеу және құру үшін арналған бағдарлама
****
39. Бағдарламалаудағы объектілік тәсіл :
A) Объектілі – бағытталған бағдарламауға негізделген қиын БҚ құру технологиясы
B) Бағдарламаларды бірегей объект ретінде көрсетуге негізделген қиын БҚ құру технологиясы
C) Бүкіл бағдарламаны кодқа бірден ауыстырып тәуелсіз орындамалық файл құрушы
D) Қиын БҚ құру технологиясы бағдарламаларды ұйымдастырудың жаңа әдістеріне негізделген, олар мұрагерлену, полиформизм , композиция, толықтыру механизмдерінен тұрады
E) Модульдермен бағдарламауға негізделген қиын БҚ құру технологиясы
****
40. Деректер ағымының негізгі компоненттеріне қатысты емес :
A) Деректер ағыны
B) Процестер
C) Жұмыстар
D) Сыртқы мәндер
E) Деректер жинағышы
****

2-ші тақырып бойынша тест сұрақтары

1. Деректер ағынын модельдеуге қолданылатын нотация
A) IDEF3
B) IDEF0

C) IDEF2X
D) IDEF1X
E) DFD
****
2. Деректер қорын модельдеу кезінде қолданылатын нотация
A) IDEF1X
B) IDEF3
C) IDEF0
D) DFD
E) IDEF2X
****
3. Компьютер үшін программа бұл
A) Компьютер жұмысының жоспары
B) Жедел жады жұмысының жоспары
C) Есептеулерді орындау жұмысының жоспары
D) Процессор жұмысының жоспары
E) Пайдаланушы жұмысының жоспары
****
4. Өңдеу аспабы бұл
A) Визуальды редактор
B) Жұмысты орындауға арналған құрал
C) Клавиатура және монитор
D) Компьютер
E) Текстілік редактор
****
5. Case технологиясы бұл:
A) Программалау технологиясы
B) Программалаудың ақпараттық технологиясы
C) Программалық және ақпараттық аспаптар жиыны
D) Программалық құралдары өңдеудің компьютерлік технологиясы
E) Программалау тілдері
****
6. Фаза дегеніміз бұл
A) Процестің процеске қатынағандағы сұрату уақыты
B) Процестің ішіндегі уақыт аралығы
C) Процестің екі тірек нүктесінің арасындағы уақыт аралығы
D) Процесс пен процесс әрекеттескендегі жұмсалатын уақыт
E) Процестің ағындарға жұмсайтын уақыты
****
7. Диаграммалар бұл
A) Модельдің негізгі элементтері болып табылатын абстрация
B) Әртүрлі мәндерді байланыстыратын қатынастар
C) Төбелер және қабырғалардан тұратын байланысқан граф түрінде бейнеленетін элементтер жиынтығының графикалық көрсетілімі
D) Модельдегі құрылыс блогының статикалық ұйымдастырушы элементі

E) Әртүрлі қызығушылықтағы қатынастар
****
8. ERWin аспабында қолданылатын нотациялар
A) DFD, IDEF0, IDEF3
B) WORD, EXELL, NOTEPAD
C) RATIONAL ROSE, MODELMART, ARIS
D) IDEF1X, IE, DIMENSIONAL
E) IDEF1X, IDEF2X
****
9. Қандай бағдарламалық аспап өңдеудегі өзгертулерді қадағалап отырады
A) Debug
B) Rational clear case және Rational clear ouest
C) Request pro
D) BPWin және ERWin
E) Borland Delphi
****
10. BpWin бизнес-процестерді модельдеу аспабының DFD нотациясындағы құрылатын диаграмма
A) Деректер ағымының диаграммасы
B) Декомпозиция диаграммасы
C) Прецеденттер диаграммасы
D) Мән-байланыс диаграммасы (ERD)
E) Сценарийлер диаграммасы
****
11. Мүмкін болатын қиылысулар түрлерінің классификациясына жатпайтындар
A) Asynchronous OR
B) Synchronous AND
C) Asynchronous AND
D) Asynchronous TEST
E) Synchronous OR
****
12. RUP-та қолданылатын практикалар тізімі
A) Итерациялық өңдеу, талаптарды басқару, талаптарды қолдану, визуальды модельдеу, сапасын тексеру, талаптарды қадағалау
B) Талаптарды өңдеу, талаптарды басқару, архитектураны қолдану, визуальды модельдеу, сапасын тексеру, талаптарды қадағалау
C) Талаптарды басқару, модульдік архитектураны қолдану, визуальды модельдеу,сапасын тексеру, талаптарды қадағалау
D) Итерациялық өңдеу, талаптарды басқару, талаптарды қолдану, сапасын тексеру, өзгерістерді қадағалау,талаптарды орындау
E) Итерациялық өңдеу, талаптарды басқару, модульдік архитектураны қолдану, визуальды модельдеу, сапасын тексеру, өзгерістерді қадағалау
****
13. Қателер санын табуға арналған тестілік сценарийлер, процедуралар және метрикалар сипатталатын ағын

A) Тестілеу
B) Орындау ортасын талдау
C) Жобаны басқару
D) Бизнес-процестерді модельдеу
E) Талдау және жобалау
****
14. Компонентік программалаудың объектіліктен айырмашылығы
A) Жұмыс визуалдылығы
B) Сақталу жылдамдығы
C) Жадыда алатын орны
D) Жылдамдығы
E) Тарату тілінің әмбебаптығы және сыртқы тілдерінің алуан түрлілігі
****
15. UML бұл
A) Визуализациялау, спецификациялау,құрылымдау және программалық жүйелердің артефактілерін құжаттау тілі
B) Визуализациялау, құрылымдау және программалық жүйелерді программалау тілі
C) Алгоритмдеу, сұлбаларын салу, программаларын дайындау, кодын жазу және компиляциялау
D) Дайындау, орындату, тексеру, өндіріске енгізу, ашып қарау тілі
E) Алгоритмдеу, визуализациялау, спецификациялау және бағдарламалық жүйелерді бағдарламалау тілі
****
16. UML құрылыс блоктары
A) прецедент, тізбек, кооперациялар
B) мән, байланыс, диаграммалар
C) ашып қарау, енгізу, аяқтаулар
D) класс, компонент, пакеттер
E) күй, оқиға, жағдайлар
****
17. UML мәндерінің түрлері
A) Талаптар, топтастырулар, тестілеулер, қорытындылаулар
B) Автомат, топтастырушы, құрылымдық
C) Құрылымдау, тәртіптеу, орындаулар, тестілеулер
D) Құрылымдық, тәртіптеу, топтастырушы, аннотациялық
E) Тәртіптік, тестілеулік, енгізушілік, қорытындылау
****
18. UML құрылымдық мәндері бұл
A) Әрекеттесу, автомат
B) Ескертулер
C) Пакеттер, ескертулер,автомат
D) Кластар, интерфейстер, кооперация, прецедент, активті класс, компонент, түйін
E) Пакеттер

****
19. UML тәртіптік мәндері бұл
A) Кластар, интерфейстер, кооперация, прецедент, активті класс, компонент, түйін
B) Әрекеттесу, автомат
C) Пакеттер
D) Ескертулер
E) Пакеттер, ескертулер, автомат
****
20. UML топтастырушы мәндері бұл
A) Кластар, интерфейстер, кооперация, прецедент, активті класс, компонент, түйін
B) Әрекеттесу, автомат
C) Пакеттер, ескертулер, автомат
D) Ескертулер
E) Пакеттер
****
21. UML аннотациялық мәндері бұл
A) Әрекеттесу, автомат
B) Ескертулер
C) Пакеттер, ескертулер, автомат
D) Кластар, интерфейстер, кооперация, прецедент, активті класс, компонент, түйін
E) Пакеттер
****
22. Тәртіп бойынша бизнес- процестерді модельдеу CASE - құралдарының көмегімен орындалады, олар
A) DFD, IDEF0, IDEF3
B) WORD, EXELL, NOTEPAD
C) BPWin,ERWin, RATIONAL, ROSE, MODELMART, ARIS
D) FEO, TOOLS, REPORTS, ARROW, DIAGRAM
E) C++, PASCAL, JAVA, FORTRAN, VISIO
****
23. BPWin аспабында қолданылатын нотациялар
A) RATIONAL, ROSE, MODELMART, ARIS
B) WORD, EXELL, NOTEPAD
C) DFD, IDEF0, IDEF3
D)IDEF1X, IDEF2X
E) FEO, TOOLS, REPORTS, ARROW, DIAGRAM
****
24. Өңдеу аспабы бұл-
A) Мәтіндік редактор
B) Клавиатура және монитор
C) Жұмысты орындауға арналған құрал
D) Компьютер

E) Визуальды редактор
****
25. UML - тілі дамуының басталу жылы
A) 1996жыл
B) 200жыл
C) 1994жыл
D) 1995 жылы
E) 1992жылы
****
26. Бизнес –процестерді тізбектей орындау үшін қолданылатын нотация
A) IDEF0
B) IDEF1X
C)DFD
D) IDEF3
E) IDEF2X
****
27. ER win Data Modeler келесі деректерді жобалау нотацияларын қолданады
A) IDEF3
B) IDEF0
C) DFD
D) IDEF1X
E) IDEF2X
****
28. RUP аббревиатурасының мағынасын ашыңыз
A) Rational Unitized Phase
B) Rational Unit Process
C) Rational Unified Process
D) Rational Use Process
E) Rational Unknown Process
****
29. Мән бұл
A) Әртүрлі мәндерді байланыстыратын қатынастар
B) Бір қызғушылықтағы мәндер жиынтығының топтастырушылары
C) Модельдің негізігі элементері болып табылатын абстрация
D) Әртүрлі қызығушылықтағы қатынастыр
E) Модельдегі құрылыс блогының статикалық ұйымдастырушы элементі
****
30. Әдістеме бұл
A) Жобаны өңдеу кезінде қолданылатын жобаны тарату кездерін , қадамдарын және әдістерінің қолдану тәртіптерін анықтау
B) Аспаптық ортамен қолдануда болатын, графикалық нотациялар негізінде диаграммалар тұрғызатын әдістемелер аймағындағы технология
C) Процедура немесе электрондық ақпараттық жүйелер компонетерінің сипатамаларын генерациялау техникасы
D) Жүйе құрлымын, деректер элементін және өңдеу кездерін арнайы

диаграмадағы графикалық символдар арқылы бейнелеу, сондай-ақ табиғи әр формальді тілде жүйе жобасын сипаттау
E) Ақпараттық жүйелерді жобалау және талдаудың бір немее бірнеше әдістемесін қолдайтын арнайы бағдарламалар
****

31. Бағдарламаларды өңдеу кезіндегі қолданылатын модельдеу жолдары
A) динамикалық және статикалық
B) құрылымдық және динамикалық
C)тәуелді және тәуелсіз
D)логикалық және физикалық
E)алгоритмдік және объектілі - бағыталған
****

32. Алгоритімдік әдістің құрылымдық блоктары
A) блоктар және каркастар
B) объект және класс
C) процедура және функциялар
D) программалар және ішкі программалар
E) аспаптар және құрылғылар
****

33. Объектілі – бағытталған әдістің құрлымдық блоктары
A) объект және класс
B) процедура және функциялар
C) блоктар және каркастар
D) программалар және ішкі программалар
E) аспаптар және құрылғылар
****

34. BPWin бизнес-процестерді модельдеу аспабының IDEF0 нотациясында қолданылатын ішкі бағдарламалар
A) Input, Output,Control, Mechanism, Call
B) Input, Mechanism, Call, Type, Arrow
C) Input, Output.Control,Sourse, Type
D) Input, Mechanism, Call, Arrow, Name
E) Control, Mechanism, Call, Name, Sourse
****

35. RUP фазалары
A) Phase, Design, Construction, Transition
B) Inception, Elaboration, Construction, Transition
C) Centric, Iterative, Inception, Elaboration
D) Architecture, Construction, Transition
E)Incremtntal, Process, Design, View
****

36. Жоба кезеңдерін тізбектей фиксацияланған тәртіп бойынша орындайтын өмірлік цикл моделі қалай аталады
A) сырқамалы модель
B) каскадты модель

C) спиральді модель
D) шеңберлі модель
E) аралық бақылауы бар кезең аралық модель
****
37. RUP қолданылатын практикалар, ағындар және фазалар саны
A) 5 практика, 6 ағын және 3 фаза
B) 6 практика, 9 ағын және 4 фаза
C) 8 практика, 6 ағын және 3 фаза
D) 5 практика, 2 ағын және 9 фаза
E) 9практика, 4 ағын және 6 фаза
****
38. RUP қолданылатын ағындар тізімі
A) Бизнес – процестерді модельдеу, талаптарды өңдеу, талдау және жобалау, тарату, тестілеу, ашып қарау, конфигурациясын басқару
B) Талдау және жобалау,тарату , тестілеу, ашып қарау, конфигурациясын басқару, жобаны басқару, орындау ортасын талдау
C) Бизнес – процестерді модельдеу, талаптарды өңдеу, талдау және жобалау, тарату, тестілеу, ашып қарау, конфигурациясын басқару, орындау ортасын талдау
D) Бизнес – процестерді модельдеу, конфигурациясын басқару, жобаны басқару, орындау ортасын талдау
E) Талаптарды өңдеу, талдау және жобалау, тарату,тестілеу, ашып қарау, конфигурациясын басқару, жобаны басқару
****
39. Компонеттік программалаудың объектіліктен айырмашылығы
A) Тарату тілінің әмбебаптығы және сыртқы тілдерінің алуан түрлілігі
B) Жылдамдығы
C) Жадыда алатын орны
D) Жұмыс визуалдылығы
E) Сақтау жылдамдығы
****
40. UML тілінің жалпы механизмдері
A) Стереотиптер, белгіленген мәндер, шектеулер тәуелділіктер, мәндер
B) Спецификациялық, қосымшалар, қабылданған кеңейту механизімдері
C) Қабылданған бөлулер, кеңейту механизімдері, диаграммалар
D) Шектеулер, тәуелділіктер, мәндер
E) Спецификациялық, қосымшалар, шектеулер, мәндер
****

3-ші тақырып бойынша тест сұрақтары

1. Физикалық модель нені суреттейді?
A) деректер қорының архитектурасын
B) деректер қорының құрылысын

C) пәндік аймақта болатын деректерді
D) деректер қорындағы ақпараттың көрінісін анықтайды
E) барлық жауап дұрыс емес
****
2. ERWin-де физикалық модель нені көрсетеді?
A) байланыстарды
B) атрибуттарды
C) фактілерді
D) нақты ДҚБЖ
E) мәндерді
****
3. ERWin-ді модельдеу қай әдістемеде іске асырылады?
A) IDEF1X
B) IDEFO
C) IDEF3
D) ID
E) DFD
****
4. ERWin логикалық және физикалық модельдермен қандай жұмыс жасайды?
A) оларды бір диаграммаға біріктіреді
B) олардың арасында логикалық байланысты орнатады
C) олардың диаграммаларын тұрғызады
D) олардың атрибуттарын сәйкестендіреді
E) алғашқы кілтке сілтейді
****
5. ERWin-де объекті графикалық түрде көрсету үшін қандай кескін қолданылады?
A) үшбұрыш
B) төртбұрыш
C) шеңбер
D) доғал
E) трапеция
****
6. ERWin-де диаграммалар қандай блоктардан құрылады?
A) объект, атрибут
B) объект, байланыс
C) объект, атрибут, функция
D) объект, функция, байланыс
E) объект, атрибут, байланыс
****
7. ERWin-де байланыстың қандай түрлері бар?
A) бірінші және екінші
B) бірінші және екінші түрдегі
C) аталық және балалық
D) бір мәнді және екі мәнді

E) логикалық және физикалық
****
8. BP Win-де модельдеудің қандай түрлерін ерекше атайды?
A) IDEF0, DFD, IDEF3
B) DFD, IDE, IDEF0
C) IDE, DFD
D) IDEF0, IDEF3
E) IDE, DFD,
****
9. BPWin үлгілеуінде форманы алу үшін не қолданылады?
A) IDEFO
B) DFD
C) ID
D) IDEF3
E) FEO
****
10. Сыртқы ортамен жүйедегі қандай компоненттер көмегімен әсерлеседі?
A) кіріс
B) шығыс
C) басқару
D) механизмдер
E) аталғандардың бәрі
****
11. Жүйенің динамикалық бөліктерінің жұмысына қатысы жоқ диаграмманы көрсетіңіз?
A) прецеденттер диаграммасы
B) тізбектер диаграммасы
C) компоненттер диаграммасы
D) кооперациялар диаграммасы
E) күйлер диаграммасы
****
12. Жүйенің статикалық бөліктерімен жұмыс жасамайтын диаграмма?
A) прецеденттер диаграммасы
B) кластар диаграммасы
C) объектілер диаграммасы
D) компоненттер диаграммасы
E) тармақталу диаграммасы
****
13. UML тіліндегі жалпылау қатынасы деген не?
A) жалпы кластарды байланыстыратын арнайы қатынас
B) қолдану қатынасы
C) обьектілер арасындағы құрылымдық байланыс
D) элементтер арасындағы төменгі байланыс
E) обьектер арасындағы жоғары байланыс
****

14. UML тіліндегі ассоциация қатынасы деген не?
A) қолдану қатынасы
B) жалпы кластарды байланыстыратын арнайыланған қатынас
C) обьектілер арасындағы құрылымдық байланыс
D) элементтер арасындағы төменгі байланыс
E) обьектер арасындағы жоғары байланыс
****
15. UML – ды қай жерлерде қолданған аса тиімді?
A) өндіріс масштабының ақпараттық жүйесі; Web-жүйені үлестіру
B) телекоммуникациялар; транспорт; нарықтық сауда
C) қорғану өндірісі, авиация және космонавтика
D) медициналық электроника; ғылым
E) аталғандардың барлығы
****
16. UML тіліндегі байланыс тәуелділігі деген не?
A) қолдану байланысы
B) жалпы кластарды байланыстыратын арнайы қатынас
C) обьектілер арасындаєы құрылымдық байланыс
D) элементтер арасындаєы төменгі байланыс
E) обьектер арасындағы жоғары байланыс
****
17. UML қай тілдің үлгілеуіне жатпайды:
A) визуализация
B) спецификациялау
C) алгоритмизация
D) конструирование/құрастыру
E) документирование/құжаттау
****
18. UML тілі сөздіктерінің құрамында болмайды:
A) мәндер
B) қарым-қатынастар
C) диаграммалар
D) алгоритмдер
E) кластар
****
19. UML-да мән типі болмайтын тип:
A) құрылымдық
B) тәртіптік
C) тәуелді
D) топтайтын
E) аннотациялық
****
20. UML тілінің үлгілеріндегі зат есімдер берілген типтердің қайсысын көрсетеді?
A) құрылымдық

B) тәртіптік
C) тәуелді
D) топтайтын
E) аннотациялық
****
21. UML-дегі тәртіптік мәндерге жататын тип:
A) класс
B) кооперациялар
C) жағдайлар
D) интерфейстер
E) түйіндер
****
22. UML-дағы базалық элемент – класс мәндердің келесі тобына жатады
A) тәртіптік
B) топтастыратын
C) құрылымдық
D) аннотациялық
E) тәуелді
****
23. Күй элементі UML– да мәндердің келесі тобына жатады
A) тәртіптік
B) топтастыратын
C) құрылымдық
D) аннотациялық
E) тәуелді
****
24. Десте элементі UML– да мәндердің келесі тобына жатады
A) тәртіптік
B) топтастыратын
C) құрылымдық
D) аннотациялық
E) тәуелді
****
25. Ескерту элементі UML– да мәндердің келесі тобына жатады
A) тәртіптік
B) топтастыратын
C) құрылымдық
D) аннотациялық
E) тәуелді
****
26. UML– дағы қатынастар түріне жатпайтын қатынас
A) тәуелділік
B) ассоциация
C) жалпылау
D) іске асыру

E) өзара қарым–қатынас
****
27. UML–дағы жүйенің статикалық бөлігін бейнелемейтін диаграмма
A) кластар
B) объектілер
C) прецеденттер
D) компоненттер
E) жаю диаграммасы
****
28. UML–дағы жүйенің динамикалық бөлігін бейнелемейтін диаграмма
A) прецеденттер
B) объекттер
C) қызмет ету
D) тізбектер
E) күй
****
29. UML–да құрылымдық үлгілеуге жатпайтын диаграмма
A) визуалдау
B) спецификациялау
C) талдау
D) құрастыру
E) құжаттау
****
30. UML–дағы тәртіптік диаграммалар тобына жатпайтыны қайсысы?
A) прецеденттер
B) тізбектер
C) кооперация
D) жаю
E) қызмет ету
****
31. Прецеденттер диаграммасы нені бейнелейді?
A) хабардың уақыттық тәртіптелуі
B) объекттің құрылымдық ұйымдасуын
C) оқиға, өту және күй автоматы
D) қызметтерді басқару
E) актерлер мен олардың өзара қарым-қатынасын
****
32. Тізбектер диаграммасы нені бейнелейді?
A) хабардың уақыттық тәртіптелуі
B) объекттің құрылымдық ұйымдасуын
C) оқиға, өту және күй автоматы
D) қызметтерді басқару
E) актерлер мен олардың өзара қарым-қатынасын
****
33. Кооперация диаграммасы нені бейнелейді?

A) хабардың уақыттық тәртіптелуі
B) объекттің құрылымдық ұйымдасуын
C) оқиға, өту және күй автоматы
D) қызметтерді басқару
E) актерлер мен олардың өзара қарым-қатынасын
****
34. Күй диаграммасы нені бейнелейді?
A) хабардың уақыттық тәртіптелуі
B) объекттердің құрылымдық ұйымдасуын
C) оқиға, өту және күй автоматы
D) қызметтерді басқару
E) актерлер мен олардың өзара қарым-қатынасын
****
35. Қызмет ету диаграммасы нені бейнелейді?
A) хабардың уақыттық тәртіптелуі
B) объекттердің құрылымдық ұйымдасуын
C) оқиға, өту және күй автоматы
D) қызметтерді басқару
E) актерлер мен олардың өзара қарым-қатынасын
****
36. ERWin – нің логикалық үлгісі нені бейнелейді?
A) байланысты
B) атрибутты
C) фактілерді
D) нақты ДҚБЖ –ны
E) мәндерді
****
37. ERWin – нің физикалық үлгісі нені бейнелейді?
A) байланысты
B) атрибутты
C) фактілерді
D) нақты ДҚБЖ –ны
E) мәндерді
****
38. Rational Rose дегеніміз не?
A) құрастырылатын бизнес - модель
B) қарапайым сұлбалар салу аспабы
C) өндіріске енгізу аспабы
D) талдау аспабы
E) құрастырылатын жүйенің бейнесі
****
39. Rational Rose – дағы көрсетілім түрлері?
A) use case view, logical view, component view, deployment view
B) tools, report, component view
C) query,window, deployment view

D) component view, deployment view, file
E) көрсетілім қолданылмайды
****
40. Rational Rose –да экран элементтерінің қанша түрі бар?
A) 9
B) 3
C) 5
D) 12
E) 7
****

4-ші тақырып бойынша тест сұрақтары

1. Visual Studio аспаптарының басқа визуальды құрушы аспаптарынан айырмашылығы неде?
A) Жылдамдығы
B) Жұмыс визуальдылығы
C) Тарату тілінің әмбебаптығы және сыртқы тілдердің алуан түрлілігі
D) Жадыдағы алатын орны
E) Дұрыс жауап жоқ
****
2. CASE технологиясы – бұл
A) программалаудың ақпараттық технологиясы
B) программалау технологиясы
C) аппараттық және программалық аспаптардың жиыны
D) программалық құралдарды өңдеудің компьютерлік технологиясы
E) барлық жауап дұрыс
****
3. Объект дегеніміз не?
A) Программа құрушы палитрасындағы компонент
B) Формадағы компонент нұсқасы
C) Ақпаратты өңдеу процесінде қолданылатын, ақпараттық мәндер абстракциясы және типі
D) Класс көшірмесі – ПП таратылатын ақпараттық элемент
E) Код + деректер – жеке модуль түріндегі бейнелеу
****
4. Бір жұмыс орнында орындалатын элементер әрекеті?
A) операция
B) функция
C) бизнес процесс
D) бизнес модель
E) дұрыс жауап жоқ
****
5. Қазіргі заман бағдарламасында бағдарламаларды өңдеудің қандай

технологиялары қолданылады?
A) Құрылымдық және модульдік технологиялар
B) Модульдік және визуалды жобалау
C) Визуалды, уақиғалық, объектілі-бағытталған технологиялар
D) Визуалды модельдеу және текстуалды бағдарламалау
E) Java жјне Dot.Net
****

6. Визуалды бизнес – модельдеуі үшін қандай аспаптар қолданылады?
A) ERWin
B) BPWin
C) MS Visio
D) Far
E) Fregat
****

7. Бизнес – процесс модельдерінің қандай стандарттары ақпаратты өңдеуді сипаттайды?
A) Байланыстырушы және шектеулі
B) Кіріс және шығыс
C) Кіріс, шектеулер, орындаушылар, шығыс
D) Байланыстырушы, кіріс, шығыс
E) Шартты, кіріс, шығыс, байланыстырушы
****

8. IDEF обьектісінде қандай бағдаршалар қолданылады?
A) Байланыстырушы және шектеулі
B) Кіріс және шығыс
C) Кіріс, шектеулер, орындаушылар, шығыс
D) Байланыстырушы, кіріс, шығыс
E) Шартты, кіріс, шығыс, байланыстырушы
****

9. IDEF3 моделі не үшін қолданылады?
A) Деректер ағыны
B) Ақпараттар ағыны
C) Жұмыстар ағыны
D) Функциялар көпшесі
E) Деректер көпшесі
****

10. BPWin құрамындағы ABC не үшін қолданылады?
A) Деректер ағыны
B) Жұмыстың орындалу графигін құру үшін
C) Бизнес байланыстардың аудитін орындау үшін
D) Бизнес процестердің байланысына талдау жүргізу үшін
E) Базалық талдау – жұмысқа кететін шығындар жайында ақпарат жинау үшін қолданылатын есептемелер
****

11. Визуалды жобалаудың базалық элементі?

A) Интерпретатор немесе компилятордың бар болуы
B) ОС және JCL командаларының бар болуы
C) Программаны құруда командамен емес құрылымдық мүмкіндіктерді пайдалану
D) Макростардың бар болуы
E) Компановканың керектігі
****
12. Қандай аспаптық құрал IDEF1X методологиясын қолдайды
A) ешқайсысы қолдамайды
B) BPwin
C) ARIS
D) Rational Rose
E) Erwin
****
13. UML – да неше диаграмма түрін салуға болады?
A) 6
B) 7
C) 9
D) 12
E) 8
****
14. UML – де класстарға қолданатын стандартты стереотиптерді атаңыз
A) metaclass, powertype, stereotype, utility
B) stereotype, metaclass
C) metaclass
D) stereotype
E) стереотип қолданылмайды
****
15. UML – дің стандартты элементтеріне не жатады?
A) шектеулер, белгіленген мәндер, стереотиптер
B) стереотиптер
C) белгіленген мәндер, қатынастар
D) диаграммалар, қатынастар, мәндер
E) дұрыс жауабы жоқ
****
16. Итерация дегеніміз не?
A) бұл жобаланған функциялардың белгілі бір аяқталған этапы
B) фазалардың аяқталуы
C) ағындардың жұмысының аяқталуы
D) ағын және фаза жұмысының аяқталу
E) ешнәрсе жасамайды, тек жүйе жұмысы өзгереді
****
17. Диаграмма бұл -
A) элементтердің сызбалық көрсетілімі
B) қарапайым сызбалар

C) әрекеттердің көрсетілімі
D) обьектілердің көрсетілімі
E) көптеген элементтердің графикалық көрсетілімі. Әдетте төбелері (мәндері) және қабырғалары (қатынастар) бар граф түрінде болды
****

18. Интерфейс бұл -
A) программаның сыртқы түрі
B) класс немесе компонент көрсететін спецификациялар қызметінің құрамынан тұратын, көптеген операциялар
C) түйін көрсететін спецификациялар қызметінің құрамынан тұратын, көптеген операциялар
D) пакеттер көрсететін спецификациялар қызметінің құрамынан тұратын, көптеген операциялар
E) дұрыс жауабы жоқ
****

19. Компонент бұл-
A) класстар спецификациясын тарататын, жүйенің физикалық ауыстырылатын бөлігі
B) түйіндер спецификациясын тарататын, жүйенің физикалық ауыстырылатын бөлігі
C) интерфейстер спецификациясын тарататын, жүйенің физикалық ауыстырылатын бөлігі
D) прецеденттер спецификациясын тарататын, жүйенің физикалық ауыстырылатын бөлігі
E) дұрыс жауап жоқ
****

20. Прецедент бұл-
A) орындалатын операциялар тізбегі
B) орындалатын процедуралар тізбегі
C) белгілі бір актермен қадағаланатын, жүйемен орындалатын, көптеген тізбекті уақиғалар сипаты
D) орындалатын функциялар тізбегі
E) дұрыс жауабы жоқ
****

21. Пакет бұл-
A) элементтерді топтарға біріктіретін әмбебап механизм
B) класстарды топтарға біріктіретін әмбебап механизм
C) түйіндерді топтарға біріктіретін әмбебап механизм
D) прецеденттерді топтарға біріктіретін әмбебап механизм
E) қатынастарды топтарға біріктіретін әмбебап механизм
****

22. Композиция бұл -
A) агрегациялау формасы
B) ассоцияциялау формасы
C) таратылу формасы

D) тармақталу формасы
E) жалпылау формасы
****
23. Компилятор – бұл программалық аспап, не үшін арналған?
A) программадағы қатені табу үшін
B) программаны жолдап орындау үшін арналған
C) программаны бүтіндей орындау және ауыстыру үшін
D) файлдармен жұмыс жасау үшін
E) барлық жауап дұрыс
****
24. Файлдық менеджер – бұл программалық аспап, не үшін арналған?
A) программадағы қатені табу үшін
B) программаны жолдап орындау үшін арналған
C) программаны бүтіндей орындау және ауыстыру үшін
D) файлдармен жұмыс жасау үшін
E) барлық жауап дұрыс
****
25. Интерпретатор – бұл программалық аспап, не үшін арналған?
A) программадағы қатені табу үшін
B) программаны жолдап орындау үшін арналған
C) программаны бүтіндей орындау және ауыстыру үшін
D) файлдармен жұмыс жасау үшін
E) барлық жауап дұрыс
****
26. Модельдеуге не жатады
A) белгілеулер жүйесі
B) белгілеулер жүйесі, модельдеу тілінің синтаксисі, модельдеуде қолданылатын графикалық объектілер жиынтығы
C) модельдеу тілінің синтаксисі
D) моделдеуьде қолданылатын графикалық объектілер жиынтығы
E) белгілеу
****
27. Кейбір бизнес процесстердің құрылымдық элементі болып табылатын бизнес-процесс
A) операция
B) функция
C) бизнес-процесс
D) ішкі процесс
E) бизнес-модель
****
28. Жүйелік талдау құралы – бұл
A) Erwin
B) JAM
C) BPwin
D) Rational Rose

E) ARIS
****
29. Функционалды моделдеу методологиясы
A) SADT
B) IDEF0
C) IDEF1X
D) SADT, IDEF0
E) UML диаграммалар
****
30. Bpwin методологиясы қолданатын аспапты атаңыз
A) IDEF0, DFD
B) IDEF0, IDEF3, DFD, IDEF1X
C) IDEF1X
D) IDEF0, IDEF3, IDEF1X
E) IDEF0, IDEF3, DFD
****
31. Функционалды модельдермен олардың диаграммаларын көрсететін аспаптық құралдарды атаңыз
A) CALS
B) BPwin, Erwin
C) Rational Rose
D) ARIS
E) GPSS
****
32. UML диаграммасының аппараттарды көрсететін құралдарын атаңыз
A) CALS
B) BPwin, Erwin
C) Rational Rose
D) ARIS
E) GPSS
****
33. Қандай аспаптық құрылғы объектілі – бағдарланған модельдеу тілінің құралы болып табылады
A) Erwin
B) BPwin
C) ARIS
D) Rational Rose
E) GPSS
****
34. Кластарды анықтауға кірмейтін элемент?
A) абстракция
B) есімдер
C) атрибуттар
D) операциялар
E) міндеттемелер

****
35. IDEF3 моделіндегі Relational (қатынас байланысы) нені көрсетеді?
A) алдын алу байланысын
B) идеясыз, концепциясыз әдістерді көрсетеді
C) м/д жұмыспен немесе м/д жұмыспен обьект арасындағы өзара байланысты көрсетеді
D) кейбір обьектілер 2 немесе одан да көп жұмыста қолданылатынын көрсетеді
E) моделдеу процестегі логикалық сұлбаның тармақталуын көрсетеді
****
36. IDEF3 моделіндегі Precedence нені білдіреді?
A) алдын алу байланысын
B) идеясыз, концепциясыз әдістерді көрсетеді
C) м/д жұмыспен немесе м/д жұмыспен обьект арасындағы өзара байланысы
D) кейбір обьектілер 2 немесе одан да көп жұмыста қолданылатынын көрсетеді
E) моделдеу процестегі логикалық сұлбаның тармақталуын көрсетеді
****
37. IDEF3 диаграммасында Junctions (қиылыс) нені білдіреді?
A) алдын алу байланысы
B) идеясыз, концепциясыз әдістерді көрсетеді
C) м/д жұмыспен немесе м/д жұмыспен обьект арасындағы өзара байланысты көрсетеді
D) кейбір обьектілер 2 немесе одан да көп жұмыста қолданылатынын көрсетеді
E) моделдеу процестегі логикалық сұлбаның тармақталуын көрсетеді
****
38. IDEF3 диаграммасында Reference (сілтеу объектісі) нені білдіреді?
A) алдын алу байланысын
B) идеясыз, концепциясыз әдістерді көрсетеді
C) м/д жұмыспен немесе м/д жұмыспен обьект арасындағы өзара байланысты көрсетеді
D) кейбір обьектілер 2 немесе одан да көп жұмыста қолданылатынын көрсетеді
E) моделдеу процестегі логикалық сұлбаның тармақталуын көрсетеді
****
39. «Контекстті» диаграмманы құруда келесі қадамдар орындалады:
A) үлгілеу аумағын анықтау
B) үлгілеу мақсаты
C) көзқарас
D) үлгілеу аумағын анықтау
E) аталғанның бәрі дұрыс
****
40. ER диаграммада мәнді қалай бейнелейді?
A) төртбұрыш
B) сызық
C) стрелкасы бар сызық
D) үзік сызық
E) эллипс

****

5-ші тақырып бойынша тест сұрақтары

1. Өмірлік цикл (ӨЦ) моделі дегеніміз не?
A) Есептер, жұмыстар, процесстер құрылымы.
B) Инфрақұрылым құру, басқару, жетілдіру, оқып үйрену
C) Құжаттау, конфигурациясын басқару, сапасын қамтамасыздандыру, верификация, аттестация, бірге талдау, аудит, проблеманы шешу
D) Тапсырыс беру, жеткізу, өңдеу, эксплуатациялау, бірге жүру
****
2. Процесстің түрлері және олардың ӨЦ құрамындағы саны?
A) 6 негізгі, 7 қосымша, 3 ұйымдастырушы.
B) 8 негізгі, 5 қосымша, 5 ұйымдастырушы
C) 5 негізгі, 8 қосымша, 4 ұйымдастырушы
D) 4 негізгі, 5 қосымша, 8 ұйымдастырушы
****
3. Негізгі процесстерді атаңыз?
A) Инфрақұрылым құру, басқару, жетілдіру, оқып үйрену
B) Тапсырыс беру, жеткізу, өңдеу, эксплуатациялау, бірге жүру
C) Құжаттау, конфигурациясын басқару, сапасын қамтамасыздандыру, верификация, аттестация, бірге талдау, аудит, проблеманы шешу
D) Есептер, жұмыстар, процесстер құрылымы.
****
4. Қосымша процесстерді атаңыз?
A) Инфрақұрылым құру, басқару, жетілдіру, оқып үйрену
B) Тапсырыс беру, жеткізу, өңдеу, эксплуатациялау, бірге жүру
C) Есептер, жұмыстар, процесстер құрылымы.
D) Құжаттау, конфигурациясын басқару, сапасын қамтамасыздандыру, верификация, аттестация, бірге талдау, аудит, проблеманы шешу
****
5. Ұйымдастырушы процесстерді атаңыз?
A) Инфрақұрылым құру, басқару, жетілдіру, оқып үйрену
B) Құжаттау, конфигурациясын басқару, сапасын қамтамасыздандыру, верификация,аттестация, бірге талдау, аудит, проблеманы шешу
C) Тапсырыс беру, жеткізу, өңдеу, эксплуатациялау, бірге жүру
D) Есептер, жұмыстар, процесстер құрылымы
****
6. Өңдеу процессі қандай жұмыстардан тұрады?
A) Құжаттау, конфигурациясын басқару, сапасын қамтамасыздандыру, верификация,аттестация, бірге талдау, аудит, проблеманы шешу
B) Талаптарды талдау, жобалау, программалау, жинау, тестілеу, әрекетке дайындау, қабылдау
C) Инфрақұрылым құру, басқару, жетілдіру, оқып үйрену
D) Есептер, жұмыстар, процесстер құрылымы.

****
7. Талаптарды талдау жұмысын құжаттау және орнату кезінде қандай талаптар орындалу керек?
A) Сыртқы интерфейске
B) Ақпаратты жүргізу және ұйымдастыру
C) Ақпаратты қорғау
D)Функциональды және техникалық, сыртқы интерфейске, квалификациялық, функционалдау қауіпсіздігі, ақпаратты қорғау, қолмен жасалатын операция эргономикалығы, ақпаратты жүргізу және ұйымдастыру мен құжаттау.
****
8. RUP қандай жұмыстардан тұрады?
A) Зерттеу (Inception), жоспарды нақтылау (Elaboration), құру (Construction), ашып қарау (Transition).
B) Жобаны басқару, құру және өңдеу ортасын қадағалау
C) Талаптарды дайындау, жобалау, программалау, тестілеу, тарату
D) Талаптарды жинау және талаптарды басқару, талдау және моделдеу, кодтау, тестілеу
****
9. Егер тек ОӘК сүйенсек, онда неше балл алуға болады?
A) 60 баллдан төмен
B) 50. баллдан төмен
C) 70 баллдан төмен
D) 40 баллдан төмен
****
10. Компьютер үшін программа дегеніміз не?
A) Есептеулерді орындау жұмысының жоспары.
B) Процессор жұмысының жоспары
C) Компьютера жұмысының жоспары
D) Пайдаланушы жұмысының жоспары
****
11. Төменгі деңгейде программалау дегенді қалай түсінесіз?
A) Программаны C++ тілінде жазу.
B) Программаны микропроцессор немесе ассемблер тілінде жазу
C) Программаны Delphi тілінде жазу.
D) Программаны кез – келген тілде жазу.
****
12. 5GL деңгейінде программалау дегеніміз не?
A) Delphi программаларды қолдану
B) Текстуальды құрушы программасын қолдану
C) Visual Studio программаларды қолдану
D) Визуальды құрушы программасын қолдану
****
13. Деректерді өңдеуді тарату нені білдіреді?
A) Ағындар арасындағы басқару
B) Процесстер арасындағы басқару

C) Файлдарды тарату
D) Деректер ағымын бөлу және процессорлар арсындағы басқару (жұмысты орындаушылар).
****
14. Турбо қабықша қандай құралдарды береді?
A) Өңдеу панелі (форма), графикалық және текстік редактор.
B) Текстік редактор, компилятор және линкер, менеджер файлы, мэйкер, программа жөндегіш
C) Компоненттер кітапханасы, өңдеу панелі (форма), компоненттер қасиетінің инспекторы, графкалық және текстік редактор, компилятор және линкер, жөндегіш, диаграммалар редакторы, мастерлер (жол сілтеушілер), мэйкер, файлдар менеджері.
D) Мастерлер (жол сілтеушілер).
****
15. Аспаптарға қатынау жасау үшін менюдің қандай пункті қолданылады?
A) Tools.
B) File
C) Edit
D) Help
****
16. Жұмысты орындау ортасын қалыптастыру үшін меню пунктінің қайсысы қолданылады?
A) Window
B) Debug
C) Environment.
D) Format
****
17. Визуальды құрастырушы программасының құралдарын атаңыз?
A) Текстілік редактор, компилятор және линкер, менеджер файлы, мэйкер, программа жөндегіш.
B) Компоненттер қасиетінің инспекторы, графикалық және текстілік редактор.
C) Компилятор және линкер, жөндегіш, файлдар менеджері.
D) Компоненттер кітапханасы, өңдеу панелі (форма), компоненттер қасиетінің инспекторы, графкалық және текстік редактор, компилятор және линкер, жөндегіш, диаграммалар редакторы, мастерлер (жол сілтеушілер), мэйкер, файлдар менеджері.
****
18. Визуальды құрастырушы аспаптарын қалай бөлуге болады?
A) Сыртқы және ішкі (ішіне орналасқан).
B) Сыртқы
C) Ішкі
D) Бөлуге болмайды
****
19. Компиляция фазалары?
A) Сканирлеу, кодты генерациялау.

B) Сканирлеу, лексикалық талдау, синтаксистік талдау, кодты генерациялау.
C) Лексикалық талдау, синтаксистік талдау.
D) Сканирлеу, лексикалық талдау.
****

20. Визуальды программалу негізінде қандай элементтер технологиясы бар?
A) Drug and dock, VCL.
B) Drug and drop, VCL.
C) Drug and drop, drug and dock, VCL.
D) Drug and drop, drug and dock.
****

21. Visual Studio аспаптарының басқа визуальды құрушы аспаптарынан айырмашылығы неде?
A) Жылдамдығы
B) Жұмыс визуальдылығы
C) Тарату тілінің әмбебабтығы және сырқы тілдердің алуан түрлілігі.
D) Жадыдағы алатын орны
****

22. Компоненттік программалаудың обьектіліктен айырмашылығы?
A) Жұмыс визуальдылығы
B) Жылдамдығы
C) Жадыдағы алатын орны
D) Жеңілдігі
****

23. Қасиет пен атрибуттың айырмашылығы?
A) Тарату мүмкіндігі
B) Ауыстыру жүргізу мүмкінідігі
C) Өзгерту мүмкіндігі
D) Редакторлау мүмкіндігі
****

24. Программа инсталляциясы дегеніміз не?
A) Нақты бір есепке қою
B) Нақты бір процесске қою
C) Нақты бір компьютердағы оның тарату ортасына орналасуы
D) Нақты бір орындалу ортасына қою
****

25. Контексттік анықтама дегеніміз не?
A) Қарапайым көмек
B) Прграммадағы активтендірілген орындалға байланған.
C) Тек менюда берілетін көмек түрі
D) Маустың сол басқышын басқан сайын пайда болатын көмек түрі
****

26. Макропроцессор (препроцессор) жұмысы?
A) Макрошақыруларды (макрокомандалар) макроанықтауыштар текстіне ауыстырады (макрокеңейтулер).
B) Макроанықтауыштарды макрошақырулардың текстіне ауыстырады

C) Макрошақыруларды директивалармен ауыстырады
D) Макроанықтауыштарды командалармен ауыстырады
****

27. Линкер не жасайды?
A) Әртүрлі программаларды бір түйінге жинайды
B) Әртүрлі бөліктерден (объектілік модулдерден) программа құрайды
C) Бір программаны әртүрлі бөліктерге бөлуді орындайды
D) Программаны бөледі және жинайды
****

28. Программадағы инвариант дегеніміз не?
A) Есептеу жолдарына және әсер етуші факторларға байланысты мәндерді сақтау нәтижесі.
B) Қарапайым мәндерді сақтау нәтижесі.
C) Нәтиже
D) Есептеу жолдарына және әсер етуші факторларға байланысты емес мәндерді сақтау нәтижесі.
****

29. Жүйелік реестр сипаттамасы неде?
A) Компьютердегі жұмысты орындау ортасы.
B) Процессордағы жұмысты орындау ортасы
C) Жадыдағы жұмысты орындау ортасы
D) Файлдағы жұмысты орындау ортасы
****

30. Программа жөндегішінің жұмысы неде?
A) Программаның орындалуын автоматты түрде басқару
B) Деректерді талдау
C) Аралық деректердің талдау және программаның орындалуын қолмен басқару.
D) Программаның орындалуын қадамдап басқару

Қолданылған әдебиеттер тізімі

1 Б.У. Боэм «Инженерное проектирование программного обеспечения». М.: Радио и связь. 1985.
2 Д.Райли. «Использование языка Модула-2». М.: Мир. 1993.
3 Ю.В. Иванов «Программы и их жизненные циклы» (реферат по дисциплине «Метрология ПО»). 1998.
4 Леффингуал, Дин, Ундри, Дон. Принципы работы с требованиями к ПО. Унифицированный подход. М., 2002г.
5 У. Боггс, М. Боггс. UML, Rational Rose. М., ЛОРИ, 2000 г.
6 Сэм Канер и др. Тестирования программного обеспечения. Киев, 2000г.
7 С.В. Маклаков BPWin, и ERWin. CASE-разработки информационных систем. - М.:ДИАЛОГ-МИФИ, 2000 - 256 с.
8 Грейди Буч, Джеймс Рамбо, Айвар Джекобсон, Язык UML. Руководство пользователя: Пер. с англ - М.: ДМК Пресс, 2001
9 Джим Арлоу, Айла Нейштадт., UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование, 2- ое издание, Символ-Плюс, 2007г.
10 Дж. Рамбо, М. Блаха., UML 2.0. Объектно-ориентированное моделирование и разработка, Питер, 2007г.
11 Мацяшек Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0, 3-е издание, Вильямс. 2008г.
12 Киммел П. UML. Основы визуального анализа и проектирования, НТ Пресс, 2008г.
13 Ю.Мартин Фаулер. UML. Основы. Краткое руководство по стандартному языку объектного моделирования, Символ-Плюс 2011г.