Sh.A. NAZIROV,
G.S. IVANOVA, S.M. GAYNAZAROV

# DASTURLASH TEXNOLOGIYASI

Toshkent – 2014

O‘ZBEKISTON RESPUBLIKASI OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

Sh.A. NAZIROV,
G.S. IVANOVA, S.M. GAYNAZAROV

# DASTURLASH TEXNOLOGIYALARI

*Oliy va o‘rta maxsus ta’lim vazirligi tomonidan «Informatika va axborot texnologiyalari», «Kompyuter inginiring», «Dasturiy inginiring», «Axborot xavfsizligi», «Informatika va amaliy matematika», «Elektron tijorat», «Axborotlashtirish va kutubxonashunoslik», «Axborot servisi» ta’lim yo‘nalishlari talabalari uchun darslik sifatida tavsiya etilgan*

O‘zbekiston faylasuflari milliy jamiyati nashriyoti
Toshkent — 2014

**UO'K 004:681.5(075)**
**KBK 32.97-018**
**N-18** 4196

**Nazarov Sh.A.**

N-18 **Dasturlash texnologiyalari:** darslik / Sh.A. Nazirov, G.S. Ivanova, S.M. Gaynazarov; O'zbekison Respublikasi Oliy va o'rta maxsus ta'lim vazirligi. – Toshkent: O'zbekiston faylasuflari milliy jamiyati nashriyoti, 2014. – 280 b.

**UO'K 004:681.5(075)**
**KBK 32.97-018**

**Taqrizchilar:**

**M.M. Aripov** – Mirzo Ulug'bek nomidagi O'zMU professori , f.-m.f.d.,

**B.F. Abduraximov** – O'zMU professori, f.-m.f.d.,

**H.N. Zaynidinov** – TATU «Axborot texnologiyalari» kafedrasi professori, t.f.d.,

**R.N. Usmanov** – TATU «Kompyuter tizimlari» kafedrasi professori, t.f.d.

Dasturlash texnologiyalarining asosiy tushunchalari, uning hayotiy sikli, dasturiy vositalar va ulardagi xatolar manbalari, dasturiy vositalar sifatini ta'minlash yo'llari afzalliklari bayon etilgan.

Murakkab dasturiy ta'minot yaratishda qo'llaniladigan asosiy usullar va notatsiyalar batafsil yoritilgan. Dasturiy tizimlarning tarkibli, obyektli, komponentli yondashuvlar va algoritm yordamida loyihalashga e'tibor qaratilgan. Talab etiladigan texnologik xossalar atroflicha batafsil bayon etilgan. Dasturiy ta'minotdan foydalanish interfeysini loyihalash tamoyillari sinflarga ajratilgan va tahlil etilgan. Algoritmik dasturlash asoslari, dasturiy ta'minot verifikatsiyasi, testlash va sozlash usullari batafsil bayon etilgan. Dasturiy vositalarni ishlab chiqishning boshqarish usullari yoritilgan. Darslikda ko'plab sondagi illustrativ misollar va tushuntirish uchun rasmlar keltirilgan.

ISBN 978-9943-391-81-9

© O'zbekiston faylasuflari milliy jamiyati nashriyoti, 2014.

# KIRISH

Dasturiy tizimlarni yaratish mashaqqatli mehnat talab etadigan masala hisoblanadi. Buning uchun hozirgi kunda yaratilayotgan oddiy dasturiy ta'minotlar kamida 100 mingdan ortiq operatorlardan tashkil topishini qayd etishning o'zi yetarlidir. Demak, shunday ekan dasturiy ta'minotni yaratish bo'yicha bo'lajak mutaxassis dasturiy tizimlarni tahlil etish, loyihalash, yaratish (realizatsiya qilish), testlash usullari hamda bu sohada mavjud zamonaviy yo'llar va texnologiyalar bo'yicha yetarli darajada ko'nikmalarga ega bo'lishlari lozim.

Darslik o'n bitta bobdan iborat bo'lib, mavzular dasturiy ta'minotni yaratish bosqichlariga qat'iy rioya qilingan holda joylashtirilgan. Faqat dasturlash texnologiyalarining umumiy masalalari bundan mustasno.

Darslikning birinchi bobida dasturlash texnologiyalari va uning asosiy tushunchalari yoritilgan. Ishonchli dasturiy vosita dasturlash texnologiyalarining mahsuloti ekanligi, dasturlash texnologiyalari rivojining asosiy bosqichlari keltirilgan murakkab dasturiy ta'minotni yaratishda blokli-iyerarxik yondashuv ko'rsatilgan va mazkur yondashuv dasturiy mahsulotlarni yaratishda o'ziga xosligi e'tiborga olingan.

Ikkinchi bob dasturiy mahsulotlarning texnologikligini ta'minlash usullariga bag'ishlangan. Shuning uchun ham mazkur bobda dasturiy ta'minot tushunchasi, modellar va ularning xususiyatlari, dasturiy ta'minotning quyilashuvi va ko'tariluvchi ishlab chiqish usullari, tarkibli va «notarkibli» dasturlash yondashuvlari va ularning algoritmlari bayoni vositalari, dasturni rasmiylashtirish uslubi hamda strukturali dasturlashning nazorati to'g'risida materiallar joy olgan.

Dasturiy ta'minot va uni loyihalash uchun kiritilayotgan ma'lumotlar, talablarni aniqlash yo'llari uchinchi bobda o'z aksini topgan. Bu yerda dasturiy mahsulotlarning funksional alomati bo'yicha tasnifi, dasturiy ma'lumotlarga nisbatan asosiy tasarru-



AXBOROT RESURS MARKAZI
47265

fiy talablar, predmet sohasida loyiha oldi tadqiqotlari, texnik talablarni ishlab chiqish, loyihalashning boshlang'ich bosqichlariga oid prinsipial yechimlar to'g'risida ma'lumotlar yoritilgan. Tarkibli yondashuvda dasturiy ta'minot talablarini tahlil etish va spetsifikatsiyasini aniqlashning o'ziga xos xususiyatlari hamda ushbu yondashuvda dasturiy ta'minotni loyihalash uslubiyati xususiyatlari mos ravishda to'rtinchi va beshinchi boblarda yoritilgan.

Obyektli yondashuv uchun xuddi shunday ma'lumotlar mos ravishda oltinchi va yettinchi boblarda keltirilgan. Shu bois bu yerda obyektli yondashuv asosidagi ishlanmalarning kuchli va amaliy vositasi yordamida modellarni tahlil etish hamda loyihalashning asosiy tili sifatida UML (Unified Modelling Languange) – unifikatsiyalangan modellashtirish tili qo'llanilgan.

Sakkizinchi bobda foydalanuvchi interfeysini loyihalash muammolari va interfeysni yaratish modellari keltirilgan.

Dasturiy vositalar testlash va sozlash mos ravishda to'qqizinchi va o'ninchi boblarda yoritilgan bo'lib, testlash va sozlashning qator usullari batafsil ravishda keltirilgan.

Dasturiy vositalar hujjatlarini yaratish o'n birinchi bobda keng qamrovli tarzda yoritilgan. Bu yerda dasturiy hujjatlar turlari, tushuntirish yozuvnomasi, foydalanuvchi va tizimli dasturchilar uchun qo'llanmalar hamda dasturiy hujjatlarni rasmiylashtirishning asosiy qoidalari keltirilgan.

Darslikning har bir bobi nazariy savollar va mashqlar bilan to'ldirilgan.

Mazkur darslik Oliy ta'lim muassasalari uchun birinchi marta yaratilayotgani uchun unda kamchiliklar bo'lishi mumkin. Shu sabab darslik sifatini yaxshilash uchun takliflarni bajoni-u dil qabul qilamiz.

# 1. DASTURLASH TEXNOLOGIYASI. ASOSIY TUSHUNCHALAR VA YONDASHUVLAR

*Dasturlash* – predmet va texnikaning nisbatan yosh va tez rivojlanayotgan sohasi. Mavjud dasturiy va texnik vositalarni real ishlab chiqish va tarmoqlashtirishni olib borish tajribasi doimo qayta izlanishni talab etadi. Natijada yangi usullar, uslublar va texnologiyalar paydo bo'ladi, ular o'z navbatida, dasturiy ta'minlashni ishlab chiqishning yanada zamonaviy vositalari uchun asos bo'lib xizmat qiladi. Yangi texnologiyalarni yaratish jarayonlarini tekshirish va ularning asosiy yo'nalishlarini aniqlash, bu texnologiyalarni dasturlashning rivojlanish darajasi hamda dasturchilar ixtiyoridagi dasturiy va apparat vositalarining xususiyatlari bilan solishtirish maqsadga muvofiqdir.

## 1.1. Ishonchli dasturiy vosita dasturlash texnologiyasining mahsuli sifatida. Dasturlashtirishning tarixiy va ijtimoiy konteksti

Ma'lumotlarga ishlov berish jarayonining axborot muhiti tushunchasi. Dastur jarayonning formallashgan tavsifi sifatida. Dasturiy vosita haqida tushuncha. Dasturiy vositadagi xato tushunchasi. To'g'ri dastur tushunchasining konstruktiv emasligi. Dasturiy vositaning ishonchliligi. Dasturlash texnologiyasi ishonchli dasturiy vositalarni ishlab chiqish texnologiyasi sifatida. Dasturlash texnologiyasi va jamiyatning axborotlashtirilishi.

### 1.1.1. Dastur ma'lumotlarga ishlov berish jarayonining formallashgan tavsifi sifatida. Dasturiy vosita

Dasturlashtirishning bosh maqsadi ma'lumotlarga ishlov berish jarayonlari (bundan buyon *jarayonlar* deb ataladi)ni tavsiflashdan iborat. Axborotga ishlov berish bo'yicha xalqaro federatsiya (IFIP)da qabul qilinganidek, ma'lumotlar *(data)* bu qandaydir jarayonda uzatish va qayta ishlash uchun yaroqli bo'lgan formallashgan ko'rinishdagi faktlar va g'oyalarning taqdimotidir, axborot *(information)* esa ma'lumotlarni taqdim

etishda ularga berilgan ma'noni bildiradi. Ma'lumotlarga ishlov berish *(data processing)* ushbu ma'lumotlar ustida muntazam ketma-ketlikdagi amallarni bajarish demak. Ma'lumotlar *ma'lumot* tashuvchilarda taqdim etiladi va saqlanadi. Ma'lumotlarga biron-bir ishlov berish paytida qo'llanadigan ma'lumot tashuvchilar majmuini *axborot muhiti (data medium)* deb ataymiz. Axborot muhitida ma'lum bir vaqt mobaynida mavjud bo'lgan ma'lumotlar to'plamini ushbu *axborot muhitining holati* deb ataymiz. Jarayonga esa shunday ta'rif berish mumkin: jarayon bu biron-bir axborot muhitida o'zaro almashinib keluvchi holatlar ketma-ketligi.

Jarayonga tavsif berish bu berilgan axborot muhitidagi holatlar ketma-ketligini aniqlash demak. Agar biz talab qilinayotgan jarayon unga berilgan tavsifga ko'ra biron-bir kompyuterda *avtomatik tarzda* hosil bo'lishini xohlasak, bu holda ushbu tavsif formallashgan bo'lishi lozim. Bunday tavsif *dastur* deb ataladi. Bu o'rinda dastur insonga ham tushunarli bo'lmog'i lozim. Chunki dasturlarni ishlab chiqishda ham, ulardan foydalanishda ham ushbu dastur aynan qanday jarayonni yuzaga keltirayotganini aniqlab olishga to'g'ri keladi. Shu bois dastur inson uchun qulay bo'lgan formallashgan *dasturlash tili*da tuziladi hamda *translyator* deb ataluvchi boshqa bir dastur yordamida avtomatik tarzda tegishli kompyuter tiliga o'tkaziladi. Inson *(dasturchi)* o'zi uchun qulay dasturlash tilida dastur tuzishdan avval, katta tayyorgarlik ishini olib borishi lozim: masalaning qo'yilishini aniqlab olish, uni yechish usullarini tanlash, talab qilinayotgan dasturni qo'llashdagi o'ziga xos xususiyatlarni aniqlash, ishlab chiqilayotgan dasturni tashkillashtirishning umumiy jihatlariga aniqlik kiritish va h.k. Bu axborotdan foydalanish inson uchun dasturni anglab yetish masalasini ancha osonlashtirishi mumkin. Shuning uchun uni alohida hujjatlar ko'rinishida qayd etib borish g'oyat foydalidir (bu o'rinda bunday hujjatlar odatda formallashmagan bo'lib, faqatgina tushunchalarga oydinlik kiritishga mo'ljallangan bo'ladi).

Odatda dasturlar ularni ishlab chiqishda ishtirok etmaydigan insonlar (ular *foydalanuvchilar* deb ataladi) foydalana oladigan qilib ishlab chiqiladi. Dasturni o'zlashtirish uchun foydalanuvchiga uning matnidan tashqari yana qandaydir qo'shimcha hujjatlar ham kerak bo'ladi. Ma'lumot tashuvchilardagi dasturiy hujjatlar bilan ta'minlangan dastur yoki mantiqan bog'langan dasturlar majmui *dasturiy vosita* (DV) deb ataladi. Dastur ma'lumotlarga kompyuterda ba'zi bir avtomatik ishlov berishni amalga oshirish imkonini beradi. Dasturiy hujjatlar dasturiy vositaning u yoki bu dasturi qanday vazifalarni bajarishini, dastlabki ma'lumotlarni qanday tayyorlash va talabdagi dasturni qanday ishga tushirish kerakligini, shuningdek olinayotgan natijalar nimani bildirishini (yoki ushbu dasturning bajarilishidan olinayotgan samara nimada ekanligini) tushunib olishga yordam beradi. Bundan tashqari, dasturiy hujjatlar dasturning o'zini tushunib yetishga yordam beradiki, bu uni modifikatsiya qilishda g'oyat muhimdir.

## 1.2. Dasturlash texnologiyasi va uning rivojlanishining asosiy bosqichlari

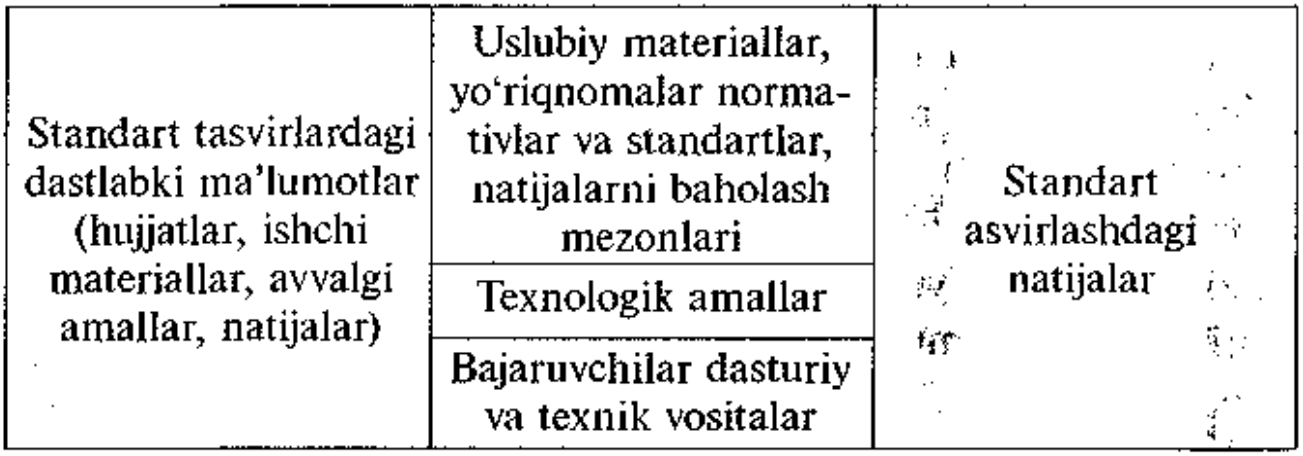
Dasturiy ta'minotining ishlab chiqish jarayonida foydalaniladigan usullar va vositalar yig'indisi dasturlash texnologiyalari deb ataladi. Har qanday boshqa texnologiya kabi, dasturlash texnologiyasi o'z ichiga quyidagi texnologik yo'riqnomalar to'plamini olgan:

- texnologik operatsiyalarni bajarish ketma-ketligini ko'rsatish;
- u yoki boshqa operatsiya bajariladigan shartlarni sanab o'tish;
- har bir operatsiya uchun dastlabki ma'lumotlar, natijalar, shuningdek, yo'riqnomalar, normativlar, standartlar, baholash mezonlari va usullari belgilangan operatsiyalarining ta'riflari (1.1-rasm)

Operatsiyalar (amallar) to'plami va ular ketma-ketligidan tashqari, texnologiya, shuningdek ishlab chiqarishning muay-

yan bosqichida foydalaniladigan loyihalashtirilayotgan tizimni, aniqrog'i, modelni ta'riflash usulini belgilaydi.

Ishlab chiqishning muayyan bosqichlarida yoki bu bosqichning alohida masalalarini hal etish uchun foydalaniladigan texnologiyalar yoki ishlab chiqish jarayonini qamrab oluvchi texnologiyalar farqlanadi. Birinchisi asosida, qoidaga ko'ra, aniq (vazipredmeti ) hal etish imkonini beradigan cheklangan darajada qo'llaniladigan usul mavjud. Ikkinchisi asosida tayanch (bazaviy) usul yoki ishlab chiqishning turli bosqichlarida foydalaniladigan usullar birligini yoki uslubiyatini belgilovchi yondashish mavjud.




**1.1-rasm.** Texnologik amallar yuritmasining tarkibi.

Dasturlashning mavjud texnologiyalarini farqlash va ularning asosiy yo'nalishlarini aniqlash uchun dasturlashning predmet sifatida rivojlanishining asosiy bosqichlarini ajratib, bu texnologiyalarni tarixiy kontekstda ko'rib chiqish maqsadga muvofiqdir.

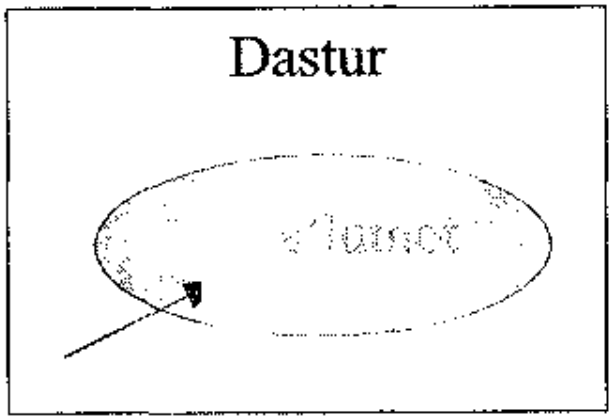
Birinchi bosqich – «stixiyali» dasturlash. Bu bosqich dastlabki hisoblash mashinalarining paydo bo'lishi paytidan boshlab, XX asrning 60-yillarigacha bo'lgan davrni qamrab oladi. Bu davrda tuzilgan texnologiyalar yo'q bo'lib, dasturlash asosan san'at darajasida edi. Dastlabki dasturlar sodda tuzilishga ega bo'lgan. Ular mashina tilidagi dasturdan va u ishlov beradigan ma'lumotlardan iborat (1.2-rasm).

Mashina kodlaridagi dasturlarning murakkabligi dasturchining bir vaqtning o'zida bajarilayotgan operatsiyalarning ketma-

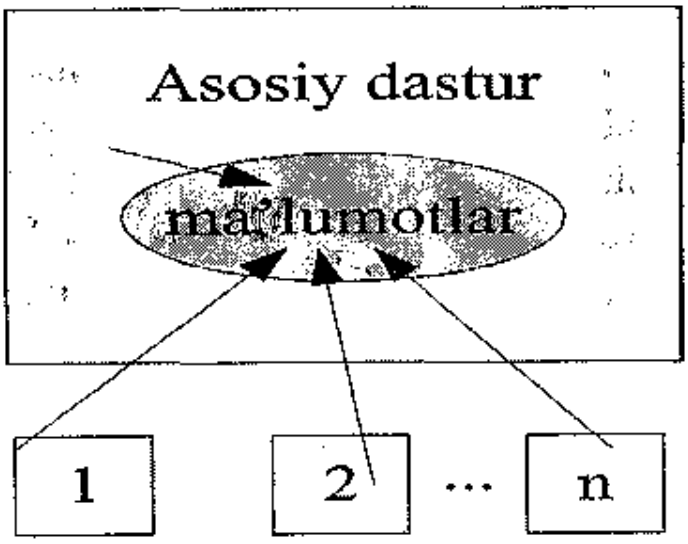
ketligini va dasturlashdagi ma'lumotlarning joylashishini fikran kuzatish qobiliyati bilan chegaralangan.

Assemblerning paydo bo'lishi ikkilik yoki o'n oltilik kodlar o'rniga ma'lumotlarning ramzi y nomlari va operatsiyalar kodlarining mnemonikasidan foydalanish imkonini berdi. Natijada dasturlar yanada «o'qitiladigan» bo'ldi.

FORTRAN va ALGOL kabi yuqori darajali dasturlash tillarining yaratilish operatsiyalarini detallashtirish darajasini pasaytirib, hisoblashlarni dasturlashni ancha soddalashtiradi. Bu, o'z navbatida, dasturlarning murakkabligini oshirishga yordam beradi.




**1.2-rasm.** Dastlabki dasturlar tarkibi.



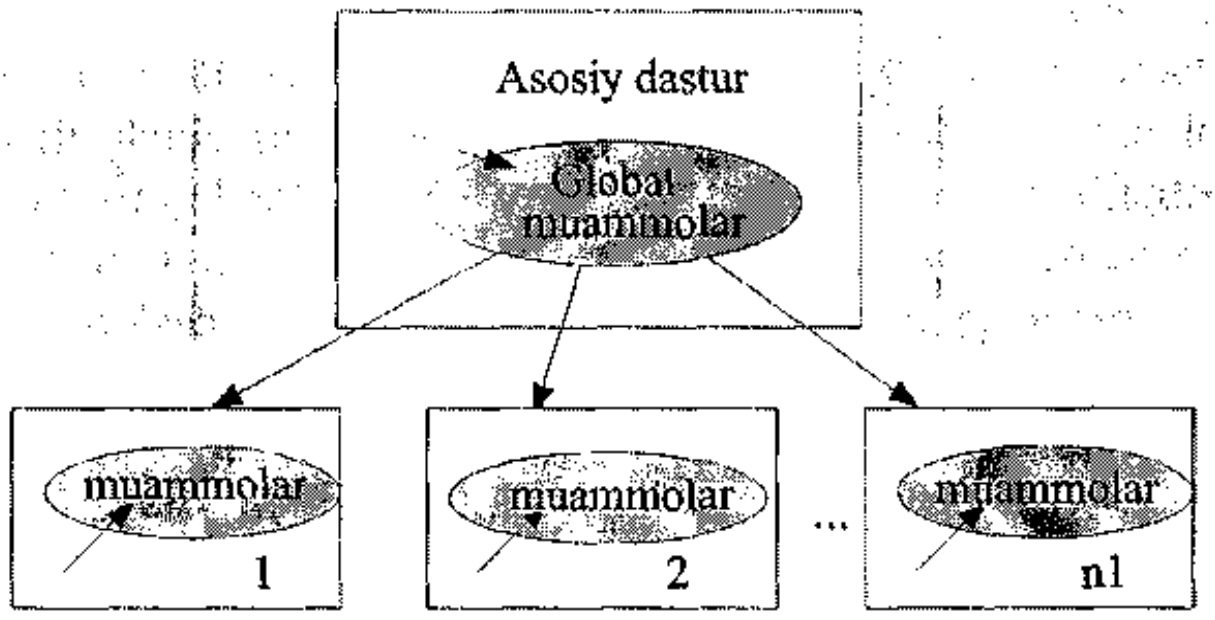

Qism dasturlar

**1.3-rasm.** Global sohali ma'lumotlar uchun dasturlar arxitekturasi.

Tillarda kichik dasturlar bilan ishlash imkonini beruvchi vositalarining paydo bo'lishi inqilob darajasida edi (kichik dasturlarni yozish g'oyasi ancha ilgari paydo bo'lgan). Kichik dasturlarni boshqa dasturlarda saqlash va ulardan foydalanish mumkin edi. Natijada kerakliligi darajasiga ko'ra ishlab chiqilayotgan dasturdan chiqarib olinadigan hisobiy va xizmat kichik dasturlarining yirik kutubxonalari tashkil etildi.

O'sha davrning namunaviy dasturi barcha ma'lumotlar yoki ularning qismiga ishlov beruvchi asosiy dastur, *global ma'lumotlar* sohasi va kichik dasturlar to'plami (asosan kutubxonaga tegishli) dan iborat edi (1.3-rasm).

Bunday arxitekturaning zaif tomoni shundan iborat ediki, kichik dasturlar miqdori ko'payishi bilan global ma'lumotlar qismining qandaydir kichik dastur tomonidan buzib ko'rsatish ehtimoli ortar edi. Masalan, kesmani o'rtasidan bo'lish usuli bo'yicha berilgan intervalda tenglama ildizlarini qidirishning kichik dasturi interval kattaligini o'zgartiradi. Agar kichik dasturdan chiqishda boshlang'ich intervalning tiklanishi ko'zda tutilmasa, u holda global sohada intervalning noto'g'ri qiymati hosil bo'ladi. Bunday xatolar miqdorini kamaytirish uchun kichik dasturlarda *lokal ma'lumotlarni* joylashtirish taklif etiladi (1.4-rasm).




**1.4-rasm.** Lokal ma'lumotli tag dastur (dastur qismi)lar ishlatuvchi dasturlar arxitekturasi.

Lokal ma'lumotlarga ega kichik dasturlardan foydalanishda ishlab chiqilayotgan dasturiy ta'minotning murakkabligi, avvalgidek, dasturchining ma'lumotlarga ishlov berish jarayonini endi yangi darajada kuzatish imkoniyati bilan cheklanar edi. Ammo kichik dasturlar ta'minot vositalarining paydo bo'lishi dasturiy ta'minotni bir necha dasturchilar tomonidan parallel ishlab chiqishni amalga oshirish imkonini berdi.

XX asrning 60-yillari boshida «dasturlash inqirozi» yuz berdi. U operatsion tizimlar kabi murakkab dasturiy ta'minotni ishlab chiqishga uringan firmalar loyihalarni tugallashning barcha muddatlaridan kechikkan edi. Loyiha joriy etilishidan oldinroq eskirib qolar, uning qiymati ortib borar va, natijada, ko'pgina loyihalar shu tariqa tugallanmay qolib ketar edi.

Obyektiv darajada buning hammasiga dasturlash texnologiyasining nomukammalligi sabab bo'ldi. Eng avvalo, «quyidan yuqoriga» ishlab chiqilishdan stixiyali (betartib) foydalanilgan. Ya'ni oldin nisbatan sodda kichik dasturlar loyihalashtirilib, amalga oshirilgan yondashish, ulardan keyin murakkab dastur tuzishga urinilgan. Kichik dasturlar va ularni loyihalashtirish usullarini ta'riflash aniq modellarining yo'qligidan har bir kichik dasturning yaratilishi murakkab vazifaga aylanar, kichik dasturlar interfeyslari murakkab bo'lib, dasturiy mahsulotni yig'ishda ko'p miqdorda xatolar aniqlanar edi. Bunday xatolarning tuzatilishi, odatda, avval ishlab chiqilgan kichik dasturlarning ancha o'zgartirilishini talab qilar, bu vaziyatni yanada murakkablashtirardi. Chunki bunda dasturga yangi xatolar kiritilar, ularni ham tuzatish kerak bo'lar edi. Oqibatda dasturni testdan o'tkazish va sozlash jarayoni ishlab chiqish vaqtining 80% ni egallar yoki umuman tugamas edi. Kun tartibida loyihalashtirish xatolari ehtimolini kamaytiruvchi murakkab dasturiy mahsulotlarni yaratish texnologiyasini ishlab chiqish masalasi turardi.

Ko'pgina xatolarning kelib chiqish sabablari tahlili «tarkibiy» deb nomlangan dasturlashga yangi yondashishni shakllantirish imkonini berdi.

**Ikkinchi bosqich dasturlashga tarkibiy yondashish** (XX asrning 60–70-yillari). *Dasturlashga tarkibiy yondashish* dasturiy ta'minlashni ishlab chiqishning barcha bosqichlarining bajarilishini qamrab oluvchi tavsiya etilayotgan texnologik usullar majmuasidan iborat. Tarkibiy yondashish asosida uncha katta bo'lmagan (40–50 operatorgacha) alohida kichik dasturlar ko'rinishida keyingi amalga oshirish maqsadida murakkab tizimlar *dekompozitsiyasi* (qismlarga bo'laklash) yotadi. Dekompozitsiya boshqa tamoyillari (obyektli, mantiqiy va h.) ning paydo bo'lishidan keyin ushbu usul protsedurali dekompozitsiya deb ataladi.

Ilgariroq foydalanilgan dekompozitsiyaga protsedurali yondashishdan farqli ravishda tarkibiy yondashish eng sodda tuzilishdagi kichik masalalar iyerarxiyasi ko'rinishida masalaning taqdim etilishini talab qilar edi. Loyihalashtirish, shu tariqa «quyidan yuqoriga» amalga oshirilib, kichik dasturlar interfeyslarining ishlab chiqilishini ta'minlagan holda umumiy g'oyaning amalga oshirilishini ko'zda tutgan edi. Bir vaqtning o'zida algoritmlar konstruksiyasiga cheklashlar kiritilar, ular ta'rifining formal modellari, shuningdek, algoritmlarni loyihalashtirishning maxsus usuli – qadam-baqadam detallashtirish usuli tavsiya etilgan edi.

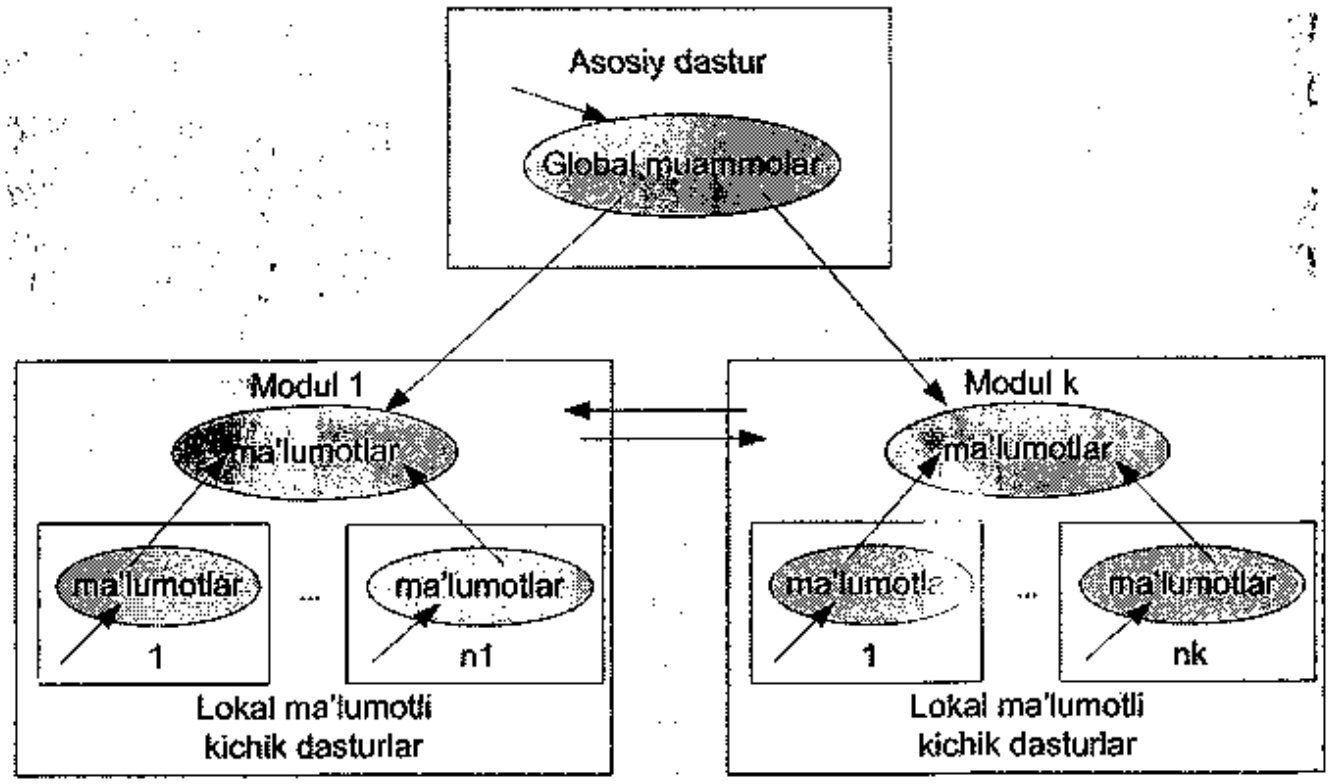
Tarkibiy dasturlash tamoyillarini ta'minlash dasturlashning *protsedurali* tillari asosiga kiritilgan. Qoidaga ko'ra, ular boshqaruvni uzatishning asosiy «tarkibiy» operatorlarini, kichik dasturlar kiritilishi, lokallashtirish va ma'lumotlar sohasining «ko'rinishliligi»ning chegaralanishini ta'minlar edi. Bu guruhning eng ma'lum tillaridan PL/1, ALGOL-68, Pascal, C ni misol qilib ko'rsatish mumkin.

Tarkibiy dasturlash bilan bir vaqtda boshqa konsepsiyalarga asoslangan ko'p miqdordagi tillar paydo bo'ldi, ammo ularning ko'pchiligi raqobatga bardosh bera olmadi. Qaysidir tillar unutildi, boshqa tillarning g'oyalaridan, rivojlanayotgan tillarning keyingi versiyalaridan keyinchalik foydalanildi.

Ishlab chiqilayotgan dasturiy ta'minot murakkabligi va o'lchamlarining keyingi o'sishi ma'lumotlar tuzilishining rivojlanishini talab qildi. Buning oqibatida tillarda ma'lumotlarning foydalanish turlarini belgilash imkoniyati paydo bo'ldi. Ayni paytda global ma'lumotlar bilan ishlashda kelib chiqadigan xatolar miqdorini kamaytirish uchun dasturning global ma'lumotlaridan foydalanishni chegaralashga urinish kuchaydi. Natijada modulli dasturlash texnologiyasi paydo bo'lib, rivojlana boshladi.

**Modulli dasturlash** aynan bir xil global ma'lumotlarni alohida modullarga (kichik dasturlar kutubxonasi) ajratilishini ko'zda tutadi. Masalan, grafik resurslar moduli, printerga chiqarish kichik dasturlari moduli (1.5-rasm).

Ushbu texnologiyadan foydalanishda modullar o'rtasidagi aloqa maxsus interfeys orqali amalga oshiriladi, ayni vaqtda modulning amalga oshirilishidan (kichik dasturlar tanasi va ba'zi «ichki» o'zgaruvchan) foydalanish taqiqlangan. Bu texnologiyani Pascal va C (C++) tillarining zamonaviy versiyalari, Ada va Modula tillari ta'minlaydi.




**1.5-rasm.** Modullardan tashkil topgan dasturlar arxitekturasi.

Modulli dasturlashdan foydalanish dasturiy ta'minotning bir qancha dasturchilar tomonidan ishlab chiqilishini ancha osonlashtirdi. Endilikda ulardan har biri modullar o'zaro ta'sirini maxsus aytib o'tilgan modullararo interfeyslar orqali ta'minlab, o'z modullarini mustaqil ishlab chiqishi mumkin edi. Bundan tashqari, modullardan keyinchalik o'zgarishlarsiz boshqa ishlab chiqishlarda foydalanish mumkin bo'lgan. Bu dasturchilarning mehnat unumdorligini oshiradi.

Amaliyot shuni ko'rsatdiki, modulli dasturlash bilan birgalikda tarkibiy yondashish o'lchami 100 000 operatordan oshmagan yetarli darajada ishonchli dasturlarni yaratish imkonini beradi. Kichik dasturni chaqirishda interfeysdagi xato faqat dastur bajarilgandagina aniqlanishi (modullarning ajratilgan *kompilyatsiyasi* tufayli bu xatolarni ilgariroq aniqlashning iloji yo'q) modulli dasturlashning tor joyi hisoblanadi. Odatda dastur o'lchami kattalashganda modullararo interfeyslarning murakkabligi ortadi va qaysidir daqiqadan boshlab dastur alohida qismlarining o'zaro ta'sir ko'rsatishini oldindan bilishning hech ham iloji yo'q. Katta ko'lamdagi dasturiy ta'minotni ishlab chiqish uchun obyektli yondashishdan foydalanish taklif etildi.

**Uchinchi bosqich – dasturlashga obyektli yondashuv** (XX asrning 80-yillari o'rtasidan 90-yillar oxirigacha). *Obyektga mo'ljallangan dasturlash* har biri muayyan turdagi (klassdagi) nusxa bo'lgan obyektlar majmuasi ko'rinishida dasturning taqdim etilishiga asoslangan murakkab dasturiy ta'minlashni yaratish texnologiyasi sifatida aniqlanadi, bunda klasslar xususiyatlarni meros qilib olish bilan birga iyerarxiyani hosil qiladi.

Bunday tizimdagi dasturiy obyektlarning o'zaro harakati xabarlarni uzatish yo'li orqali amalga oshiriladi (1.6-rasm).

Dasturning obyektli tuzilishidan ilk bor XX asrning 60-yillaridayoq paydo bo'lgan murakkab tizimlarni imitatsiyali modellashtirish tili – Simula da foydalanilgan. Modellashtirish tillari uchun tabiiy hisoblangan dasturni taqdim etish usuli modellashtirishning boshqa ixtisoslashtirilgan tili – Smalltalk tilida rivojlanishni

davom ettirgan (XX asrning 70-yillari), keyin esa, Pascal, C++, Modula, Java kabi dasturlashning universal tillariga ko'chirilgan.

Modulli dasturlashga nisbatan obyektga mo'ljallangan dasturlashning asosiy yutug'i dasturiy ta'minotni ishlab chiqishni ancha yengillashtiruvchi dasturiy ta'minotning «yanada tabiiyroq» bo'laklanishi hisoblanadi. Bu ma'lumotlarni yanada to'liqroq lokallashtirish va ularning ishlov berish kichik dasturlari bilan integratsiyalashishiga olib keladi. O'z navbatida, mazkur holat dasturning alohida qismlari (obyektlar)ning mustaqil ishlab chiqilishini olib borish imkonini beradi. Bundan tashqari, obyektli yondashish merosiylik, polimorfizm, kompozitsiyalash, to'ldirish mexanizmlariga asoslangan dasturlarni tashkil etishning yangi usullarini taklif etadi. Bu mexanizmlar nisbatan oddiy obyektlardan murakkab obyektlarni qurish imkoniyatini beradi. Natijada kodlardan qayta foydalanish ko'rsatkichi ancha oshadi va turli maqsadlarda qo'llash uchun sinflar kutubxonasini yaratish imkoniyati paydo bo'ladi.

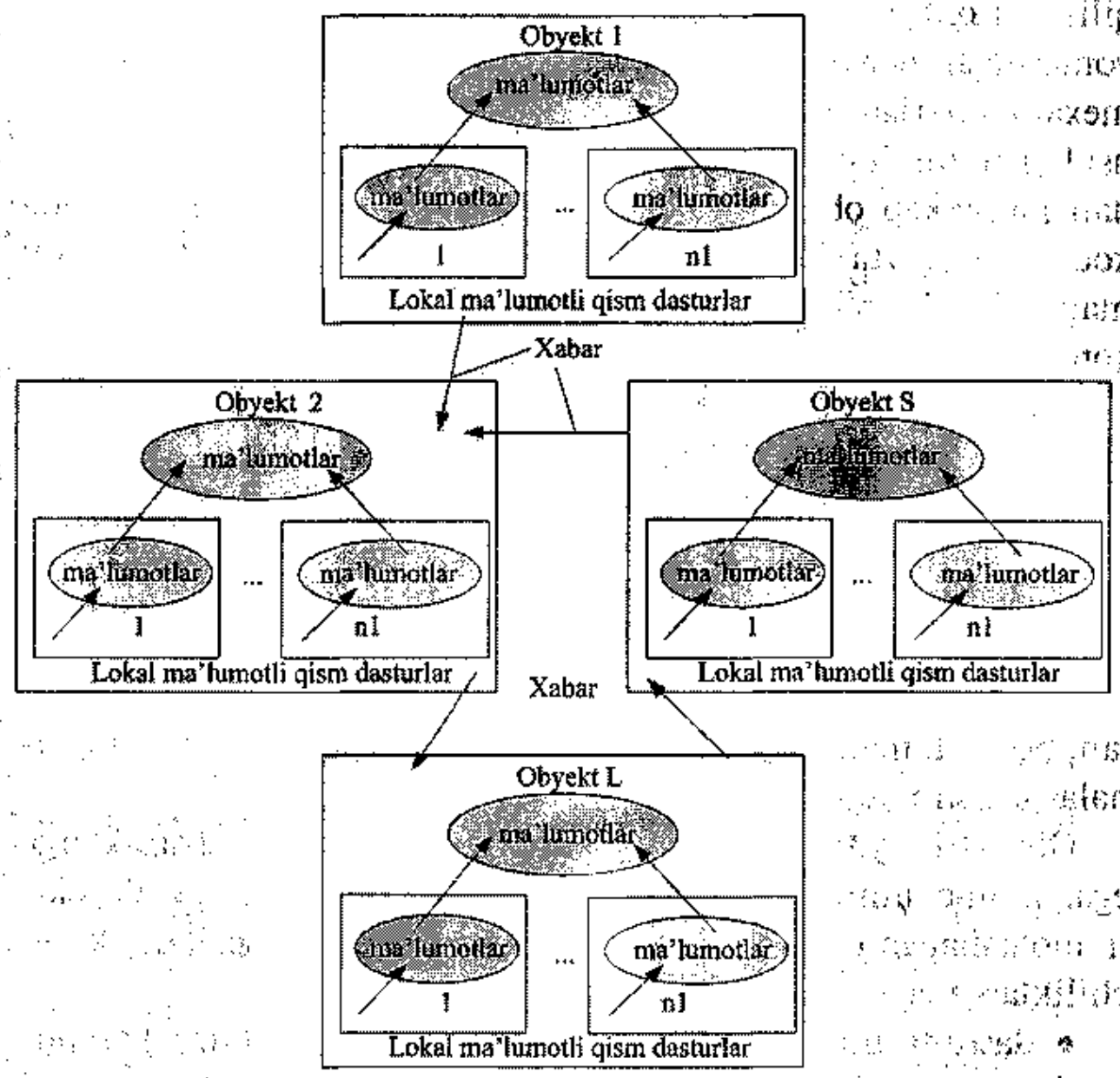
Obyektli yondashishga asoslangan dasturlash texnologiyalarining jadal rivojlanishi ko'p muammolarni hal etishga yordam berdi. Shu tariqa vizual dasturlashni ta'minlovchi muhitlar yaratildi. Masalan, Delphi, C++, Builder, Visual C++ va h.k. Vizual muhitdan foydalanganda dasturchida maxsus kutubxona komponentlarini qo'shish va sozlash vizual vositalarini qo'llagan holda ba'zi qismlarni loyihalashtirish imkoniyati paydo bo'ladi. Masalan, bo'lajak mahsulot interfeyslari muvofiq kodlar kiritilgan loyihalashtirish natijasidir.

Obyektli yondashishdan foydalanish ko'p afzalliklarga ega, ammo uning Pascal va C++ kabi dasturlashning obyektli mo'ljallangan tillarda aniq amalga oshirilishi sezilarli kamchiliklarga ega:

• dasturlashning hatto bir tili doirasida obyektlar kompilyatsiyasining ikkilik natijalarini joylashtirish standartlari mavjud emas: C++ ning turli kompilyatorlari orqali olingan obyektlarning joylashtirilishi muammosidir. Bu yuqori darajadagi das-

turlash bitta tilining hamda, bitta kompilyatorning vositalari va imkoniyatlaridan foydalangan holda dasturiy ta'minotni ishlab chiqish zaruratiga olib keladi, demak, klasslarning foydalaniladigan kutubxonalari boshlang'ich kodlarining bo'linishini talab qiladi;

• dasturiy obyektlardan birining amalga oshirilishining o'zgartirilishi, kamida, tegishli modulning qayta kompilyatsiyalanishi va ushbu obyektdan foydalanuvchi butun dasturiy ta'minlashni qayta joylashtirish bilan bog'liq.

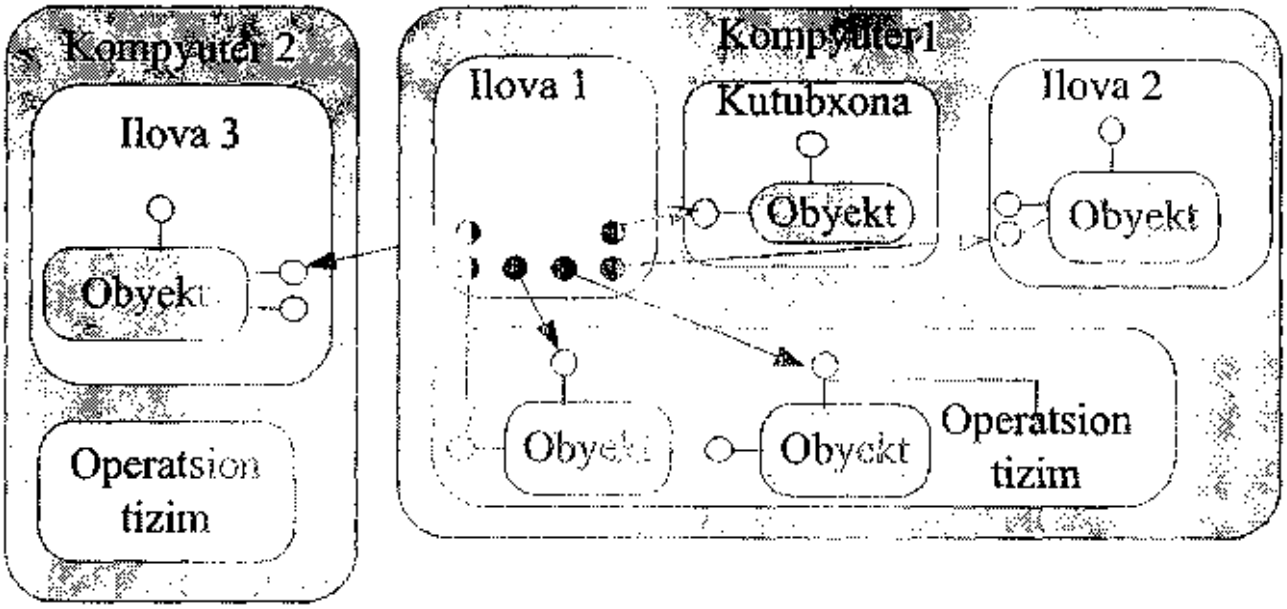



**1.6-rasm.** Obyektga mo'ljallangan dasturlashdagi dasturlar arxitekturasi.

Shu tariqa, dasturlashning bu tillaridan foydalanishda dasturiy ta'minot modullarining eksport qilinayotgan maydonlar va usullarning manzillari, shuningdek, ma'lumotlarning tarkibi va formatlariga bog'liqligi saqlanib qoladi. Bu bog'liqlik obyektiv. Chunki, modullar bir-birlarining resurslariga murojaat qilib o'zaro ta'sir ko'rsatishlari kerak. Modullar aloqasini uzish mumkin emas, ammo ularning o'zaro ta'sirini standartlashtirishga urinish mumkin, dasturlashga komponentli yondashish ham shunga asoslangan.

**To'rtinchi bosqich – komponentli yondashuv va CASE texnologiya** (XX asrning 90-yillaridan boshlab to hozirgi paytgacha). *Komponentli yondashish* – standartlashtirilgan ikkili interfeyslar orqali o'zaro ta'sir ko'rsatuvchi dasturiy ta'minotning fizik jihatdan alohida mavjud bo'lgan qismlar – dasturiy ta'minotning alohida komponentlaridan qurilishini ko'zda tutadi. Oddiy obyektlardan farqli ravishda obyekt komponentlarni dinamik chaqiriladigan kutubxonalar yoki bajariladigan fayllarga yig'ish, ikkili ko'rinishda (boshlang'ich matnlarsiz) tarqatish va muvofiq texnologiyani ta'minlovchi dasturlashning har qanday tilida foydalanish mumkin. Bugungi kunda obyektlar bozori haqiqatga aylandi, chunki Internetda ko'p miqdordagi komponentlarni taqdim etuvchi bog'lamalar mavjud, jurnallar komponentlar reklamasi bilan to'libtoshgan. Bu dasturchilarga hech bo'lmaganda qisman qayta foydalanilgan qismlardan iborat mahsulotlarni yaratish, ya'ni apparaturani loyihalashtirish sohasida o'zini ijobiy tomondan ko'rsatgan texnologiyadan foydalanish imkonini beradi.

COM (Component Object Model – obyektlarning komponentli modeli) bazasida ishlab chiqilgan texnologiyalar va CORBA (Common Object Request Broker Architecture) – obyektlarning so'rovlariga ishlov berish vositachisi bilan birga umumiy arxitektura taqsimlangan ilovalarini yaratish texnologiyalari asosida komponentli yondashish mavjuddir. Bu texnologiyalar o'xshash tamoyillaridan foydalaniladi va faqat ularning amalga oshirilishi xususiyatlari bilan farqlanadi.




**1.7-rasm.** Turli xildagi dasturiy komponentlarning o'zaro ta'siri.

Microsoft firmasining COM texnologiyasi Windows ning ilk versiyalarida tarkibiy hujjatlarni yaratish uchun foydalanilgan OLE 1 texnologiyasi (Object Linking and Embedding – obyektlarning bog'lanishi va joriy etilishi)ning rivoji hisoblanadi. COM texnologiyasi har qanday turdagi dasturlar o'zaro ta'sirining umumiy paradigmasini belgilaydi. Kutubxonalar, ilovalar, operatsion tizimning, ya'ni dasturiy ta'minotning qismlari bir jarayon doirasida, bitta kompyuterdagi turli jarayonlarda yoki turli kompyuterlarda ishlashidan qat'i nazar dasturiy ta'minotning bir qismiga ikkinchi qismi taqdim etayotgan funksiya (xizmat)lardan foydalanish imkonini beradi (1.7 rasm).

Kompyuterlar o'rtasidagi chaqiruvchilar uzatilishini ta'minlovchi COM ni modifikatsiyalash DCOM (Distibutes COM – taqsimlangan COM) deb ataladi.

COM texnologiyasiga ko'ra, SOM klasslarining nusxalari hisoblangan maxsus obyektlar, COM obyektlaridan foydalanib, ilova o'z xizmatlarini taqdim etadi. COM obyekti oddiy obyekt singari maydonlar va usullarni kiritadi, ammo oddiy obyektlardan farqli ravishda COM ning har bir obyekti uning maydonlari va funksiyalaridan foydalanishni taqdim etuvchi bir qancha in-

terfeyslarni amalga oshirishi mumkin. Bunga har bir interfeys uchun usullar manzillarining alohida jadvalini tashkil etish hisobiga erishiladi (virtual usullar jadvallarining turi bo'yicha). Bunda interfeys bir qancha turdagi funksiyalarni birlashtiradi. Bundan tashqari, COM klasslari interfeyslarining meros qilib olinishini ta'minlaydi, ammo amalga oshirishning meros qilib olinishini ta'minlamaydi, ya'ni usullar kodini meros qilib olmaydi, garchi zarurat bo'lganda ham avlod klassi obyekti «ota-ona» usulini keltirib chiqarishi mumkin.

Har bir interfeys «I» simvoldan boshlanuvchi (Interface I Dentifier) nomga ega global noyob identifikator hisoblanib IID ga ega. SOM ning har bir obyekti I Unknown interfeysini albatta amalga oshiradi (sxemalarda bu interfeys doim tepasida joylashtiriladi). Bu interfeysdan foydalanish obyektning boshqa interfeyslaridan foydalanishni qo'lga kiritish imkonini beradi.

Obyekt doimo obyektning ishlashini ta'minlovchi server – dinamik kutubxona yoki bajariladigan faylning tarkibida ishlaydi. Serverlarning uchta turi farqlanadi:

- ichki server – ilova – mijozga ulanadigan va u bilan bitta manzilli fazoda ishlaydigan dinamik kutubxonalar orqali amalga oshiriladi. U eng effektiv server, bundan tashqari maxsus vositalarni talab qilmaydi.
- Lokal server – mijoz bilan bitta kompyuterda ishlaydigan alohida jarayon (yexe moduli) yordamida hosil qilinadi.
- Uzoqlashtirilgan server – boshqa kompyuterda ishlaydigan jarayon yordamida hosil qilinadi.

Masalan, Microsoft Word lokal server hisoblanadi. Unga boshqa ilovalar foydalanishi mumkin bo'lgan ko'pgina obyektlar kiradi.

Xizmatlarga murojaat qilish uchun mijoz tegishli interfeysga ko'rsatma olishi kerak. Obyektga birinchi murojaatdan oldin mijoz tizimda qayd qilingan barcha COM obyektlarining klasslari to'g'risidagi axborotni saqlovchi COM kutubxonasiga so'rov yuboradi va unga klass nomi, interfeys identifikatori va server

turini topshiradi. Kutubxona kerakli serverni ishga tushiradi, talab qilinadigan obyektlarni hosil qiladi va obyektlar hamda interfeyslar uchun ko'rsatmalarni qaytaradi. Ko'rsatmalarni olgandan keyin mijoz obyektning kerakli funksiyasini chaqirishi mumkin.

Mijoz va serverning o'zaro ta'siri COM yoki DCOM ning bazaviy mexanizmlari bilan ta'minlanadi, shuning uchun mijozga obyekt joylashgan o'rnining farqi yo'q. Lokal va uzoqlashtirilgan serverlardan foydalanishda mijozning manzilli fazosida proxy – obyekt hosil qilinadi – COM obyektining o'rinbosari, COM serverining manzilli fazosida mijozga to'g'ri keluvchi tiqin hosil qilinadi. Mijozdan topshiriq olib, o'rinbosar uning parametrlarini joylashtiradi va operatsion tizim xizmatlaridan foydalanib tiqinga chaqiruvni uzatadi. Tiqin topshiriqni ochib uni COM obyektiga topshiradi. Natijada mijozga qayta tartibda qaytib keladi.

COM va uning taqsimlangan versiyasi DCOM bazasida dasturiy ta'minotni ishlab chiqishning turli masalalarini hal qiluvchi komponent texnologiyalari ishlab chiqilgan edi.

*«OLE – automation»* yoki oddiygina »Automation» (avtomatlashtirish) –ilovalarning ichki xizmatlaridan foydalanishni ta'minlovchi dasturlashtiriladigan ilovalarni hosil qilish texnologiyasi Dispinterfeys (dispinterface) tushunchasini kiritadi – obyekt funksiyalarini chaqiruvini yengillashtiruvchi maxsus interfeys. Bu texnologiyani, masalan, boshqa ilovalarga o'z xizmatlarini taqdim etgan holda Microsoft Excel ta'minlaydi.

*«ActiveX – OLE automation»* bazasida qurilgan texnologiya ham bitta kompyuterga qaratilgan, ham tarmoqda taqsimlangan dasturiy ta'minlashni hosil qilish uchun vizual dasturlashdan foydalanishni ko'zda tutadi. Shu tariqa hosil qilingan boshqarish elementlarini uzoqlashtirilgan serverdan masofadan turib kompyuterga o'rnatish mumkin, bunda o'rnatiladigan kod foydalanilayotgan operatsion tizimga bog'liq. Bu Internet ilovalarining mijozli qismlarida ActiveX boshqarish elementlarini qo'llash imkonini beradi.

ActiveX texnologiyasining tarqalishini ta'minlovchi asosiy afzalliklar quyidagilar:

• dasturiy kodning tez yozilishi – server va mijozning o'zaro ta'sirining tashkil qilinishi bilan bog'liq barcha ishlarni COM dasturiy ta'minlashga olganligi sababli, tarmoqli ilovalarni dasturlash alohida kompyuter uchun dasturlashga o'xshab ketadi;

• oshkoralik va *mobillik* – texnologiyaning spetsifikatsiyalanishi oshkora standart sifatida yaqinda Open Group ga topshirilgan edi;

• ishlab chiqishning tanish bo'lgan vositalaridan masalan, Visual Basic, Visual C++, Borland Delphi, Borland C++ va Java da ishlab chiqishning har qanday vositalaridan foydalanib ilovalarni yozish imkoniyati;

• katta miqdorda mavjud bo'lgan bepul ActiveX dasturiy elementlari (har qanday OLE dasturiy komponenti ActiveX texnologiyalari bilan moslashgan va tarmoqli ilovalarda modifikatsiyalarsiz qo'llash mumkin);

• standartlilik – ActiveX texnologiyasi bir tomondan keng foydalaniladigan Internet standartlari (TCP/IP, HTML, Java)ga va bir vaqtlar Microsoft tomonidan kiritilgan, hamda, moslashuvni saqlab qolish uchun zarur bo'lgan COM, OLE standartlarga asoslangan.

*MTS* (Microsoft Transaction Server – tranzaksiyalarni boshqarish serveri) – uzatilayotgan ma'lumotlarning katta hajmlarida xavfsizlik va taqsimlangan ilovalarining barqaror ishlashini ta'minlaydigan texnologiya.

*MIDAS* (Multitier Distributed Application Server – ko'p bo'g'inli taqsimlangan ilovalar serveri) tarmoq yuklanishini muvozanatlashtirishini hisobga olib turli kompyuterlar ma'lumotlaridan foydalanishni tashkil qiluvchi texnologiya.

Barcha ko'rsatilgan texnologiyalar COM ga kiritilgan komponentli yondashuvni amalga oshiradi. Shunga ko'ra, COM nuqtayi nazaridan ActiveX boshqaruv elementi – OLE – automation texnologiyasini ta'minlovchi ichki server. Dasturchi uchun esa,

ActiveX elementi – ilovalarini yaratishda qurilish bloki sifatida foydalanishi mumkin bo'lgan xususiyatlar, usullar va voqealarga ega «qora quti».

OMG kompaniyalari guruhi (Object Management Group – dasturlashning obyektli texnologiyasini joriy qilish guruhi) tomonidan ishlab chiqilgan CORBA texnologiyasi CORBA obyektlari va interfeyslari asosida SOM ga o'xshash yondashuvlarni amalga oshiradi. CORBA ning dasturiy o'zagi barcha asosiy apparatli va dasturiy platformalar uchun amalga oshirilgan va shuning uchun bu texnologiyadan geterogen (turli) hisoblash muhitida taqsimlangan dasturiy ta'minotni hosil qilish uchun foydalanish mumkin. CORBA dagi mijoz va serverning obyektlari o'rtasida o'zaro ta'sirni tashkil qilish Visi Broker deb nomlangan maxsus vositachi va boshqa ixtisoslashtirilgan dasturiy ta'minot yordamida amalga oshiriladi.

Yondoshuvni o'zgartirishdan tashqari dasturlash texnologiyasining hozirgi zamon rivojlanish bosqichida o'ziga xos xususiyatga ega va CASE texnologiya deb nom olgan (Computer – Aided Software / System Engineering – kompyuter texnologiyasini qo'llab-quvvatlovchi dasturiy ta'minotni / dasturiy tizimni yaratish) avtomatlashtirilgan ishlanma texnologiyasini yaratish, joriy etish hamda dasturiy ta'minotni kuzatib borish tushuniladi. Avtomatlashtirish vositalarisiz hozirgi paytda ancha murakkab dasturiy ta'minotni ishlab chiqishni amalga oshirish qiyin. Inson xotirasi dasturiy ta'minotni ishlab chiqishda hisobga olishi kerak bo'lgan barcha tafsilotlarni qayd qilishga qodir emas. Bugungi kunda dasturlashga ham tarkibiy, ham obyektli (jumladan, obyektli) yondashuvlarni ta'minlovchi CASE texnologiyalari mavjud.

Yangi yondashuvning paydo bo'lishi butun dasturiy ta'minot dasturiy komponentlardan hosil qilinishini bildirmaydi, ammo murakkab dasturiy ta'minotni ishlab chiqishning mavjud muammolari tahlili ancha keng ko'lamda qo'llanishini ko'rsatadi.

rayonida nisbatan past darajadagi tillardan foydalaniladi. Bu dasturiy ta'minotni yaratish jarayonida operatsiyalarning erta detallashtirilishiga olib keladi va dasturlash tilining yuz minglab operatorlaridan oshib ketuvchi ishlab chiqiladigan mahsulotlarni tavsiflash hajmini ko'paytiradi. Dasturlashning universal tilidan yanada yuqoriroq darajadagi murakkab diskretli tizimlar xatti-harakatlarini batafsil tavsiflash imkonini beruvchi vositalar esa mavjud emas.

*Jamoa bo'lib ishlab chiqish.* Loyihalarning katta hajmlari tufayli dasturiy ta'minotni ishlab chiqish mutaxassislar jamoasi tomonidan olib boriladi. Jamoada ishlab turib, alohida mutaxassislar loyihaning butunligini ta'minlab, bir-birlari bilan o'zaro aloqada bo'lishlari kerak, bu yuqorida eslatib o'tilgan murakkab tizimlar xatti-harakatlarini tavsiflashning qoniqarli vositalari yo'qligida ancha murakkab. Shu bilan birga, ishlab chiquvchilar jamoasi qanchalik katta bo'lsa, ish jarayonini tashkil etish shunchalik qiyin.

*Kodlar takrorlanishi darajasining ortishi zaruriyati.* Ishlab chiqilayotgan dasturiy mahsulotning murakkabligiga mehnat unumdorligini oshirish uchun kompaniyalar keyingi ishlab chiqishlarda foydalanishi mumkin bo'lgan komponentlar kutubxonasini yaratishda intilishlari ham ta'sir ko'rsatadi. Ammo bu holatda komponentlarni yanada universal qilishga to'g'ri keladi, oqibat natijada bu ishlab chiqish murakkabligini oshiradi.

Birgalikda olinganda bu omillar ishlab chiqish jarayonining murakkabligini ancha oshiradi. Ammo, shunisi ayonki, ularning hammasi ishlab chiqish obyekti — dasturiy tizimning murakkabligi bilan to'g'ridan to'g'ri bog'langan.

## 1.4. Murakkab tizimlarning yaratilishida blokli — iyerarxiyali yondashuv

Amaliyot shuni ko'rsatadiki, murakkab tizimlarning ko'pchiligi tabiatdagi kabi, texnikada ham iyerarxiyali ichki tuzilishga ega. Bu odatda, murakkab tizimlar elementlarining aloqalari ham turi, ham kuchi bo'yicha turliligi bilan bog'liq, bu o'z navbatida,

shu tizimlarni o'zaro bog'liq kichik tizimlar qandaydir majmuasi sifatida qarash imkonini beradi. Bunday kichik tizimlar elementlarining ichki aloqalari kichik tizimlar o'rtasidagi aloqalardan kuchliroq. Masalan, kompyuter protsessor, xotira va tashqi qurilmalardan iborat. Quyosh tizimiga esa, quyosh va uning atrofida aylanuvchi sayyoralar kiradi.

O'z navbatida, aloqalarning shu farqidan foydalanib, har bir kichik tizimni yanada kichik tizimlarga bo'lish mumkin va shu tariqa eng quyi «elementar» darajagacha davom ettirish mumkin, shu bilan birga, komponentlarni elementar deb hisoblanishi kerak bo'lgan darajani tanlash imkoniyati tadqiqotchiga beriladi. Elementar darajada tizim qoidaga ko'ra, turlicha kombinatsiyalangan va tashkil qilingan kichik tizimlarning ozgina turlaridan iborat. Bunday turdagi iyerarxiyalar «butunlik – qism» nomini olgan.

Tizimning butun holdagi xatti-harakati, odatda, alohida qismlarning xatti-harakatlaridan murakkabdir, shu bilan birga yanada kuchliroq ichki aloqalar tufayli tizimning xususiyatlari, asosan, uning qismlari sifatida emas, qismlari o'rtasidagi munosabatlar bilan asoslangan.

Tabiatda iyerarxiyaning yana bir turi mavjud – «oddiy va murakkab» iyerarxiya yoki evolyutsiya jarayonidagi tizimning rivojlanishi (murakkablashishi) iyerarxiyasi. Bu iyerarxiyada ishlaydigan har qanday tizim yanada soddaroq tizimning rivojlanish natijasidir. Iyerarxiyaning aynan shu turi obyektli – taxminiy dasturlashning meros qilib olingan mexanizmi orqali amalga oshiriladi.

Ko'p darajada tabiiy va texnik tizimlarning aks etishi bo'lib, dasturiy tizimlar odatda, iyerarxiyali hisoblanadi, ya'ni yuqorida tavsiflangan xususiyatlarga ega. Iyerarxiyali tizimlarning bu xususiyatlariga ularni tadqiq qilish yoki yaratishga blokli – iyerarxiyali yondashuv quriladi. Bu yondashuv bunday obyektlarning qismlari (bloklar, modullar)ni yaratishni, keyin esa, ulardan obyektning o'zini qurishni mo'ljallaydi.

Murakkab obyektni nisbatan mustaqil qismlarga ajratish jarayoni dekompozitsiya nomini olgan. Dekompozitsiyada alohida qismlar ichidagi elementlar aloqasidan ko'ra kuchsizroq bo'lishi hisobga olinadi. Bundan tashqari, olingan qismlardan ishlab chiqilayotgan obyektni yig'ish uchun dekompozitsiya jarayonida qismlar o'rtasidagi aloqalarning barcha turlari belgilanishi kerak.

Juda murakkab obyektlarni yaratishda dekompozitsiya jarayoni ko'p marotalab bajariladi: har bir blok, o'z navbatida, ishlab chiqish nisbatan oson bo'lgan bloklar olinmaguncha qismlarga ajratiladi. Ishlab chiqishning ushbu usuli qadam-baqadam detallashtirish nomini olgan.

Shunisi ham ahamiyatliki, dekompozitsiyalash jarayonida umumiy asosda ishlab chiqish mumkin bo'lgan o'xshash bloklarni ajratishga harakat qiladilar. Shu tariqa, yuqorida eslab o'tilganidek, kodlarning takrorlanish darajasining ortishi muvofiq ravishda, ishlab chiqish qiymatining pasayishi ta'minlanadi.

Dekompozitsiyalash natijasi, odatda, iyerarxiyaning sxemasi sifatida taqdim etiladi, uning quyi darajasida nisbatan sodda bloklar, yuqori darajasida ishlab chiqilishi kerak bo'lgan obyekt joylashtiriladi. Har bir iyerarxiyali darajada bloklarning tavsiflanishi detallashtirishning ma'lum darajasi bilan bajariladi, ahamiyatga ega bo'lmagan detallardan mavhumlashgan holda amalga oshiriladi. Demak, har bir daraja uchun har bir blok tomonidan bajariladigan jarayonlar mohiyatini aks ettiruvchi hujjatlashtirishning o'z shakllari va o'z modellaridan foydalaniladi. Shu tariqa umumiy obyekt uchun, odatda, faqat eng umumiy talablarni ifodalash mumkin bo'ladi, quyi daraja bloklar esa, ulardan haqiqatda ham ishlaydigan obyektni yig'ish mumkin bo'lgan darajada spetsifikatsiyalanishi kerak.

Boshqacha qilib aytganda, blok qanchalik katta bo'lsa, uning tavsifi shunchalik mavhum bo'lishi kerak (1.8-rasm).

Bu tamoyilga rioya qilgan holda, ishlab chiquvchi loyihani tushunib yetish imkoniyatini saqlab qoladi va demak, har bir

bosqichda eng to‘g‘ri qaror qabul qilish mumkin, bu lokal optimallashtirish deb ataladi (haqiqatda ham murakkab obyektlar uchun har doim ham iloji bo‘lmagan obyektlar tavsiflarini global optimallashtirishdan farqli ravishda).

**Eslatma.** Shuni nazarda tutish kerakki, texnologiyalarning tobora mukammallashgani sayin murakkab obyekt tushunchasi o‘zgaradi va kecha murakkab bo‘lgan narsa ertaga ham murakkab bo‘lib qolmaydi.

Shunday qilib, blokli iyerarxiyali yondashuv asosida dekompozitsiya qandaydir iyerarxiyali tartiblashtirish mavjud. Shuningdek, quyidagi tamoyillar ham muhim ahamiyatga ega:

- ziddiyatli emasligi – elementlarning o‘zaro kelishuvini nazorat qilish;
- to‘liqlilik – ortiqcha elementlarning mavjudligini nazorat qilish;
- formallashtirish – uslubiy yondashuvning qat’iyligi;
- takrorlanish – ishlab chiqishni arzonlashtirish va tezlashtirish uchun bir xil bloklarni ajratishning zarurligi;
- lokal optimallashtirish – iyerarxiyali darajasi doirasida optimallashtirish;

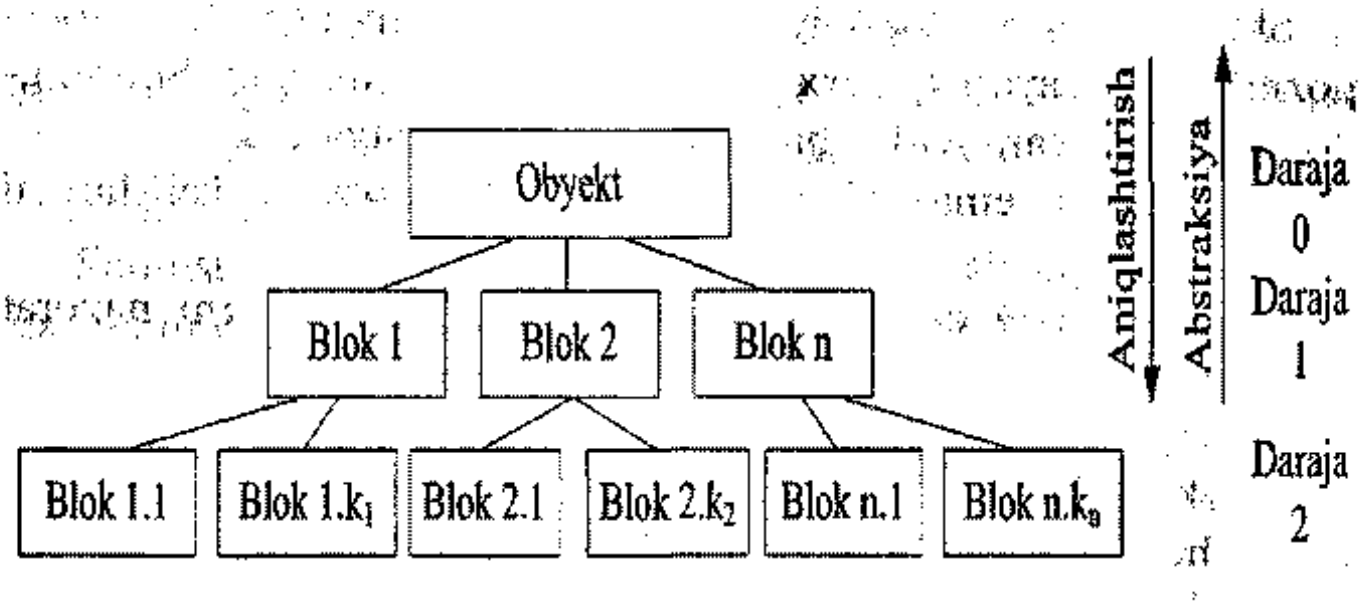
Modellar tillari, vazifalarning qo‘yilishi, ba’zi iyerarxiyali darajasini tavsiflash usullarining majmuasini *loyihalashtirish darajasi* deb atash qabul qilingan.

Loyihalashtirish jarayonidagi har bir obyektni, odatda, bir necha tomondan ko‘rib chiqishga to‘g‘ri keladi. Loyihalashtirish obyektiga bo‘lgan turli qarashlarni loyihalashtirish aspektlari deb atash qabul qilingan.

Blokli – iyerarxiyali yondashuvdan foydalanish murakkab tizimlar hosil qilinishiga imkon berishidan tashqari, shuningdek:

- umuman tizimning, shuningdek, alohida bloklarning ishlash qobiliyatining tekshirilishini osonlashtiradi;
- tizimlarni modernizatsiyalash, masalan, ishonchli bo‘lmagan bloklar interfeyslarini saqlagan holda, ular almashtirilishining imkoniyatini ta’minlaydi.

Shuni qayd etish kerakki, dasturiy tizimlarga nisbatan blokli – iyerarxiyali yondashuvdan foydalanish faqat yondashuvning umumiy qoidalari aniqlashtirilganidan va loyihalashtirish jarayoniga ba'zi bir o'zgarishlar kiritilganidan keyin mumkin bo'lib qoldi. Bunda tarkibiy yondashuv faqat iyerarxiyaning «butun – qism» xususiyatlarini hisobga oladi, obyektli yondashuv esa iyerarxiyaning «sodda – murakkab» xususiyatlaridan foydalanadi.




**1.8-rasm.** Blokli-iyerarxiyali yondashuvdagi bloklarning tavsifidagi mavhumlik va aniqlik munosabati.

## 1.5. Dasturiy ta'minotning hayotiy sikli va ishlab chiqish bosqichi

Dasturiy ta'minotning hayotiy sikli deb, ayrim dasturiy ta'minotni yaratish g'oyasi boshlangan vaqtdan uning ishlab chiquvchi firma yoki kuzatuvchi vazipredmeti bajaruvchi firma tomonidan qo'llab-quvvatlash tugagan vaqtgacha bo'lgan davrga aytiladi.

Hayotiy sikl jarayonining tarkibi xalqaro standart ISO/IEC 12207:1995 «Information Technology – Software Life Cycle Processes» («Axborot texnologiyalari – dasturiy ta'minot hayotiy siklining jarayoni») bilan tartibga solinadi. ISO – International

Organization for Standardization – Standartlashtirish xalqaro tashkiloti.

IEC – International Electro technical Commission – Elektrotexnika bo'yicha xalqaro komissiya. Ushbu standart dasturiy ta'minot hayotiy siklining tuzilmasini va uning jarayonini tavsiflaydi.

Hayotiy sikl jarayoni ayrim kirish ma'lumotlarining chiqish ma'lumotlariga o'zgaruvchi o'zaro bog'liq bo'lgan amallar majmui kabi aniqlanadi. 1.9-rasmda standart bo'yicha hayotiy sikl jarayonlari ko'rsatilgan. Har bir jarayon aniq vazifalar va usullarning xulosalari, shuningdek, dastlabki ma'lumotlari va natijalari bilan xarakterlanadi.

Ishlab chiqish jarayoni (Development process) standartiga muvofiq ishlab chiquvchi tomonidan bajarilayotgan amallar va vazifalarni ko'zda tutadi, hamda, berilgan talablarga muvofiq dasturiy ta'minot va uning komponentlarini yaratish bo'yicha ishlarni, jumladan, loyiha va ekspluatatsion hujjatlarni rasmiylashtirishni, shuningdek, ishlash qobiliyatini va xodimlarni o'qitish uchun zarur bo'lgan dasturiy ta'minot materiallar sifatiga muvofiqligini tekshirish uchun zarur materiallar tayyorlashni qamrab oladi.

Standart bo'yicha ishlab chiqish jarayoni quyidagi amallarni o'z ichiga oladi:

- *tayyorgarlik ishi* – standartlar, usullar va ishlab chiqish vositalari hayotiy siklining modelini tanlash, shuningdek, ishlar rejasini tuzish;
- *tizimga qo'yiladigan talablarni tahlil qilish* – uning funksional imkoniyatlarini, ishonchlilik va xavfsizlikka qo'yiladigan talablar, tashqi interfeyslarga qo'yiladigan talablar va hokazolarni aniqlash;
- *tizim arxitekturasini loyihalashtirish* – zarur uskuna, dasturiy ta'minot tarkibi va xizmat ko'rsatuvchi xodimlar tomonidan bajariladigan amallarni aniqlash;
- *dasturiy ta'minotga qo'yiladigan talablarni tahlil qilish* – funksional imkoniyatlarni jumladan, ishlab chiqarish tavsifi,

komponentlar, tashqi interfeyslarning ishlash muhitini, ishonchlilik va xavfsizlik talablarni foydalaniladigan ma'lumotlarga, o'rnatishga, qabul qilishga, foydalanish uchun hujjatlarga, ekspluatatsiya qilishga va kuzatishga qo'yiladigan talablarni aniqlash;

• *dasturiy ta'minot arxitekturasini loyihalash* — dasturiy ta'minot tuzilmasini aniqlash, interfeyslar, uning komponentlarini hujjatlashtirish, foydalanish hujjatlarining dastlabki versiyasini, shuningdek, testlarga qo'yiladigan talablarni va integratsiya qilish rejasini ishlab chiqish;




**1.9-rasm.** Dasturiy ta'minot hayotiy zanjiri jarayonining tuzilmasi.

• *dasturiy ta'minotni batafsil loyihalash* – dasturiy ta'minot va ular o'rtasidagi interfeyslar komponentlarining batafsil izohi, foydalanish hujjatlarini yangilash, dasturiy ta'minot komponentlarining testlash rejasi va testlarga qo'yiladigan talablarni hujjatlashtirish, komponentlar integratsiya rejasini yangilash;

• *dasturiy ta'minotni kodlash va testlash* – har bir komponentni, shuningdek, testli protseduralarning jamini va ularni testlash uchun ma'lumotlarni ishlab chiqish va hujjatlashtirish, komponentlarni testlash, foydalanish hujjatlarini yangilash, dasturiy ta'minot integratsiya rejasini yangilash, dasturiy ta'minotni integratsiya qilish – integratsiya qilish rejasiga muvofiq dasturiy komponentlarni yig'ish va o'z xususiyatlariga mos keluvchi, hamda, berilgan ekspluatatsiya qilish sharoitlarida foydalanishga tayyor sifatida dasturiy mahsulotni kvalifikatsiyalash zarur bo'lgan mezonlar to'plamini yoki sharoitlardan iborat kvalifikatsion talablarga muvofiqligiga dasturiy ta'minotni testlash;

• *dasturiy ta'minotni malakaviy (kvalifikatsion) testlash* - dasturiy ta'minotni buyurtmachi ishtirokida uning ekspluatatsiya qilish talablariga muvofiqligi va tayyorgarligini namoyish qilish uchun testlash, bunda texnik va foydalanish hujjatlarining tayyorgarligi va to'liqligi tekshiriladi;

• *tizimning integratsiyasi* – barcha tizim komponentlarini, jumladan, dasturiy ta'minot va uskunani yig'ish;

• *tizimni malakaviy testlash* – talablarning tizimga muvofiqligi uchun tizimni testlash, hujjatning rasmiylashtirilishini va to'liqligini tekshirish;

• *dasturiy ta'minotni o'rnatish* – dasturiy ta'minotni buyurtmachining uskunasiga o'rnatish va uning ishlash qobiliyatini tekshirish;

• *dasturiy ta'minotni qabul qilish* – dasturiy ta'minotni malakaviy testlash natijalarini va butun tizimni baholash hamda, buyurtmachi bilan birgalikda baholashning natijalarini, buyurtmachiga dasturiy ta'minotning oxirgi uzatishini hujjatlashtirish.

Ushbu ko‘rsatib o‘tilgan amallarni dasturiy ta’minotni ishlab chiqishning quyidagi asosiy bosqichini shartli ajratgan holda guruhlanishi mumkin (qavslarda GOST 19.102–77 «ishlab chiqish bosqichi» bo‘yicha tegishli ishlab chiqish bosqichlari ko‘rsatilgan):

- masalaning qo‘yilishi («Texnik topshiriq bosqichi»);
- talablarni tahlil qilish va xususiyatlarni ishlab chiqish («Eskizli loyiha» bosqichi);
- amalga oshirish («Ishchi loyiha» bosqichi).

Ishlab chiqish jarayoni an’anaviy tarzda kuzatish bosqichini o‘z ichiga oladi (ushbu bosqich GOST bo‘yicha «Tadbiq qilish» bosqichiga mos keladi).

Biroq dasturiy ta’minotni ishlab chiqish industriyasida sodir bo‘ladigan o‘zgartirishlarga muvofiq xalqaro standart bo‘yicha ushbu jarayon alohida ko‘rib chiqiladi.

Bosqichlarni ajratish oldin qabul qilingan qarorlarni istalgan bosqichiga bog‘liq bo‘ladi.

**Masalaning qo‘yilishi.** Vazifalarni qo‘yish jarayonida dasturiy ta’minot vazifalari aniq shakllantiriladi va ularga qo‘yiladigan asosiy talablarni belgilaydi. Har bir talab dasturiy ta’minotning zarur yoki kerakli xususiyat bayonidan iborat. Ishlab chiqilayotgan dasturiy ta’minotni bajaradigan funksiyalarni aniqlaydigan funksional talablarga va uning faoliyat ko‘rsatish xususiyatini aniqlaydigan ekspluatatsion talablarga bo‘linadi.

Prototiplarga ega dasturiy ta’minotga qo‘yiladigan talablar amaldagi dasturiy ta’minotning tuzilmasini va tavsifini hisobga olgan holda, analogiya bo‘yicha aniqlanadi. Analogiyalarga ega bo‘lmagan dasturiy ta’minotga qo‘yiladigan talablarni shakllantirish uchun loyihadan oldingi deb nomlanadigan maxsus tadqiqotlarni o‘tkazish zarur. Bunday tadqiqotlar jarayonida vazifalarning hal etilganligi aniqladi, ular xulosalarining usullarini ishlab chiqiladi (agar ular yangi bo‘lsa) va ishlab chiqiladigan dasturiy ta’minotning ahamiyatli tavsifini belgilanadi. Loyiha-

dan oldingi tadqiqotlarni bajarish uchun ilmiy-tadqiqot ishlarini bajarishga shartnoma tuziladi.

Vazifalarni qo'yish jarayonning istalgan bosqichida qayd etadigan texnik topshiriqlarni ishlab chiqish va asosiy loyiha qarorlarini qabul qilish bilan tugallanadi.

**Talablarni tahlil qilish va xususiyatlarni aniqlash.** Xususiyatlar deb, funksiyalar va ishlab chiqilayotgan dasturiy ta'minot cheklanishlarining aniq tavsifiga aytiladi. Mos ravishda funksional va ekspluatatsion xususiyatlarga bo'linadi.

Barcha xususiyatlar loyihalashtirilayotgan dasturiy ta'minotning umumiy mantiqiy modelidan iborat.

Xususiyatlarni olish uchun texnik topshiriq talablari tahlil qilinadi, mazmunli vazifalar qo'yilishini shakllantiriladi, matematik shaklga keltirish apparati tanlanadi, predmet sohasining modeli quriladi, kichik topshiriqlar aniqlanadi va ularni hal etish usullari tanlanadi yoki ishlab chiqiladi. Xususiyatlarning qismi loyihalashtirishdan oldingi tekshirishlar jarayonida aniqlanishi mumkin va mos ravishda texnik topshiriqda qayd etiladi.

Ushbu bosqichda loyihalashtirilayotgan dasturiy ta'minotda, kutilayotgan natijani ko'rsatgan holda, xatolarni izlash uchun testni shakllantirish maqsadga muvofiq bo'ladi.

**Loyihalashtirish.** Ushbu bosqichning asosiy vazifalari bo'lib, ishlab chiqilayotgan dasturiy ta'minotning batafsil xususiyatlarini aniqlash hisoblanadi. Murakkab dasturiy ta'minotni loyihalash jarayoni, odatda, quyidagilarni o'z ichiga oladi:

- umumiy tuzilmani loyihalash – asosiy komponentlarni va ularning o'zaro aloqalarini aniqlash;
- komponentlarni dekompozitsiyalash va blok – iyerarxiyali yondashuv tavsiyalariga muvofiq tuzilmaviy iyerarxiyaning tuzilishi;
- komponentlarni loyihalashtirish.

Loyihalashtirishning natijasi bo'lib, ishlab chiqilayotgan dasturiy ta'minot, uning barcha turdagi darajalar komponentlari bilan birgalikdagi batafsil modeli hisoblanadi. Model tipi tanlangan

yondashuvdan (tuzilmaviy, obyektiv yoki komponentli) va loyihalashtirishning aniq texnologiyasiga bog'liq. Biroq, loyihalashtirish jarayonining istalgan holatida loyihalash dasturi (kichik dastur) va ular o'rtasidagi o'zaro aloqalarni aniqlash kabi ushbu dasturlar yoki kichik dasturlar o'zaro ishlaydigan ma'lumotlarni loyihalashtirishni ham qamrab oladi.

Loyihalashtirishni ikkita aspektga bo'lish mumkin:

• mantiqiy loyihalashtirish, o'z ichiga kelgusidagi dasturiy ta'minotning faoliyat ko'rsatish muhitini tashkil etuvchiga ega bo'lgan va dasturiy vositalarga bevosita bog'liq bo'lmagan loyiha operatsiyalarini oladi;

• fizik loyihalashtirish – faoliyat ko'rsatish muhitining aniq texnik va dasturiy vositalariga bog'liqligi, ya'ni o'ziga xos chegaralanishlarini hisobga olish.

**Amalga oshirish.** Tanlangan dasturlar kodining bosqichma-bosqich yozilish (kodlash), ularni testlash va tuzatib olish jarayonini amalga oshirishdir.

**Birga qo'shish.** Yangi versiya dasturiy ta'minotning yaratish va tadbiq qilish jarayonini birga qo'shishdir. Yangi versiyalarni chiqarish sabablari quyidagilar hisoblanadi:

• oldingi versiyalarni ekspluatatsiya qilish jarayonida aniqlangan xatolarni tuzatish zarurligi;

• oldingi versiyalarni takomillashtirish zarurligi, masalan, interfeysni yaxshilash, bajariladigan funksiyalar tarkibini kengaytirish yoki uning unumdorligini oshirish;

• faoliyat ko'rsatish muhitini o'zgartirish, masalan, birga qo'shiladigan dasturiy ta'minot, o'zaro ishlaydigan yangi texnik vositalar yoki dasturiy ta'minotlarning yuzaga kelishi.

Ushbu bosqichda dasturiy mahsulotga zarur o'zgartirishlar kiritiladi, qolgan holatlardagi kabi oldingi istalgan bosqichda qabul qilingan loyiha qarorlarini qayta ko'rib chiqish talab qilinishi mumkin.

Dasturiy ta'minot hayotiy siklining modeli o'zgarishi bilan ushbu bosqichning roli ancha oshdi, chunki mahsulotlar ite-

ratsion tarzda yaratilmoqda: avval nisbatan oddiy versiya ishlab chiqariladi, keyin katta imkoniyatlar, so'ngra keyingisi va hokazo. Xuddi shu, ISO/IEC12207 standartga muvofiq hayotiy siklning ayrim jarayonida birga qo'shilish bosqichini ajratishga sabab bo'ldi.

Ko'rib chiqilayotgan standart – dasturiy ta'minot hayotiy siklining jarayonlarini nomlaydi va aniqlaydi, u ushbu jarayonga kirgan amallar va vazifalarni qanday amalga oshirish yoki bajarishni batafsil aniqlashtirmaydi. Ushbu masalalar tegishli usullar, uslublar va boshqalar bilan tartibga solinadi. Oxirgilarni batafsil ko'rib chiqishga o'tishdan oldin yuzaga kelgan vaqtdan hozirgi vaqtgacha dasturiy ta'minotni ishlab chiqish sxemasini tahlil qilish zarur.

### 1.6. Dasturiy ta'minot hayotiy sikli modulining evolyutsiyasi

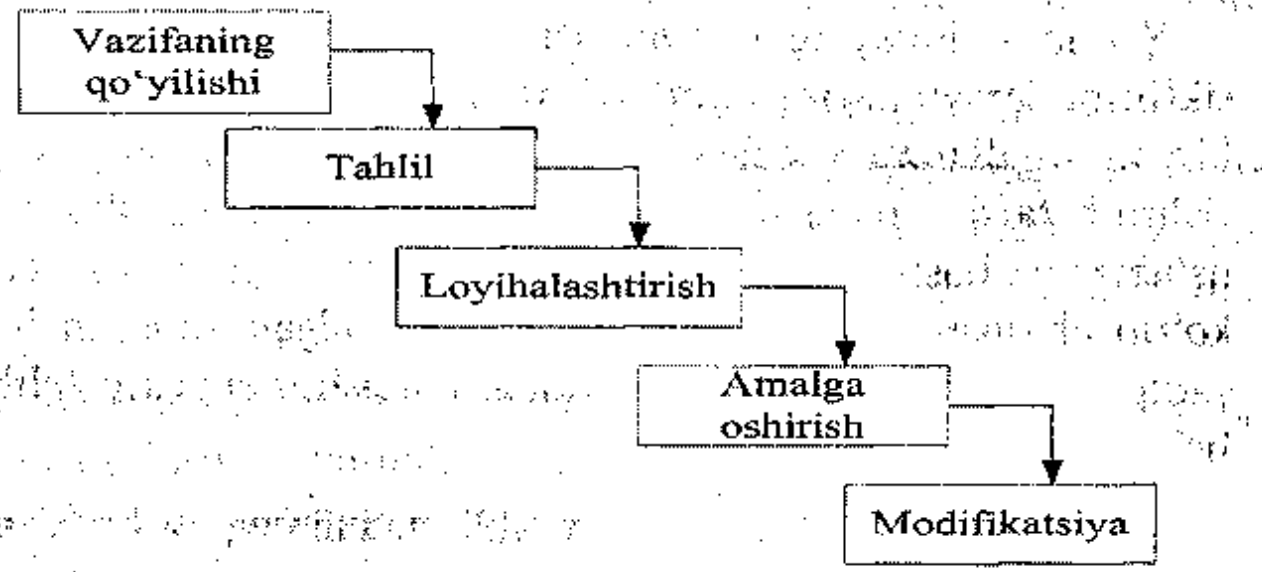
Oxirgi o'ttiz yil davomida dasturlashda dasturiy ta'minot hayotiy siklining quyidagi uchta modeli almashdi: kaskadli, oraliq nazoratli model, spiral model.

**Kaskad modeli.** Dastlab (1970–1985-yillar) keyingi bosqichga o'tish oldingi bosqichning loyiha operatsiyalari to'liq tugagandan keyin va keyingi bosqich uchun barcha dastlabki ma'lumotlar olingandan keyin amalga oshirilishi tavsiya qilinadigan dasturiy ta'minotni ishlab chiqishning kaskad sxemasidan foydalanildi (1.10-rasm). Ushbu sxemaning afzalliklari quyidagilar hisoblanadi:

- to'liqlik va moslashish talablariga javob beruvchi loyiha hujjatlarini tugallangan to'plamning har bir bosqich oxirida olinadi;
- ishlab chiqish jarayonini loyihalashtirishning oddiyligi.

Xuddi shunday sxema ishlab chiqish samaradorligining o'ta yuqori parametrlarini ta'minlab, murakkab texnik obyektlarni ishlab chiqishga blok – iyerarxik yondashuvda foydalaniladi. Ushbu sxema ishlab chiqish boshida barcha talablarni aniq va to'liq muvaffaqiyatli shakllantirish uchun tizimni yaratishda qo'llanishi mumkin.

Bu holat oldingi bosqichda muvaffaqiyatli qarorni qabul qilish bilan bog'liq muammolarni ishlab chiqish jarayonida yuzaga kelish ehtimolligini kamaytiradi. Amaliyotda bunday ishlab chiqishlar juda kam uchraydi.




**1.10-rasm.** Dasturiy ta'minotni ishlab chiqishning kaskadli sxemasi.

Oldingi bosqichda variantlarning zarurligi quyidagi sabablarga asoslanadi:

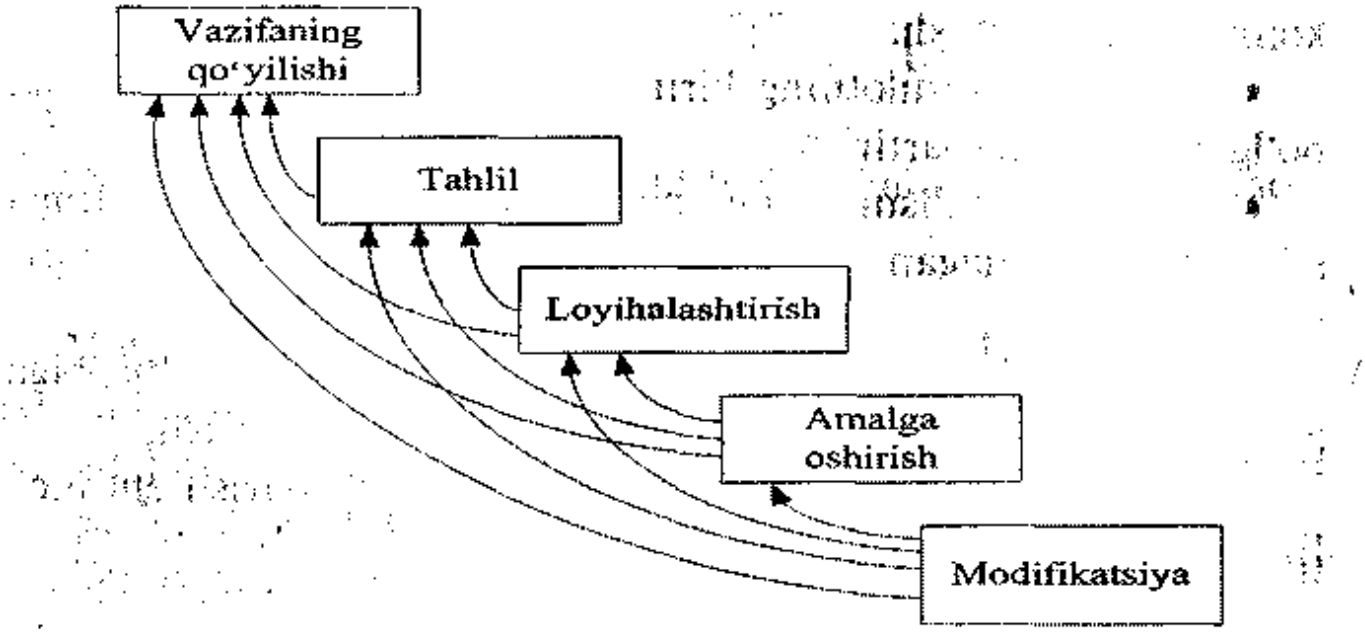
- ishab chiqish jarayonida aniqlashtirish qabul qilingan qarorlarni qayta ko'rib chiqish zarurligiga olib keladigan noaniq xususiyatlar;
- bevosita ishlab chiqish jarayonida buyurtmachi talablarining o'zgarishi;
- foydalanilayotgan texnik va dasturiy vositalarning ma'naviy tez eskirishi;
- predmetning qo'yilishi, tahlil, loyihalashtirish bosqichida ishlab chiqish bayonining qanoatlanitiruvchi vositalari mavjud emasligi.

Foydalanuvchi uskuna va dasturiy muhitning almashishini hisobga olishni rad etish natijasida ma'naviy eskirgan mahsulot olinadi. Muvaffaqiyatli bo'lmagan loyiha qarorlarni qayta ko'rib chiqishni rad etish dasturiy mahsulot tuzilmasini yomonlashtirishga va mos ravishda, yaratish jarayonining vaqt bo'yicha mu-

rakkablashishiga, cho'zilishiga va qimmatlashuviga olib keladi. Shunday qilib, ishlab chiqishning real jarayoni iteratsion xarakterga ega bo'ladi.

**Oraliq nazoratli model.** Ishlab chiqish jarayonining iteratsion xususiyatlarini ta'minlovchi sxema oraliq nazoratli sxema deb nomlanadi (1.11-rasm). Tugallangan har bir bosqichdan keyin ushbu sxema bo'yicha bajariladigan nazorat, zarur bo'lganda istalgan darajaga qaytarish imkoniga ega bo'ladi va zarur o'zgartirishlarni kiritadi.

Ushbu sxemadan foydalanishning asosiy xavfsizligi ishlab chiqarish hech qachon tugamasligi, doimo aniqlashtirish va takomillashtirish holatida bo'lganligi bilan bog'liqligi hisoblanadi.




**1.11-rasm.** Oraliq nazoratli dasturiy ta'minotni ishlab chiqish sxemasi.

**Spiral modeli.** Muammolarni yengish uchun XX asrning 80-yili o'rtalarida spiral sxemasi taqdim etildi (1.12-rasm). Ushbu sxemaga muvofiq dasturiy ta'minot prototipni yaratishga asoslangan prototiplash usulidan foydalangan holda, iteratsion holatda yaratiladi. Prototiplashning yuzaga kelishi dasturiy ta'minotni modifikatsiya qilish jarayoni «zarur yomonlik» kabi qabul qilinishdan to'xtalishiga alohida muhim jarayon kabi qabul qilinish boshlanishiga olib keladi.

Prototip deb, ishlab chiqilayotgan dasturiy ta'minotning va tashqi interfeyslarini amalga oshiradigan dasturiy mahsulotga aytiladi.

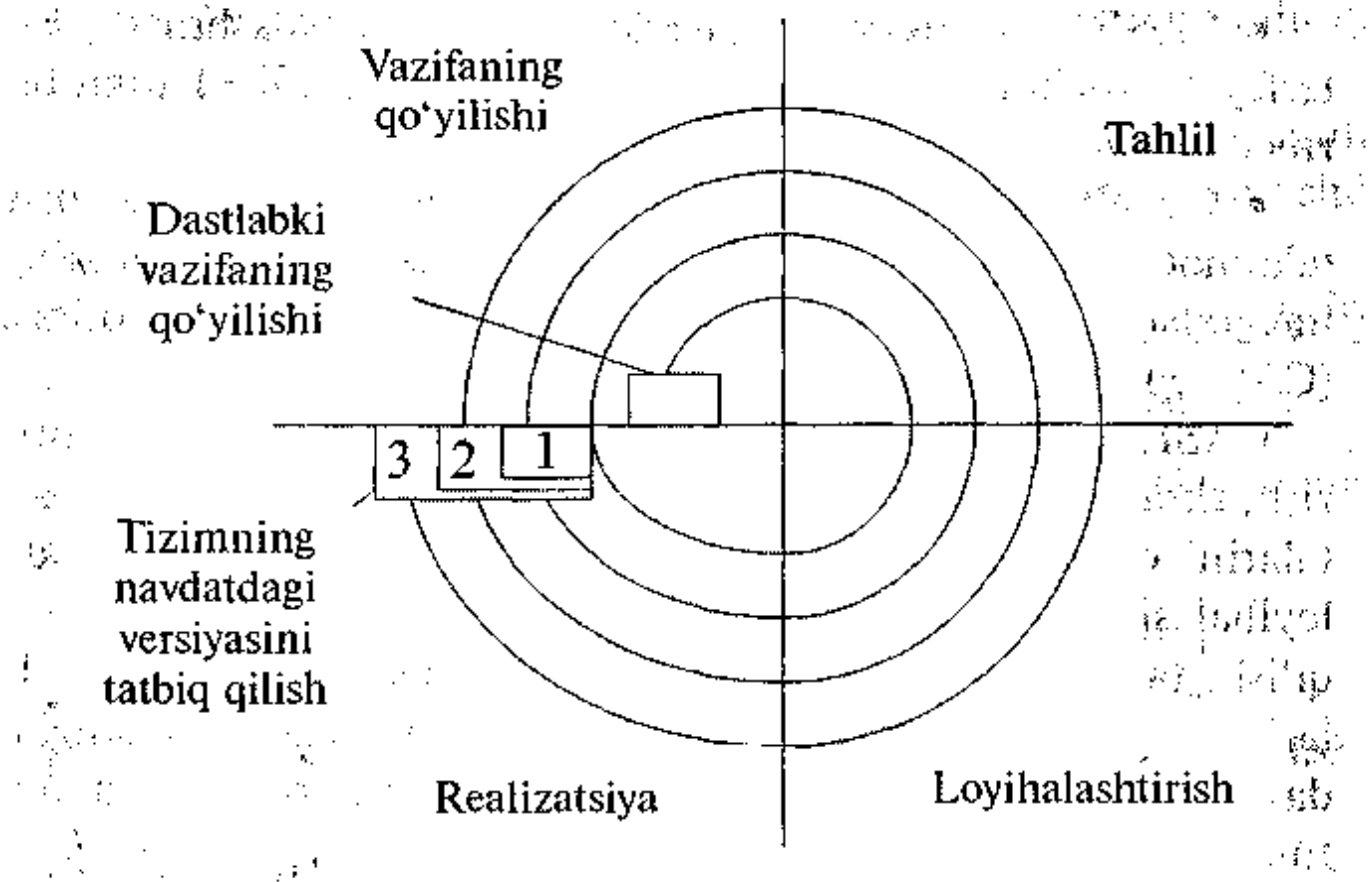
Qoidaga ko'ra, birinchi iteratsiya foydalanuvchining interfeysini spetsifikatsiya qiladi, loyihalashtiradi, amalga oshiradi va testlaydi. Ikkinchi iteratsiyada ayrim cheklangan funksiyalar to'plami bilan to'ldiriladi. Oxirgi bosqichda ushbu to'plam ushbu mahsulot imkoniyatini ko'paytirib kengaytirish hisoblanadi.

Ushbu sxemaning asosiy afzalliklari aniq funksional to'liqlik ta'minlangan ayrim iteratsiyadan boshlanib, foydalanuvchiga mahsulotni taqdim etish hisoblanadi, o'z navbatida quyidagi imkoniyatga ega bo'ladi:

- dasturiy mahsulotning birinchi versiyasi yuzaga kelguncha bo'lgan vaqtni qisqartirish;
- bozorda mahsulotning quyidagi versiyasi tez oldinga siljishini ta'minlagan holda ko'plab foydalanuvchilarni qiziqtirish;
- mahsulotdan foydalanish amaliyoti yuzaga kelishi hisobiga xususiyatlarni shakllantirish va aniqlashtirishni tekshirish;
- ishlab chiqish vaqtida tizimning ma'naviy eskirish ehtimolini kamaytirish.

Spiral sxemasidan foydalanishning asosiy muammosi quyidagi bosqichga o'tish vaqtini aniqlash hisoblanadi. Uni hal etish uchun, odatda, ekspert baholashga asoslangan har bir bosqichning o'tish muddati cheklanadi.

**CASE texnologiyalardan foydalanilganda dasturiy ta'minotning hayotiy sikl o'zgarishi.** *CASE texnologiyalar* o'zaro bog'langan avtomatlashtirish vositalari kompleksi bilan ta'minlanadigan tarkibiy kabi obyekt yondashuviga asoslangan murakkab dasturiy tizimning tahlil qilish, loyihalashtirish, ishlab chiqish va kuzatish metodologiyasi majmuidan iborat. Istalgan CASE texnologiyalar asosida metodologiyalar/usul/notatsiya/vosita paradigma yotadi.




**1.12-rasm.** Dasturiy ta'minotni ishlab chiqishning spiral yoki iteratsion sxemasi.

Metodologiya ayrim yondashuv bazasida tuziladi va ishlar qadamini, ularning ketma-ketligini, shuningdek, taqsimlash va belgilash usullarining qoidalarini bajarish qadami u yoki bu maqsadga erishish usulini belgilaydi.

*Notatsiya* deb, modellarning ayrim sinfini bayon etish uchun foydalaniladigan belgilashlar tizimiga aytiladi. Notatsiya — grafik (grafiklar, diagrammalar, jadvallar, sxemalar va boshqalar) va matnli (forma va tabiiy tillarda modellarni bayon qilish) bo'ladi. Notatsiya CASE texnologiyalarida loyihalashtiriladigan tizim tuzilmasi, ma'lumotlar elementlari, ishlov berish bosqichlarini va boshqalarni bayon qilish uchun foydalaniladi.

*Vositalar* — grafik loyihani yaratish va tahrir qilish vositasi, abstraksiyaning iyerarxiya darajasi ko'rinishida loyihani tashkil qilish, shuningdek turli darajalar komponentlariga muvofiqligini tekshirish usullarni ta'minlash uchun instrementariya hisoblanadi. Ular quyidagilarga bo'linadi:

• CASE – xususiyatlar, talablar tahlili, loyihalashtirish, interfeyslar tuzilmasi, tahrir qilishning vositasi (CASE–I ning birinchi avlodi);

• CASE – dastlabki matnlar generasiyasining va dasturiy ta'minotni ishlab chiqishning to'liq hayotiy siklini tà'minlashning integratsiya qilishni qamrab olishni amalga oshirishning vositasi (CASE–II ikkinchi avlodi).

CASE–I asosan grafik modellar, xususiyatlarni loyihalashtirish, ekranli tahrirlar va ma'lumotlar lug'atlarini ta'minlash vositalarini o'z ichiga oladi. CASE–II – tizimli axborotni hamda loyihalashtirish jarayonini boshqarish bo'yicha axborotni nazorat qilish, tahlil qilish va bog'lash, tizim protokollarini va modellarini qurish, testlash, verifikatsiyalash va generatsiyalanadigan dasturlarni tahlil qilish, ta'minlash imkoniyatlari bilan ajralib turadi.

Ko'p mehnat talab qiladigan operatsiyalarni avtomatlashtirib, zamonaviy CASE vosita dasturchilarning mehnat unumdorligini oshiradi va tuzilayotgan dasturiy ta'minotni yaxshilaydi. Ular quyidagilarni amalga oshiradi:

• loyiha xususiyatini moslashuvining avtomatlashtirilgan nazoratini ta'minlaydi;

• tizim prototipini yaratish vaqtini kamaytiradi;

• loyihalashtirish va ishlab chiqish jarayonini tezlashtiradi;

• loyiha hujjatlarining shakllanishini zamonaviy standartlarga muvofiq hayotiy siklning barcha bosqichlari uchun avtomatlashtiradi;

• ishlab chiqishning turli platformalari uchun dasturlar kodlarini qisman generatsiya qiladi;

• tizim komponentlaridan takroran foydalanish bo'yicha texnologiyalar bilan ta'minlaydi;

• mavjud bo'lgan dastlabki kodlar bo'yicha loyiha hujjatlarini tiklash imkonini ta'minlaydi. CASE texnologiyalarning yuzaga kelishi dasturiy ta'minotning barcha hayotiy sikl bosqichlarini o'zgartiradi, bunda eng katta o'zgarish ishlab chiqilayotgan das-

turiy ta'minotni qat'iy va amaliy bayonini ko'zlaydigan tahlil qilish va loyihalashtirishga tegishlidir.

1.1-jadvalda dasturiy ta'minotni ishlab chiqish jarayonining qaysi sifatli o'zgarishi CASE vositadan foydalanishga o'tishda amalga oshirilishi ko'rsatilgan.

*1.1-jadval*

| An'anaviy ishlab chiqish | CASE vositadan foydalanib ishlab chiqish |
|---|---|
| Kodlash va testlash uchun asosiy kuchlar | Tahlil qilish va loyihalashtirish uchun asosiy kuchlar |
| «Qog'ozli» xususiyat | Iteratsion prototiplashtirish |
| Qo'lda kodlash | Kodlarni avtomat tarzda generatsiya qilish |
| Qo'lda hujjatlashtirish | Hujjatni avtomat tarzda generatsiya qilish |
| Kodlarni testlash | Loyihani avtomat tarzda nazorat |
| Kodlarni kuzatish | Loyihalashtirish xususiyatini kuzatish |

ASE vositadan foydalanish asosan hujjatlashtirish va nazorat qilish jarayonlarini avtomatlashtirish hisobiga murakkab dasturiy ta'minotni ishlab chiqish uchun ko'p mehnat xarajatlarini kamaytirish imkonini beradi.

CASE vositadan foydalanish hujjatlashtirish va nazorat jarayonlarni avtomatlashtirish hisobiga asosan murakkab dasturiy ta'minotni (1.2-jadval) ishlab chiqishga ketgan mehnat uchun xarajatlarni ancha kamaytirish imkoniga ega.

Biroq zamonaviy CASE vosita qimmat, undan foydalanish ishlab chiqaruvchilardan ancha yuqori malakani talab qiladi. Binobarin, ulardan murakkab loyihalarda foydalanish mumkin, bunda ishlab chiqilayotgan dasturiy ta'minot murakkab bo'lgani sayin, CASE texnologiyalardan foydalanishda foyda ko'proq bo'ladi. Hozirgi kunda amalda barcha sanoatda ishlab chiqari-

ladigan murakkab dasturiy ta'minot CASE vositadan foydalangan holda ishlab chiqiladi.

*1.2-jadval*

| Ishlab chiqish usuli | Ishlab chiqish bosqichining ko'p mehnat xarajatlari, % | | | |
|---|---|---|---|---|
| | Tahlil | Loyihalashtirish | Kodlash | Testlash |
| An'anaviy ishlab chiqish | 20 | 15 | 20 | 45 |
| Tarkibli yondashish | 30 | 30 | 15 | 25 |
| CASE texnologiyalar | 40 | 40 | 5 | 15 |

## 1.7. Dasturiy ta'minot ishlab chiqarilishini tezlashtirish

Dasturiy ta'minot hayotiy siklning spiral modelini va CASE texnologiyalarni ishlab chiqish dasturiy ta'minotni yaratish muddatlarini qisqartiradigan shartlarni ifodalash imkoniga ega bo'ldi.

Dasturiy ta'minotni loyihalashtirish, ishlab chiqish va kuzatishning zamonaviy texnologiyalari quyidagi talablarga javob berishi kerak:

- dasturiy ta'minotning to'liq hayotiy zanjirini ta'minlash;
- belgilangan sifat bilan va belgilangan vaqtda ishlab chiqish maqsadlariga kafolatlangan erishish;
- tarkibiy qismlarni keyinchalik integratsiya qilish va umumiy loyiha joriy qilinishini muvofiqlashtirishni cheklangan bajaruvchilar sonining (3–7 kishi) ishlab chiqilayotgan guruhlarning tizim qismi ko'rinishida, yirik loyihalarni bajarish imkoniyati;
- ishlash imkoniyatiga ega tizimni yaratishning minimal vaqti;
- loyiha konfiguratsiyasini boshqarish, loyiha versiyasini joriy etish va har bir versiyasi bo'yicha loyiha hujjatlarini avtomatik tarzda chiqarish imkoniyati;

• amalga oshirish vositalardan (MBBT, operatsion tizimlar, dasturlash tillari va tizimi) bajariladigan loyiha qarorlariga mustaqillik;

• jarayonlarni avtomatlashtirishni ta'minlaydigan, hayotiy zanjirning barcha bosqichlarida bajariladigan kelishilgan CASE vositalar kompleksini ta'minlash.

Ushbu talablarga RAD (Rapid Application Development – ilovalarni tez ishlab chiqish) texnologiyalari javob beradi. Ushbu texnologiya ishlab chiqilayotgan dasturiy ta'minotning birinchi versiyasini maksimal tez olish uchun mo'ljallangan. U quyidagi shartlar bajarilishini ko'zda tutadi:

• ishlab chiquvchilarning katta bo'lmagan guruhlari bilan (3–7 kishi) ishlab chiqishni joriy qilish, ushbu guruhlarning har biri loyihaning ayrim tizim qismlarini loyihalashtiradi va amalga oshiradi, loyihaning boshqariluvchanligini yaxshilash imkoniga ega bo'ladi;

• iteratsion yondashuvdan foydalanish ishlash qobiliyatiga ega prototipni olish vaqtini kamaytirishga ega bo'ladi;

• uch oydan ko'p bo'lmagan muddatga mo'ljallangan ishlab chiqilgan aniq sikl grafigining mavjudligi ish samaradorligini sezilarli darajada oshiradi.

Bunda ishlab chiqish jarayoni quyidagi bosqichlarga bo'linadi: foydalanuvchilar talablarini tahlil qilish va loyihalashtirish, loyihalashtirish, amalga oshirish, joriy etish.

*Talablarni tahlil qilish va loyihalashtirish bosqichida* loyiha masshtabini cheklaydigan eng ustuvor talablar shakllantiriladi.

*Loyihalashtirish* bosqichida CASE vositadan foydalangan hamda, tizim jarayonini batafsil bayon qilinadi, ma'lumotlardan foydalanishni cheklash talablarini va zarur hujjatlar tarkibini belgilaydi. Bunda eng murakkab jarayonlar uchun qisman prototip yaratiladi.

Ekran shakli va dialog ishlab chiqiladi. Jarayon tahlilining natijasiga ko'ra, funksional nuqtalar sonini belgilaydi va ishlab chiqishda qatnashadigan tizim qismi va mos ravishda komandalarning soni to'g'risidagi qarorni qabul qiladi.

RAD texnologiyalardagi *funksional nuqta* deganda, ishlab chiqilayotgan tizimning quyidagi funksional elementlaridan istalgani tushuniladi:

- ilovaning kirish elementi (kirish hujjati yoki ekranli shakl);
- ilovaning chiqish elementi (hisobot, hujjat yoki ekranli shakl);
- so'rov («savol/javob»);
- mantiqiy fayl (ilova ichida foydalaniladigan ma'lumotlar yozuvining jami);
- ilova interfeysi (boshqa ilovalarga uzatiladigan yoki boshqa ilovalardan olinadigan ma'lumotlar yozuvining jami).

Kodlari sezilarli darajada takrorlanadigan tizim uchun ekspert baholanishidan kelib chiqib, hisoblab chiqilgan normalar quyidagicha aniqlanadi:

- 1 mingdan kam funksional nuqtalar – 1 kishi;
- 1 dan 4 minggacha funksional nuqtalar – ishlab chiquvchilarning bir komandasi;
- 4 mingdan ko'p funksional nuqtalar – har bir 4 ming nuqtaga bitta komanda.

Ushbu normalarga muvofiq ishlab chiqilayotgan tizimni ma'lumotlar va funksiyalarga bo'sh bog'liq bo'lgan tizim qismlarga bo'linadi va turli qismlar o'rtasidagi interfeyslarni aniq belgilaydi. Bunda CASE vositalardan foydalanish loyiha to'g'risidagi axborotni bosqichdan bosqichga uzatishda ma'lumotlarning nazorat qilmaydigan buzilishlarining oldini olish imkoniga eng bo'ladi.

Keyinchalik ishlab chiqish o'z tizim qismlarini ishlab chiqilishini davom ettiradigan ishlab chiquvchi guruhlar bilan olib boriladi. Turli guruh ishlab chiqaruvchilarning harakatlari yaxshi muvofiqlashtirilgan bo'lishi kerak. Amalga oshirish bosqichida real tizimning iterativ tuzilishi bajariladi, bunda tuzilayotgan tizimga qo'yiladigan talablar bajarilishini nazorat qilish uchun foydalanuvchilar jalb qilinadi. Qismlar tizimga asta-sekin integratsiya qilinadi, bunda har bir qism ulanganda testlash jarayoni bajariladi. Ishlab chiqishning yakuniy bosqichida tizimda ishlab

chiqiladigan va ulanadigan tegishli ma'lumotlar bazasini yaratish zarurligi aniqlanadi. Apparat vositalariga qo'yiladigan talablarni shakllantiradi, unumdorlikni oshirish usullarini belgilaydi va loyiha bo'yicha hujjatlarni tayyorlash ishlarini tugatadi. Joriy etish bosqichida foydalanuvchilar o'qitiladi va yangi tizimga astasekin o'tish amalga oshiriladi, bunda yangi tizim to'liq tadbiq qilingunga qadar eski versiyada ishlash davom etadi.

RAD texnologiya aniq buyurtmachi uchun ishlab chiqilayotgan katta bo'lmagan loyihalar uchun o'zining afzalligini namoyon etadi. Bunday tizimlar loyihalashtirishning va qat'iy loyihalashtirishning yuqori darajalarini talab qilmaydi. Biroq ushbu texnologiya murakkab hisoblash dasturlari, operatsion tizimlar yoki real vaqt masshtabida murakkab obyektlarni boshqarish dasturlarining ya'ni, noyob kodning katta foizli dasturlarning tuzilishi uchun qo'llanilmaydi.

Ushbu texnologiyani insonlar xavfsizligi bilan bog'liq bo'lgan ilovalarni yaratishda qo'llab bo'lmaydi, masalan, samolyotlarni yoki atom elektrostansiyalarni boshqarishda, chunki RAD texnologiyaning dastlabki bir nechta versiyalari to'liq ish qobiliyatiga ega bo'lmaydi, ushbu holat uchun bu istisno qilinadi.

## 1.8. Dasturiy ta'minotni yaratish jarayonining sifatini baholash

Yuqorida ko'rsatib o'tilganlardan ko'rinib turibdiki, dasturiy ta'minot bozorida joriy davr dasturiy ta'minotning donalab ishlab chiqishdan sanoatda yaratishga o'tishi bilan xarakterlanadi, bu o'z navbatida, ularni ishlab chiqish jarayonlarini takomillashtirilishini talab qiladi. Hozirgi vaqtda ishlab chiquvchi tashkilot ta'minlaydigan ushbu jarayonlarning sifatini baholash bilan bog'liq bir nechta standartlar mavjud.

Ularga quyidagilar kiradi:

- ISO 9000 seriyali xalqaro standartlar (ISO 9000 – ISO 9004);
- SMM – Capability Maturity Model – dasturiy ta'minotni yaratish jarayonlarining yetuklik (takomillashtirilgan) modeli,

SEI (Software Engineering Institute – Karnegi – Mellon universiteti qoshidagi dasturlash instituti) ilova qilinadi;

• ISO/IEC 15504 xalqaro standartning ishchi versiyasi: Information Technology – Soft ware Process Assessment; Ushbu versiya SPICE (Software Process Improvement and Capability determination – dasturiy ta'minotni yaratish jarayonining imkoniyatlarini aniqlash va uni yaxshilash) nomi bilan tanish.

**ISO 9000 standart seriyasi.** ISO 9000 seriyada jarayonni tashkil qilishning ayrim minimal darajasiga erishish uchun zarur shartlar shakllangan, lekin jarayonni keyinchalik takomillashtirish bo'yicha tavsiyalar bermaydi.

**SMM.** SMM ishlab chiquvchi – tashkilotning yetukligini baholash mezonlarini va amaldagi jarayonlarni yaxshilash retseptlarini o'zida jamlaydi.

SMM ishlab chiquvchi – tashkilotlarning beshta yetuklik darajalarini belgilaydi, bunda har bir keyingi daraja barcha oldingi asosiy tavsiflarni o'z ichiga oladi.

**1. Boshlang'ich daraja** (initial level) keyingi darajalar bilan solishtirish uchun asos sifatida standartda bayon qilingan. Tashkil qilishning bunday darajasi korxonada sifatli dasturiy ta'minotni yaratish uchun barqaror shartlar mavjud emas. Istalgan loyiha natijalari menejerning shaxsiy malakasi va dasturchining tajribasiga to'liq holda bog'liq, bir loyihadagi muvaffaqiyat xuddi shu menejer va dasturchini belgilagan holda yana takrorlanishi mumkin. Agar shu menejer va dasturchilar korxonadan ketsa, ishlab chiqilayotgan dasturiy mahsulot sifati keskin kamayib ketadi. Bunday vaziyatlarda ishlab chiqish jarayoni kodni yozishga va uni minimal testlashga keltiriladi.

**2. Takrorlanadigan daraja** (repeatable level) – korxonada loyihalarni boshqarish texnologiyalari joriy qilingan. Bunda loyihalashtirish va boshqarish orttirilgan tajribaga asoslanadi, ishlab chiqilayotgan dasturiy ta'minot standarti (bunda ushbu standartlarga rioya qilish ta'minlanadi) va sifatni ta'minlashning maxsus guruhi mavjud. Zarur bo'lganda tashkilot subpudratchi bi-

lan o‘zaro hamkorlik qilishi mumkin. Keyin sharoitlarda jarayon boshlang‘ich darajaga tushadigan tendensiyaga ega bo‘ladi.

**3. Aniq daraja** (defined level) – dasturiy ta’minotni yaratish va kuzatishning standart jarayoni to‘liq hujjatlashtirilgan (shu jumladan dasturiy ta’minotni ishlab chiqish va loyihalar bilan boshqarish) standartlashtirish jarayonida eng samarali amaliyot va texnologiyalarga o‘tiladi. Bunday standartni tashkilotda yaratish va ta’minlab turish uchun maxsus guruh tuzilishi kerak. Ushbu darajaga erishish uchun majburiy shartlardan biri bo‘lib, korxonada malakani doimiy oshirish va xodimlarni o‘qitish dasturining mavjudligi hisoblanadi. Ushbu darajadan boshlab, tashkilot aniq ishlab chiquvchilarning sifatiga bog‘liq bo‘lmaydi va jarayon noqulay vaziyatlarda past darajaga tushish tendensiyalari bo‘lmaydi.

**4. Boshqariladigan daraja** (managed level) – tashkilotda dasturiy ta’minotga hamda, butun jarayonga sifatning miqdoriy ko‘rsatkichlari belgilanadi. Shunday qilib, yuqori bosqichga yetishish uchun korxonaning xodimlarini o‘qitish va malaka oshirish uchun dastur bo‘lishi kerak. Bunda, jarayonni ishlab chiqishda ma’noga ega variatsiyani yaxshi o‘zlashtirilgan tasodifiy variatsiyadan (shovqin) farqlash mumkin.

**5. Optimallashtiruvchi daraja** (optimizing level) – yaxshilash bo‘yicha tadbirlar amaldagi (mavjud) jarayonlardan tashqari, yangi texnologiyalarni samarali kiritilishini baholash uchun ham qo‘llanishi mumkinligi bilan xarakterlanadi. Ushbu darajadagi butun tashkilotning asosiy vazifasi bo‘lib, amaldagi jarayonlarning doimiy takomillashtirish hisoblanadi. Bunda jarayonlarni yaxshilash xatolar va nuqsonlarning oldini olishga yordam berishi kerak. Bundan tashqari, dasturiy ta’minotni ishlab chiqish narxining kamayishi bo‘yicha ishlar olib borilishi kerak, masalan, komponentlarni yaratish va qaytadan foydalanish yordamida.

Barcha asosiy sohadagi muvofiqlikning sertifikasiya bahosi 10 balli shkala bo‘yicha o‘tkaziladi. Ushbu asosiy sohaning muvaffaqiyatli malakasini oshirish uchun kamida 6 ball to‘plash

zarur. Asosiy sohani baholash quyidagi ko'rsatkichlar bo'yicha amalga oshiriladi:

- ushbu sohaga rahbarlarning qiziqishi, masalan, qaralayotgan asosiy sohani (dasturlash nuqtayi nazaridan) amaliyotda qo'llash rejalashtirilganmi, mazkur sohaning zarurliligini rahbariyat tushunadimi va hokazo;
- ushbu soha tashkilotlarda qanchalik keng qo'llanishga mos keladi, masalan, 4 balli baholashda fragmentlar qo'llashga mos keladimi va hokazo;
- ushbu sohaning amaliyotda muvaffaqiyatli foydalanishi, masalan, 0 ball har qanday samara mavjud emasligiga mos keladi, 8 ball butun tashkilotda muntazam va o'lchangan ijobiy natija mavjud bo'lganda qo'yiladi.

Asosan, faqat bitta jarayonni yoki tashkilot bo'linmasini sertifikatlash mumkin. Masalan, IBM kompaniyasining dasturiy ta'minotini ishlab chiqish bo'linmasi 5 darajada sertifikatlandi. Dunyoda 5 darajaga ega kompaniyalar uncha ko'p emas. Boshqa tomondan 3 va 4 daraja bilan sertifikatlangan kompaniyalar bir necha mingtani tashkil qiladi, ya'ni yetuklikning optimallashgan darajasi va oldingi darajalar o'rtasida juda katta farq mavjud. Biroq, juda katta farq yana boshlang'ich darajadagi tashkilotlar va ancha rivojlangan tashkilot o'trasida kuzatilmoqda. Kuzatishlar, barcha ishlab chiqaruvchi kompaniyalar 70% dan oshig'i CMM darajada turibdi.

**SPICE.** SPICE standarti oldingi standartlardan ko'p jihatlarni o'zlashtirgan. ICO 9001 va CMM asosan SPICE CMM ni o'z ichiga olgan. Tashkilotning asosiy vazifasi bo'lib, dasturiy ta'minotni ishlab chiqish jarayonini doimo yaxshilash hisoblanadi. Bundan tashqari, SPICE standartida turli imkoniyatlar darajasi bo'lgan sxemalardan foydalaniladi (SPICE da 6 ta turli darajalar aniqlangan), lekin, ushbu darajalar butun tashkilotda emas, balki, alohida jarayonda ham qo'llanilishi mumkin.

Standartning asosi bo'lib, jarayonni baholash hisoblanadi. Ushbu baholash standartda model tarzda bayon qilinadi. Ush-

bu tashkilotdagi dasturiy ta'minotni ishlab chiqish jarayonini taqqoslash yo'li bilan bajariladi. Ushbu bosqichda olingan natijalar tahlili jarayonning kuchli, shuningdek, ushbu jarayonga taalluqli bo'lgan ichki xavflarni va kuchsiz tomonlarini aniqlash imkonini beradi. Bu o'z navbatida, jarayonlarning samaradorligini baholashga, sifatning yomonlashishi va vaqt yoki bahosi bo'yicha xarajatlarga bog'liq sabablarni aniqlashga yordam beradi.

Keyin jarayonning imkoniyatlari, ya'ni uni yaxshilash zarurligi to'g'risidagi tushuncha yuzaga kelishi mumkin. Bu paytda, jarayonni takomillashtirish maqsadi aniq shakllangan va qo'yilgan vazifalarni texnik amalga oshirish qolmoqda. Bundan tashqari xulosa qilib shuni aytish mumkinki, dasturiy ta'minot hayotiy siklining jarayonini takomillashtirish zarur. Biroq, ishlab chiqishning «juda yetuk» jarayoni tuzilishi sifatli dasturiy ta'minot yaratilishini ta'minlaydi. Bu turlicha bo'lgan jarayonlar.

Forma modellari va usullardan foydalanish ishlab chiqilayotgan dasturiy ta'minot uchun tushunarli, zid kelmaydigan spetsifikatsiyalarni yaratish imkonini beradi. Bunday usullarni joriy qilish, uning ancha qimmatli va ko'p mehnat talab qilganda ham zarur, ularni qo'llash imkoniyati juda cheklangan.

Asosiy muammo bo'lib, ishlab chiqish jarayonining takomillashuvi hal etilmagani, ishlab chiqilayotgan dasturiy ta'minotning murakkabligi hisoblanadi. Dasturiy ta'minotni yaratish bilan shug'ullanuvchilarning, masalan, dasturiy ta'minot loyihachilarining va bevosita dasturchilarning malakasini oshirish talablari asosiy vazifa bo'lib hisoblanadi.

**Nazorat savollari:**

1. Dasturlash texnologiyalari tushunchasi?
2. Yondashuv va usulning farqi?
3. Dasturlash texnologiyasi rivojlanishining bosqichlarini ayting. Bu davrlar nimalar bilan farqlanadi. Informatikadagi asosiy yondashuvlar qanday o'zgarganligini so'zlab bering.

4. Murakkab ierarxik tizim ta'rifini bering. Bunday tizimlarni yaratishda qanday yondashuvlar ishlatiladi. Bu tizimlar qanday tasniflarga asoslangan?

5. Dasturiy ta'minot hayotiy sikli deganda nima tushuniladi. Bu tushuncha qaysi asosiy jarayonlarga asoslangan?

6. Dasturiy ta'minotni yaratishning asosiy bosqichlarini sanab o'ting. Bu bosqichlarda qaysi asosiy masalalar yechiladi.

7. Hayotiy siklning asosiy modellarini ayting. Yangi modellarning paydo bo'lishi nimalar bilan bog'langan?

8. Qaysi texnologiyalar CASE texnologiya deb ataladi. Nima uchun?

9. Ixtiyoriy CASE texnologiyasini tashkil etuvchilarni ayting.

10. RID texnologiyasining ma'nosini tushuntiring. Bu texnologiya yordamida dasturiy tizimlarni yaratish mumkin emas.

11. DT sifati va uning modellari deganda nima tushuniladi?

12. Zamonaviy DT rivoji bosqichi nima bilan tavsiflanadi?

13. Dasturning axborot muhiti deganda nimani tushunasiz?

14. Dasturiy vosita (DV) nima?

15. DV dagi xato nimani bildiradi?

16. DV ning ishonchliligi qanday aniqlanadi?

# 2. DASTURIY MAHSULOTLAR TEXNOLOGIKLIGINI TA'MINLASH USULLARI

Dasturiy ta'minotni ishlab chiqish ishlab chiqilayotgan dasturlarning texnologik tavsifnomalari alohida salmoq egallaydi. Zarur texnologik xususiyatlarni ta'minlash uchun maxsus texnologik usullar qo'llaniladi va dasturiy ta'minot yaratishning butun avvalgi tajribasida shakllantirilgan muayyan metodikalar bo'yicha ish ko'riladi. XX asrning 60-yillaridayoq «dasturlashga tuzilmaviy yondashuv» degan umumiy nom ostida shakllantirilgan dekompozitsiya qoidalari, loyihalash, dasturlash va sifat nazorati usullari xuddi shunday namunalar hamda metodikalarga majburdir. Uning asosiga quyidagi asosiy konsepsiyalar qo'yilgan:

- qo'yiluvchi ishlab chiqish;
- modulni dasturlash;
- tuzilmaviy dasturlash;
- oralama tuzilmaviy nazorat.

## 2.1. Dasturiy ta'minlash texnologikligi tushunchasi

*Texnologiklik* deganda dasturiy mahsulot loyihasining sifati tushuniladi hamda uning amalga oshirilishining mehnat va moddiy xarajatlari, keyingi modifikatsiyalari shunga bog'liq. Yaxshi loyiha nisbatan tez va oson kodlanadi, testlanadi, sozlanadi hamda modifikatsiya qilinadi.

Dasturiy ta'minot ishlab chiqaruvchilar bir necha avlodining tajribasidan ma'lumki, *dasturiy ta'minlash texnologikligi uning modellari ishlanganligi, modullar mustahkamligi pog'onasi, dasturlash uslubi va kodlardan takroriy foydalanishning darajasi bilan belgilanadi.*

Ishlab chiqilayotgan dasturiy ta'minot modeli qanchalik yaxshi ishlagan bo'lsa, kiruvchi, oraliq va chiquvchi axborotni saqlovchi ma'lumotlarning tag-vazifalari, tuzilmalari shunchalik aniq, ularni amalga oshirish va loyihalash shunchalik yengil hamda tuzatish uchun dasturni jiddiy o'zgartirish talab qiluvchi xatolar ehtimoli ham bo'ladi.

Modullar mustahkamligi qanchalik yuqori bo'lsa, ularni tushunish, amalga oshirish, modifikatsiya qilish, shuningdek ulardagi xatolarni topish va tuzatish shunchalik yengil bo'ladi.

Dasturlarni rasmiylashtirish uslubi va ularning «tuzilmaviyligi» tushuniladigan dasturlash uslubi dasturiy kod o'qiluvchanligiga hamda dasturlash xatolari miqdoriga ham jiddiy ta'sir ko'rsatadi. XX asr 60-yillaridagi inqiroz yuzaga kelishiga dastur chigal ip kalavasini yoki spagetti taomini eslatuvchi dasturlash uslubi ham, «tuzilmaviy» uslubini qo'llab-quvvatlashning til konstruksiyalari yo'qligi ham sabab bo'lgandi.

Kodlardan takroriy foydalanish darajasini orttirish ilgari ishlab chiqilgan tagdasturlar yoki sinflar kutubxonasidan foydalanishni ham, joriy ishlab chiqishdagi kodlar unifikatsiyasini ham nazarda tutadi. Bunda ma'lum mezon uchun vaziyat avvalgi holatlardagidek bir xil emas: agar kodlardan takroriy foydalanish darajasi sun'iy ravishda (masalan, «superuniversal» protseduralar ishlab chiqish yo'li bilan) orttirilsa, u holda loyihaning texnologikligi jiddiy ravishda pasayishi mumkin.

Ta'birdan kelib chiqadiki, agar ko'p yillik jadal foydalanishga mo'ljallangan dasturiy mahsulot ishlab chiqilayotgan yoki uning sifatiga yuksak talablarini ta'minlash zarur bo'lsa, loyihaning yuqori texnologikligi o'ta muhimdir.

## 2.2. Modullar va ularning xususiyatlari

Yetarlicha murakkab dasturiy ta'minotni loyihalashtirishda uning umumiy tuzilmasi belgilangandan so'ng, loyihachi fikriga ko'ra, keyingi dekompozitsiyaga muhtoj bo'lmagan elementlar olingunga qadar tanlangan yondashuvga muvofiq holda komponentlar dekompozitsiyasi bajariladi.

Avval eslatib o'tilganidek, hozirgi paytda ishlab chiqilayotgan dasturiy ta'minot dekompozitsiyasining tegishli yondashuv bilan bog'liq ikki usuli qo'llaniladi:

- protsedurali (yoki tuzulmaviy-yondashuv nomi bo'yicha);
- obyektli.

**Eslatma.** Ko'rsatilgan dekompozitsiya usullaridan tashqari dasturlash nazariyasida dekompozitsiyasining boshqa usullari ham belgilanadi: mahsulot faktlari va qoidalariga – mantiqiy dekompozitsiya va hokazo. Mazkur dekompozitsiya usullari sun'iy intellekt tillarida qo'llaniladi, shu boisdan ular ushbu darslikda ko'rib chiqilmaydi.

Qaror qabuli bilan bog'liq funksiyalar yuqori darajadagi tagdasturlar, bevosita qayta ishlash esa quyi darajadagi tagdasturlar orqali ijobatlanadigan tagdasturlar iyerarxiyasi protsedurali dekompozitsiya natijasi hisoblanadi. Bu dasturlashga tuzilmaviy yondashuvning boshqa tavsiyalari bilan birgalikda shakllantirilgan vertikal boshqaruv tamoyili bilan muvofiqlashadi. U, shuningdek, har qanday tagdastur o'zining chaqirgan tagdastur boshqaruvini qaytarishini talab qilgan holda boshqaruv berilishining ehtimolli variantlarini ham chegaralaydi.

Obyektli dekompozitsiya natijasi obyektlar uyg'unligi bo'lib, tegishli maydonlar bilan ishlovchi ma'lumotlar va usullar uyg'unligini ifodalagan holda keyinroq ayrim maxsus ishlab chiqiluvchi namunalar (sinflar) sifatida aniqlanadi.

Shunday qilib, har qanday dekompozitsiya usulida amalga oshirish jarayonida modullarga aylantiriladigan kichik dasturlarning tegishli ma'lumotlari bilan bog'liq majmua vujudga keltiriladi.

**Modullar.** Avtonom kompilyatsiyalanuvchi dasturiy birlik modul deyiladi. «Modul» atamasi an'anaviy ravishda ikki ma'noda qo'llaniladi. Dastlab, dasturlar o'lchami nisbatan katta bo'lmagan va barcha tagdasturlar alohida kompilyatsiyalangan paytda kichik dastur, ya'ni murojaat nom bo'yicha bajariladigan dastur fragmentlari bog'liqligining davomiyligi modul deb tushuniladi. Vaqt o'tib, dasturlar o'lchami ancha ortdi va resurslar: konstantlar, o'zgaruvchilar, namunalar, sinflar va tagdasturlar bayonlari kutubxonasini yaratish imkoniyati paydo bo'lgach, «modul» atamasi dasturiy resurslarning avtonom kompilyatsiyalanuvchi majmui ma'nosida ham qo'llana boshlandi.

Modul ma'lumotlarni xotiraning umumiy sohalari yoki parametrlari orqali olishi mumkin.

Dastavval modullarga (hali tagdastur sifatida tushunilayotgan) quyidagi talablar qo'yilardi:

- alohida kompilyatsiya;
- bitta kirish va chiqish nuqtasi;
- vertikal boshqaruv tamoyiliga mosligi;
- boshqa modullarni chaqiruv imkoniyati;
- katta bo'lmagan o'lcham (50–60 til operatorigacha);
- chaqiruv tarixiga bog'liq bo'lmaslik;
- bitta funksiyani bajarish.

Bitta kirish nuqtasi, bitta chiqish nuqtasi, chaqiruv tarixidan mustaqillik va vertikal boshqaruv prinsipiga muvofiqlik talablarining boisi shunda ediki, o'sha paytlarda operativ xotira hajmiga bo'lgan jiddiy cheklovlar tufayli dasturchilar kodlarning maksimal takrorlanuvchanligi ehtimoli mavjud dasturlar ishlab chiqishga majbur edilar. Natijada bir necha kirish va chiqish nuqtalari mavjud tagdasturlar nafaqat oddiy hol, balki dasturlashning yuqori toifasi sanalardi. Oqibati esa shu ediki, dasturlarni nafaqat modifikatsiyalash, balki tushunish, ba'zan esa shunchaki to'liq sozlash ham juda murakkab bo'lardi.

Vaqt o'tib, tuzilmaviy yondashuvning asosiy talablarini dasturlash tillari quvvatlaydigan va modul deganda alohida kompilyatsiyalanuvchi resurslar kutubxonasi tushuniladigan bo'lgach, modullar mustaqilligi asosiy talabga aylandi.

Amaliyot ko'rsatdiki, modullar mustaqilligi darajasi qanchalik yuqori bo'lsa, shunchalik ravishda:

- alohida modulda va butun dasturda tartibni bilish hamda muvofiq ravishda uni testlash, sozlash va modifikatsiyalash yengil bo'ladi;
- eski xatolarni tuzatishda yoki dasturlarga o'zgarishlar kiritishda yangi xatolar paydo bo'lishining ehtimoli, ya'ni «to'liqligi» effekt paydo bo'lishi ehtimoli kamayadi;

• dasturchilar guruhi tomonidan dasturiy ta'minot ishlab chiqilishini tashkillashtirish osonlashadi va uni kuzatish yengillashadi.

Shunday qilib, modullar qaramligini kamaytirish loyiha texnologikligini yaxshilaydi. Modullar (dasturlar, kutubxonalar) mustaqilligi darajasini ikki mezon bo'yicha – ulashuv 2 va aloqadorlik bo'yicha baholanadi.

**Modullarni ulash.** Ulash bu modullarni o'zaro bog'liqligining o'lchovi bo'lib, modullar bir-biridan qanchalik yaxshi ajratilganligini belgilaydi. Modullar, agar ularning har biri boshqasi haqida hech qanday axborotga ega bo'lmasa, mustaqildir.

Modul boshqa modullar haqida qancha ko'p axborotni saqlasa, u shunchalik ko'p ular bilan ulashgan bo'ladi.

Modullar ulashuvining beshta tipi farqlanadi:

- ma'lumotlar bo'yicha;
- namuna bo'yicha;
- boshqaruv bo'yicha;
- ma'lumotlarning umumiy sohasi bo'yicha;
- borlig'i bo'yicha.

Ma'lumotlar bo'yicha ulashuv modullarning skalyar ifodalarda taqdim etilgan ma'lumotlar bilan almashinishlarini nazarda tutadi. Berilayotgan parametrlarning ko'p bo'lmagan miqdorida ushbu tip dasturiy ta'minotning eng yaxshi texnologik tavsiflarini ta'minlaydi.

Masalan, Max funksiyasi skalyar tipdagi parametrlar orqali ma'lumotlar bo'yicha ulashuvni nazarda tutadi:

```
Function Max (a,b: integer): integer;
begin
 if a>b then Max: =a
      else Max: = b;
end;
```

Namuna bo'yicha ulashuv modullarning tuzilmalarga birlashtirilgan ma'lumotlar bilan almashinishlarini nazarda tutadi.

Mazkur tip ham yondosh bo'lmagan tavsiflarni ta'minlaydi, biroq ular avvalgi tipdagidan ko'ra yomonroq, zero muayyan berilayotgan ma'lumotlar tuzilmalarga «yashirilgan», shu boisdan modullar o'rtasidagi aloqaning shaffofligi» kamayadi. Bundan tashqari, berilayotgan ma'lumotlar tuzilmasini o'zgartirishda undan foydalanuvchi barcha modullarni modifikatsiyalash zarur.

Jumladan, quyida bayon etilgan MaxEl funksiyasi namuna bo'yicha ulashuvni nazarda tutadi (a parametr – ochiq massiv):

```
Function MaxEl (a: array of integer): integer;
    Var i: world;
    begin
 Max El:=a[0]
 for i: 1 to High(a) do
 if a[i] > MaxEl then MaxEl:=a[i];
    end;
```

Boshqaruv bo'yicha ulashuvda bir modul boshqasiga modulning ichki mantig'ini boshqarish uchun belgilangan qandaydir axborot obyekt (bayroq) yuboradi. Bunday sozlashlar, shuningdek modullar o'zaro ta'sirining ko'rgazmaliligini pasaytiradi va shu boisdan avvalgi aloqalar tiplariga qiyosan ishlab chiqilayotgan dasturiy ta'minot texnologikligining yanada yomon tavsiflarini ta'minlaydi.

Masalan, MinMax funksiyasi boshqaruv bo'yicha ulashuvni nazarda tutadi, zero flag parametri ifodasi dastur mantig'iga ta'sir ko'rsatadi: agar MinMax funksiyasi true ga teng flag parametri ifodasini olsa, ikkitadan maksimal ifodani qaytaradi, agar false ni olsa, minimal ifodani qaytaradi:

```
Function MinMax (a,b: integer; flag; boolean): integer;
 begin
 if (a>b) and (flag) then MinMax:=a
                     else MinMax: = b;
    end;
```

Ma'lumotlarning umumiy sohasi bo'yicha ulashuv modullarni ulashda ma'lumotlarning umumiy sohasi bilan ishlashlarini nazarda tutadi. Ulashning ushbu tipidan foydalanish mumkin emas, zero:

• mazkur ulashuv tipidan foydalanuvchi dasturlar dasturiy ta'minoti kuzatishda tushunish uchun o'ta murakkab;

• umumiy ma'lumotlarning o'zgarishini keltirib chiqaradigan bitta modul xatosi boshqa modulni bajarishda paydo bo'lishi mumkin, bu esa xatolarni bartaraf etishni jiddiy darajada murakkablashtiradi;

• umumiy sohadagi ma'lumotlarga tayanishda modullarning muayyan nomlaridan foydalaniladi, bu esa ishlab chiqilayotgan dasturiy ta'minot moslanuvchanligini kamaytiradi.

Masalan, A global massivdan foydalanuvchi Max A funksiyasi asosiy dastur bilan umumiy soha bo'yicha ulashgan:

```
Function MaxA: integer;
Var i: world;
begin
    Max A: =a[Low(a)];
    for i: = Low (a)+1 to High (a) do
 if a [i] > Max A then MaxA: = a [i];
end;
```

Nazarda tutish lozimki, harakati chaqiruvlar tarixiga bog'liq «xotirali tagdasturlar» umumiy soha bo'yicha ulashuvdan foydalanadi, bu esa umumiy holatda ularning ishini oldindan bilib bo'lmaydigan qilib qo'yadi. C hamda C++ statik o'zgaruvchilar aynan shu variantdan foydalanishadi.

Mohiyatan ulashuv holatida bir modul boshqasining ichki komponentlariga murojaatga ega bo'ladi (boshqaruvni ichkariga beradi, ichki ma'lumotlarni, kodlarni o'zini o'qiydi yoki o'zgartiradi), bu esa blokli-iyerarxik yondashuvga tamomila ziddir. Ushbu holatda alohida modul blok («qora quti») bo'la olmaydi: uning borlig'i boshqa modulni ishlab chiqish jarayonida hisobga olinishi shart.

Protsedurali dasturlashning zamonaviy universal tillari, masalan Pascal, mazkur tipdagi ulashuvni oshkora tarzda quvvatlamaydi, biroq quyi darajadagi tillar, masalan Assembler uchun mazkur tipdagi ulashuv mumkinligicha qoladi.

2.1-jadvalda ekspertlik baholari bo'yicha ulashuv turli tiplarning tavsiflari berilgan. Ulashuvning dastlabki uch tipiga yo'l qo'yiladi, chunki qolganlaridan foydalanish texnologik dasturlarning keskin yomonlashuviga olib keladi.

*2.1-jadval*

**Tiplar tavsiflari**

| Ulashuv tipi | Ulashuv ball | Boshqa modullar xatolariga barqarorlik | Ko'rgazmalilik (tushunarlilik) | O'zgarish imkoniyati | Takroriy foydalanish ehtimoli |
|---|---|---|---|---|---|
| Ma'lumotlar bo'yicha | 1 | yaxshi* | yaxshi | yaxshi | katta |
| Namuna bo'yicha | 3 | o'rtacha | yomon | yomon | yomon |
| Boshqaruv bo'yicha | 4 | o'rtacha | yomon | yomon | kichik |
| Umumiy soha bo'yicha | 6 | yomon | yomon | o'rtacha | kichik |
| Borliq bo'yicha | 10 | yomon | yomon | yomon | kichik |

Qoidaga ko'ra, modullar o'zaro bir necha usullar orqali ulashadi. Buni hisobga olgan holda dasturiy ta'minot sifatini yomon tavsiflarga ega ulashuv bilan belgilash qabul qilingan. Xususan, ma'lumotlar bo'yicha ulashuvdan va boshqaruv bo'yicha ulashuvdan foydalanilgan taqdirda boshqaruv bo'yicha ulashuv belgilovchi hisoblanadi.

Ayrim hollarda modullar ulashuvini shart bo'lmagan aloqalarni olib tashlagan va zarur aloqalarni tuzilmalashtirgan holda kichraytirish mumkin. Obyektga mo'ljallangan dasturlashga misol

sifatida, unda bitta miqdordagi parametrlar o'rniga mazkur usul obyekt maydonlari joylashgan soha (tuzilma) manzilini nooshkora, qo'shish parametrlarni oshkora oladi. Natijada modullar namuna bo'yicha ulashgan bo'lib qoladi.

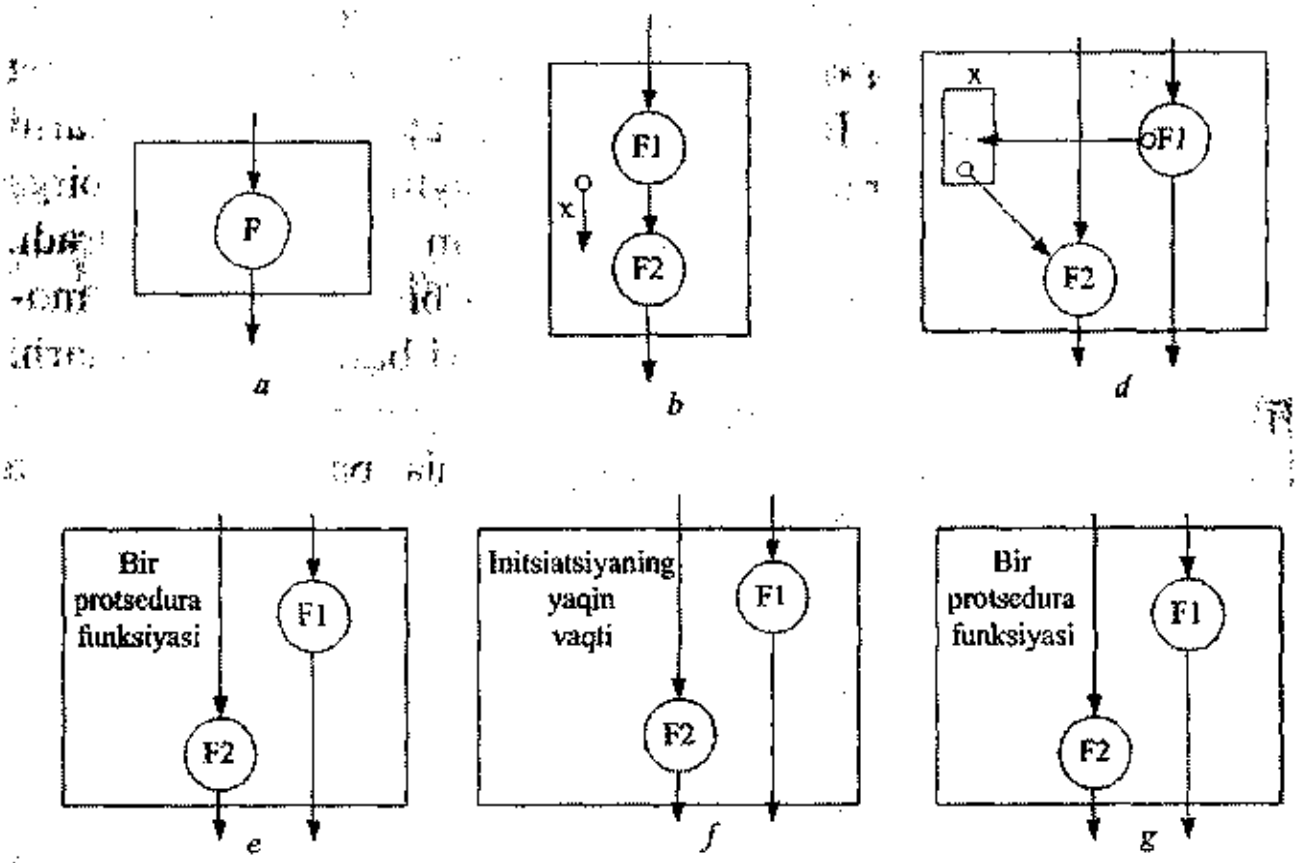
**Modullar aloqadorligi.** *Aloqadorlik* – bir modul ichidagi funksional va axborot obyektlar birikuvi mustahkamligining birligi. Agar ulashuv modullar bo'linmasi sifatini tavsiflasa, aloqadorlik bir modul amalga oshiradigan elementlar o'zaro aloqasining darajasini tavsiflaydi. Bitta modul kuchli aloqador elementlarni joylashtirish modullararo aloqalarni kichraytiradi, shu bilan birga ularning o'zaro ta'sirini tushunishni ham murakkablashtiradi. Shuningdek, zaif bog'langan elementlarni birlashtirish ham modullar texnologikligini kamaytiradi, chunki bunday elementlarni fikran boshqarish murakkab.

Aloqadorlikning quyidagi xillari (daraja pasayishi tartibiga ko'ra) farqlaniladi:

- funksional;
- ketma-ket;
- axborot (kommunikativ);
- protsedurali;
- muvaqqat;
- mantiqiy;
- tasodifiy.

*Funksional aloqadorlikda* modulning barcha obyektlari bitta funksiyani bajarish uchun belgilanadi (2.1-rasm, a): bitta funksiyani bajarish uchun birlashtirilgan operatsiyalarni yoki funksiyaga aloqador ma'lumotlar bo'ladi. Elementlari funksional aloqador modul aniq belgilangan maqsadga ega, uning chiqaruvida bitta vazifa, masalan massiv minimal elementini izlash tagdasturi bajariladi. Bunday modul maksimal aloqadorlikka ega bo'ladi, uning yaxshi texnologik sifatlari – testlash, modifikatsiyalash va kuzatish oddiyligi shuning oqibati hisoblanadi. Tuzilmaviy dekompozitsiyaning talablaridan biri – «bitta modul – bitta funksiya» talabi ham aynan shu bilan aloqadordir.

Aynan shu tushunchalardan kelib chiqqan holda modullar-resurslar kutubxonasi o'rtasida funksiyalarning tuzilmalashmagan maksimallikdan qochish lozim. Masalan, matn muharririni loyihalashda tahrir kirish funksiyasi nazarda tutilsa, funksiyalar kutubxonasini bir modulda, bir qismini boshqa modulda tashkillashtirish, modul tashkil etish afzal.




**2.1-rasm.** Modullar aloqadorligi: a – funksional; b – ketma-ket; d – axborot; e – protsedurali; f – muvaqqat; g – mantiqiy

*Funksiyalarning ketma-ket aloqadorligida* bir funksiyaning chiqishi boshqa funksiya uchun boshlang'ich ma'lumotlar bo'lib xizmat qiladi (2.1-rasm, b). Odatda, bunday modul bitta chiqish nuqtasiga ega bo'ladi, ya'ni ikkita funksiyani bajaruvchi bitta tagdasturni realizatsiya qiladi. Ketma-ket funksiyalar foydalaniladigan ma'lumotlar ketma-ket aloqador ham hisoblanadi. Funksiyalarning ketma-ket aloqadorligiga ega modulni ham ketma-ket, ham funksional aloqadorlikka ega ikki yoki undan ko'proq modulga ajratish mumkin. Bunday modul bir necha funksiyalarni

bajaradi, binobarin uning texnologikligi yomon: testlashni tashkil etish murakkab, modifikatsiyalashni bajarishda esa modul funksiyalarini fikran bo'lishga to'g'ri keladi.

Bir xil ma'lumotlarni qayta ishlovchi funksiyalar axborot aloqador hisoblanadi (2.1-rasm, c). Dasturlashning tuzilmaviy tillaridan foydalanishda funksiyalarning alohida bajarilishi har bir funksiya o'z tagdasturi bilan amalga oshirilishi mumkin. Ilgari bunday hollarda kirishning turli nuqtalaridan va bitta kichik dastur sifatida shakllantirilgan modul qo'llanilardi.

Bir necha funksiyalar birlashtirilishiga qaramay, axborot aloqador modul texnologiklikning yomon bo'lmagan ko'rsatkichlariga ega. Bu hol shu bilan izohlanadiki, ayrim ma'lumotlar bilan ishlovchi barcha funksiyalar bir joyga yig'ilgan, bu esa ma'lumotlar formati o'zgarganda faqat bitta modulni to'g'rilashga imkon beradi. Bitta funksiya qayta ishlaydigan ma'lumotlar ham axborot aloqador sanaladi.

*Bitta jarayon qismlari* hisoblangan funksiyalar yoki ma'lumotlar protsedurali aloqador (2.1-rasm, d). Odatda modulda dasturning muqobil qismlari funksiyalari birlashtirilgan taqdirda funksiyalarning protsedurali aloqadorligiga ega modul olinadi. Protsedurali aloqadorlikda modulning alohida elementlari g'oyatda zaif bog'lanadi, chunki ular ijobatlaydigan amallar faqat umumiy jarayon bilan bog'langan, binobarin mazkur aloqa xilining texnologikligi avvalgi xilga qaraganda pastroq.

Funksiyalarning *muvaqqat aloqadorligi* mazkur funksiyalar parallel ravishda yoki ma'lum vaqt davri mobaynida bajarilishini ko'zda tutadi (2.1-rasm, e). Ma'lumotlarning muvaqqat aloqadorligi ulardan ma'lum vaqt intervalida foydalanishni anglatadi. Masalan, ma'lum jarayonni initsializatsiya qilishda bajariladigan funksiyalar muvaqqat aloqadorlikka ega. Muvaqqat aloqadorlikning o'ziga xos xususiyati shuki, bunday funksiyalar ijobatlaydigan amallar odatda har qanday tartibda bajarilishi mumkin. Funksiyalarning muvaqqat aloqadorligiga ega modulning borligiga o'zgarish tamoyili xosdir: unga bitta amallar qo'shilishi yoki

undan eskilari chiqarilishi mumkin. Funksiyalar modifikatsiyalanishining bitta ehtimoli mazkur xil modullari texnologikligi ko'rsatkichlarini avvalgisiga qiyosan yanada ko'proq pasaytiradi.

*Mantiqiy aloqa* ma'lumotlarning yoki funksiyalarning bitta mantiqiy guruhga birlashtirilishiga asoslanadi (2.1-rasm, f). Misol sifatida matnli axborotlarni yoki bir xil ma'lumotlarni qayta ishlash funksiyalarini keltirish mumkin. Funksiyalarning mantiqiy aloqadorligiga ega modul bitta operatsiyaning muqobil variantlarini, masalan butun sonlarning qo'shilishi va haqiqiy sonlarning qo'shilishi kabi amallarni ko'pincha ijobatlaydi. Bunday moduldan doimo uning qandaydir qismi chaqiriladi, bunda chaqiruvchi va chaqiriluvchi modullar boshqaruv bo'yicha bog'langan bo'ladi. Mantiqiy bog'langan komponentlarga ega modullar ishi mantiqiysini tushunish muvaqqat aloqadorlikdan foydalanuvchi modullar ishini tushunishdan ko'ra, odamda murakkab, binobarin ularning texnologiklik ko'rsatkichlari yanada past.

Elementlar o'rtasidagi aloqa kichik bo'lsa yoki mavjud bo'lmagan holatda ular tasodifiy aloqadorlikka ega hisoblanadi. Elementlari tasodifiy bog'langan modul eng past texnologiklik ko'rsatkichlariga egadir, chunki unga birlashtirilgan elementlar umuman bog'lanmagan.

Keyingi uchta holatda moduldagi bir necha tagdasturlar o'rtasidagi aloqa tashqi sabablar bilan shartlanganligiga e'tibor qarating. Oxirgisida esa umuman mavjud emas. Bu tegishli ravishda modullarning texnologik tavsiflariga o'tadi. 2.2-jadvalda ekspertlik baholari bo'yicha aloqadorlikning turli xillari tavsifi taqdim etilgan.

2.2-jadval tahlili ko'rsatadiki, amaliyotda funksional, ketma-ket va axborot aloqadorlikdan foydalanish maqsadga muvofiq.

Odatda, yaxshi o'ylangan dekompozitsiyada iyerarxiyaning yuqori darajalari modullarining funksiyalari va ma'lumotlari funksional yoki ketma-ket aloqadorlikka ega bo'ladi. Ma'lumotlarga xizmat ko'rsatish modullari uchun funksiyalarning axborot aloqadorligi xosdir. Bunday modullarning ma'lumotlari turlicha

bog'lanishi mumkin. Jumladan, obyektli-mo'ljalli yondashuvda sinflar bayoniga ega modullar usullarning axborot aloqadorligi va ma'lumotlarning funksional aloqadorligi bilan tavsiflanadi. Dekompozitsiya jarayonida aloqadorlikning boshqa xillariga ega modullarni olish yetarlicha o'ylanmagan loyihalashni anglatadi. Faqat resurslar kutubxonasi bundan mustasno.

*2.2-jadval*

**Aloqadorlikning turli xillari**

| Aloqadorlik xili | Ulashuv bali | Ko'rgazmalilik (tushunarlilik) | O'zgarish imkoniyati | Kuzatiluvchanlik |
|---|---|---|---|---|
| Funksional | 10 | yaxshi | yaxshi | yaxshi |
| Ketma-ket | 9 | yaxshi | yaxshi | yaxshi |
| Axborot | 8 | o'rtacha | o'rtacha | o'rtacha |
| Protseduraviy | 5 | o'rtacha | o'rtacha | o'rtacha |
| Muvaqqat | 3 | o'rtacha | o'rtacha | o'rtacha |
| Mantiqiy | 1 | yomon | yomon | yomon |
| Tasodifiy | 0 | yomon | yomon | yomon |

*Resurslar kutubxonasi.* Ikki tipdagi resurslar kutubxonasi farqlanadi: tagdasturlar kutubxonasi hamda sinflar kutubxonasi.

Tagdasturlar kutubxonasi ahamiyati bo'yicha yaqin funksiyalarni ijobatlaydi, masalan axborotni grafik chiqarish kutubxonasi. Bunday kutubxonada tagdasturlarning o'zaro aloqadorligi mantiqiy, tagdasturlarning o'z aloqadorligi esa funksional, chunki ularning har biri odatda bitta funksiyani ijobatlaydi.

Sinflar kutubxonasi ahamiyatiga ko'ra yaqin sinflarni ijobatlaydi. Sinf elementlari aloqadorligi axborot, sinflarning o'zaro aloqadorligi qardosh yoki assotsialangan sinflar uchun funksional, boshqalari uchun mantiqiy bo'lishi mumkin.

Resurslar kutubxonalari texnologik tavsiflarini yaxshilashning vositasi sifatida hozirgi paytda modul jismini interfeys qismga va amalga oshirish sohasiga ajratish (Pascal da – Interface va Implementation seksiyalari, h va cpp – fayllar C++ va Java da) keng qo'llanilmoqda.

Mazkur holatda interfeys qism resurslar e'lonlari jamlanmasini (tagdasturlar sarlavhalari, o'zgaruvchanlar, tiplar, sinflar nomalari va h.k.) ichiga oladi, ularni tegishli kutubxona boshqa modullarga taqdim etadi. E'lonlari interfeys qismda mavjud bo'lmagan resurslar tashqaridan daxlsiz. Amalga oshirish sohasi tagdasturlar jismlarini va ushbu tagdasturlar foydalanuvchi ichki resurslarni (tagdasturlar, o'zgaruvchan tiplarni) ham ehtimol qamrab olishi mumkin. Bunday tashkillashtirishda kutubxona amalga oshirivining har qanday o'zgarishlari uning interfeysiga daxl qilmasa, kutubxona bilan bog'liq modullarni qayta qarab chiqishni talab qilmaydi, bu esa kutubxonalar modullari texnologik tavsiflarni yaxshilaydi. Bundan tashqari, bu kabi kutubxonalar, qoidaga ko'ra yaxshi sozlangan va o'ylangan, chunki turli dasturlar tomonidan tez-tez foydalaniladi.

## 2.3. Pastlashuvchi va ko'tariluvchi dasturiy ta'minlashni ishlab chiqish

Dekompozitsiyada olingan tuzilmaviy iyerarxiya komponentlarini loyihalashda, amalga oshirishda va testlashda ikki xil yondashuv qo'llaniladi:

- ko'tariluvchi;
- pastlashuvchi.

Adabiyotlarda «yadro kengayishi» nomini olgan yana bir yondashuv uchraydi. Bunda nazarda tutiladiki, birinchi navbatda qandaydir asos dasturiy ta'minlash yadrosi, masalan ma'lumotlar va ularga bog'liq protseduralar tuzilmalari loyihalashtiriladi hamda ishlab chiqiladi. So'ngra ko'tariluvchi va pastlashuvchi usullarni kombinatsiyalagan holda yadro kuchaytiriladi. Amaliyotda mazkur yondashuv yadro darajasiga bog'liq holda aslida yoki pastlashuvchi yoki ko'tariluvchi yondashuvga mujassamlashadi.

**Ko'tariluvchi yondashuv.** Ko'tariluvchi yondashuvdan foydalanishdan oldin quyi daraja komponentlari loyihalanadi va amalga oshiriladi, keyin avvalgisi bo'yicha shu ishlar qilinib, komponentlar testlanishi va sozlanishi tugallanishi davomida ularni yig'ish

amalga oshiriladi. Bunda quyi daraja komponentlari ushbu yondashuvda ko'pincha komponentlar kutubxonasiga joylashtiriladi.

Komponentlarni testlash va sozlash uchun maxsus testlovchi dasturlar loyihalanadi hamda amalga oshiriladi.

Yondashuv quyidagi kamchiliklarga ega:

- ixtisoslashtirish to'liq emasligi oqibatida komponentlar nomuvofiqligi ehtimolining ortishi;
- komponentlarga qayta tuzish mumkin bo'lmagan testlovchi dasturlarni loyihalash va amalga oshirishda sarf-xarajat mavjudligi;
- initerfeysning kechikishli loyihalanishi, muvofiq ravishda uni ixtisoslashtirilishining aniqlashtirilishi uchun buyurtmachiga namoyish etish imkoniyati yo'qligi;
- ko'tariluvchi yondashuv tarixan avval paydo bo'lgan. Bu dasturchilar mushohadasining o'ziga xosligi bilan bog'liq bo'lib, ular tahsil jarayonida o'rtacha dasturlarni yozishda avval quyi darajalar komponentlarini (tagdasturlarni, sinflarni) detallashtirishga o'rganib qolishadi. Bu esa yuqori darajalar jarayonlarini yaxshiroq anglashda ularga imkon beradi. Dasturiy ta'minotning sanoatda tayyorlanishida ko'tariluvchi yondashuvdan hozirgi paytda aslida foydalanilmayapti.

**Pastlashuvchi yondashuv.** Pastlashuvchi yondashuv shuni nazarda tutadiki, komponentlarni loyihalash va keyingi amalga oshirish «yuqoridan-quyiga» bo'yicha bajariladi, ya'ni avval iyerarxiya yuqori darajalari komponentlari loyihalanadi, so'ngra keyingilari, shu tariqa eng quyi darajalarda davom ettiriladi. Aynan shunday davomiylikda komponentlar amalga oshiruvi ham bajariladi. Bunda quyi, hali amalga oshirilmagan darajalar komponentlari dasturlashtirish jarayonida maxsus ishlab chiqilgan sozlovchi modullar «tiqin»lar bilan almashtiriladi, bu esa amalga oshirib bo'lingan qismni testlash va sozlashga imkon beradi.

Pastlashuvchi yondashuvdan foydalanishda komponentlarni loyihalash va amalga oshirish ketma-ketligini belgilashning iyerarxik, operatsion hamda kombinatsiyalangan usullari qo'llaniladi.

Iyerarxik usul ishlab chiqish qat'iyan darajalar bo'yicha bajarilishini nazarda tutadi. Ma'lumotlar bo'yicha qaramlik mavjudligida istisnolarga yo'l qo'yiladi, ya'ni qandaydir modul boshqasining natijalaridan foydalanayotgani aniqlanilsa, uni o'sha moduldan keyin dasturlash tavsiya etiladi. Mazkur usulning asosiy muammosi yetarlicha murakkab «tiqin»larning katta miqdori hisoblanadi. Bundan tashqari, mazkur usuldan foydalanishda modullarning asosiy qismi loyiha ustidagi ish yakunida ishlab chiqiladi va ijobatlanadiki, bu inson resurslarining taqsimotini qiyinlashtiradi.

Operatsion usul modullar ishlab chiqilishi ketma-ketligini dastur ishga tushirilishda ularning bajarilish tartibi bilan bog'laydi. Usulning qo'llanishi modullarni bajarish tartibi ma'lumotlarga bog'liq bo'lib qolishi bois murakkablashadi.

*Kombinatsiyalangan usul* ishlab chiqish ketma-ketligiga ta'sir qiluvchi quyidagi omillarni hisobga oladi:

- modulning erishuvchanligi – mazkur modulning chaqiruv zanjirida barcha modullar mavjudligi;
- ma'lumotlar bo'yicha qaramlik – ayrim ma'lumotlarni shakllantiruvchi modullar qayta ishlanuvchilardan oldin yaratilishi shart;
- natijalarni berish imkoniyatini ta'minlash – natijalarni chiqarish modullari qayta ishlovchilardan ilgari ishlab chiqilishi shart;
- yordamchi modullar shayligi – yordamchi modullar, masalan, fayllarni yopish, dasturni yakunlash modullari qayta ishlanuvchilardan oldin yaratilishi shart;
- zarur resurslar mavjudligi.

Bundan tashqari, boshqa teng sharoitlarda murakkab modullar soddalaridan avval ishlab chiqilishi shart, chunki ularning loyihalanishida ixtisoslashtirishlardagi noaniqliklar aniqlanishi mumkinki, bu qanchalik erta yuz bersa, shunchalik yaxshi.

Pastlashuvchi yondashuv maxsus ta'kidlangan holatlarda komponentlarni ishlab chiqishning pastlashuvchi ketma-ketligi buzilishiga yo'l qo'yadi. Jumladan, agar quyi darajadagi ayrim komponentdan yanada yuqori darajalardagi ko'plab komponentlar foydalansa, mazkur komponentni uni chaqiruvchi kompo-

nentlardan ko'ra avval loyihalash va ishlab chiqish tavsiya etiladi. Nihoyat, birinchi navbatda noto'g'ri ma'lumotlarni oxirgi qayta ishlash komponentlari loyihalashtiriladi va ishlab chiqiladi.

Modullarni amalga oshirishning ketma-ketligini belgilash misoli 4.5-mavzuda qarab chiqiladi.

Pastlashuvchi yondashuvdan obyektli-mo'ljalli dasturlashda ham odatda, foydalaniladi. Yondashuv tavsiyalariga muvofiq dastlab dasturiy ta'minlashning foydalanuvchi interfeysi loyihalashtiriladi va ishlab chiqiladi, so'ngra predmetli soha ayrim asos obyektlarining sinflari ishlab chiqiladi, shundan keyingina mazkur obyektlardan foydalangan holda boshqa komponentlar loyihalashtiriladi hamda amalga oshiriladi.

Pastlashuvchi yondashuv tomonidan quyidagilar ta'minlanadi:

- loyihalashtirilgan komponent ixtisoslashtirilishlarini maksimal to'liq belgilash va komponentlarning o'zaro muvofiqligi;
- foydalanuvchi interfeysini erta belgilash, buning namoyishi yaratilayotgan dasturiy ta'minlash talablarini aniqlashtirish uchun buyurtmachiga imkon beradi;
- pastlashuvchi testlash va kompleks sozlash imkoniyati.

## 2.4. Tuzilmaviy va «notuzilmaviy» dasturlash. Tuzilmaviy algoritmlar bayoni vositalari

Ishlab chiqilayotgan dasturiy ta'minlash texnologiklashning yuqori darajasini ta'minlash usullaridan biri *tuzilmaviy dasturlashdir.*

Dasturlar ijobatlaydigan hisoblash jarayonining uch xili farqlaniladi: yo'nalishli, tarmoqli va siklik. Hisoblash jarayonining yo'nalishli tuzilmasi natija olish uchun ayrim operatsiyalarni muayyan ketma-ketlikda bajarish zarurligini nazarda tutadi.

Hisoblash jarayonining *tarmoqli tuzilmasi* operatsiyalarning muayyan ketma-ketligi bir yoki bir necha o'zgaruvchanlar ifodalariga bog'liqligini nazarda tutadi.

Dasturlarda ko'rsatilgan hisoblash jarayonlarini amalga oshirish uchun tegishli boshqaruvchi operatorlardan foydalaniladi. Yuqori darajadagi dasturlashning FORTRAN kabi birinchi

protsedurali tillari «hisoblash jarayoni tipi» tushunchasi bilan operatsiyalar qilmasdi. Ularda operatorlarning yo'nalishli ketma-ketligini o'zgartirish uchun quyi darajadagi tillardagi singari boshqaruvni berishning shartli (ayrim sharoitlarni bajarishda) va shartsiz buyruqlaridan foydalanilardi. Shu boisdan ushbu tillarda yozilgan dasturlar hozirgi paytda faqat quyi darajali (mashina) tillariga xos chigal tuzilmaga ega bo'lardi.

*2.3-jadval*

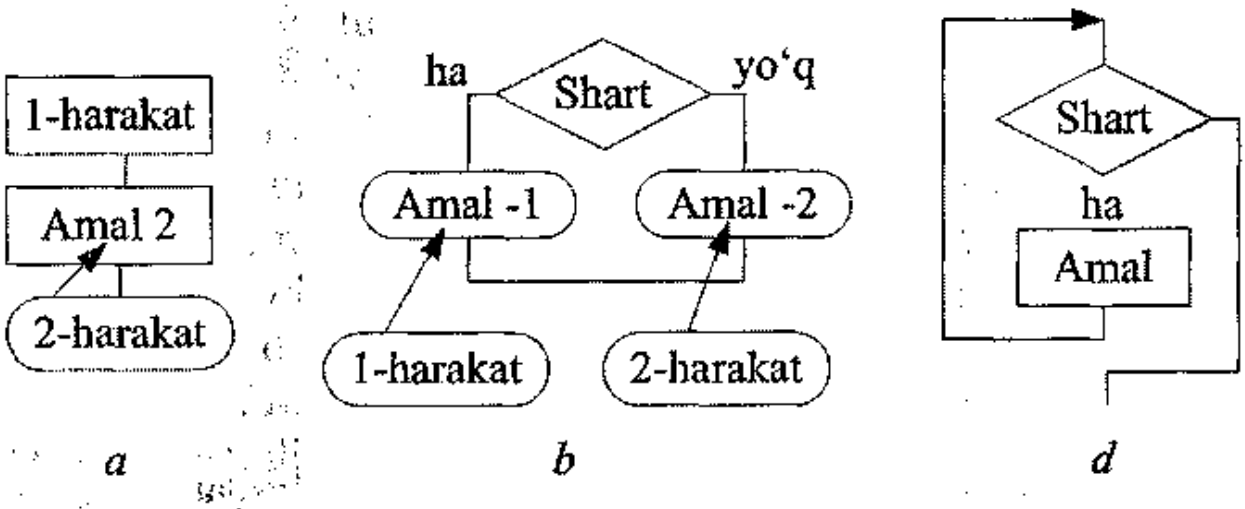
**Dasturlar algoritmlarining sxemalari**

| Blok nomi | Ifodalanish | Blok vazifasi |
|---|---|---|
| Terminator | Harakat | Dastur yoki tagdastur boshlanishi, yakuni |
| Jarayon | Harakat | Ma'lumotlarni qayta ishlash (hisoblashlar, yuborish va h.k.) |
| Ma'lumotlar | Ma'lumotlar | Kiritish-chiqarish operatsiyalari |
| Qaror | Shart | Tarmoqlar tanlash, iteratsion va qidiruv sikllari |
| Tayyorgarlik | Harakat | Hisob sikllari |
| Sikl chegarasi | Boshlanish | Maqbul sikllar |
| | Tamom | |
| Oldindan belgilangan jarayon | Nom | Protseduralar chaqiruvi |
| Biriktiruvchi | Hom | Yo'nalishlar uzilishlari rusumlanishi |
| Izoh | ....[Izoh | Operatsiyalarga tushuntirish |

Aynan shunday dasturlar algoritmlarining sxemalarini aks ettirish uchun o'z paytida Rossiya Federatsiyasining GOST 19.701-90 ishlab chiqilgandi, bunga muvofiq amallarning har bir guruhiga muvofiqlikda maxsus blok qo'yilardi (2.3-jadval). Garchi

mazkur standart siklarni ifodalash bloklarini nazarda tutsa-da, boshqaruvning ixtiyoriy berilishini ham taqiqlamaydi, ya'ni algoritmini amalga oshirishda va shartsiz berish buyruqlaridan foydalanishga yo'l qo'yadi.

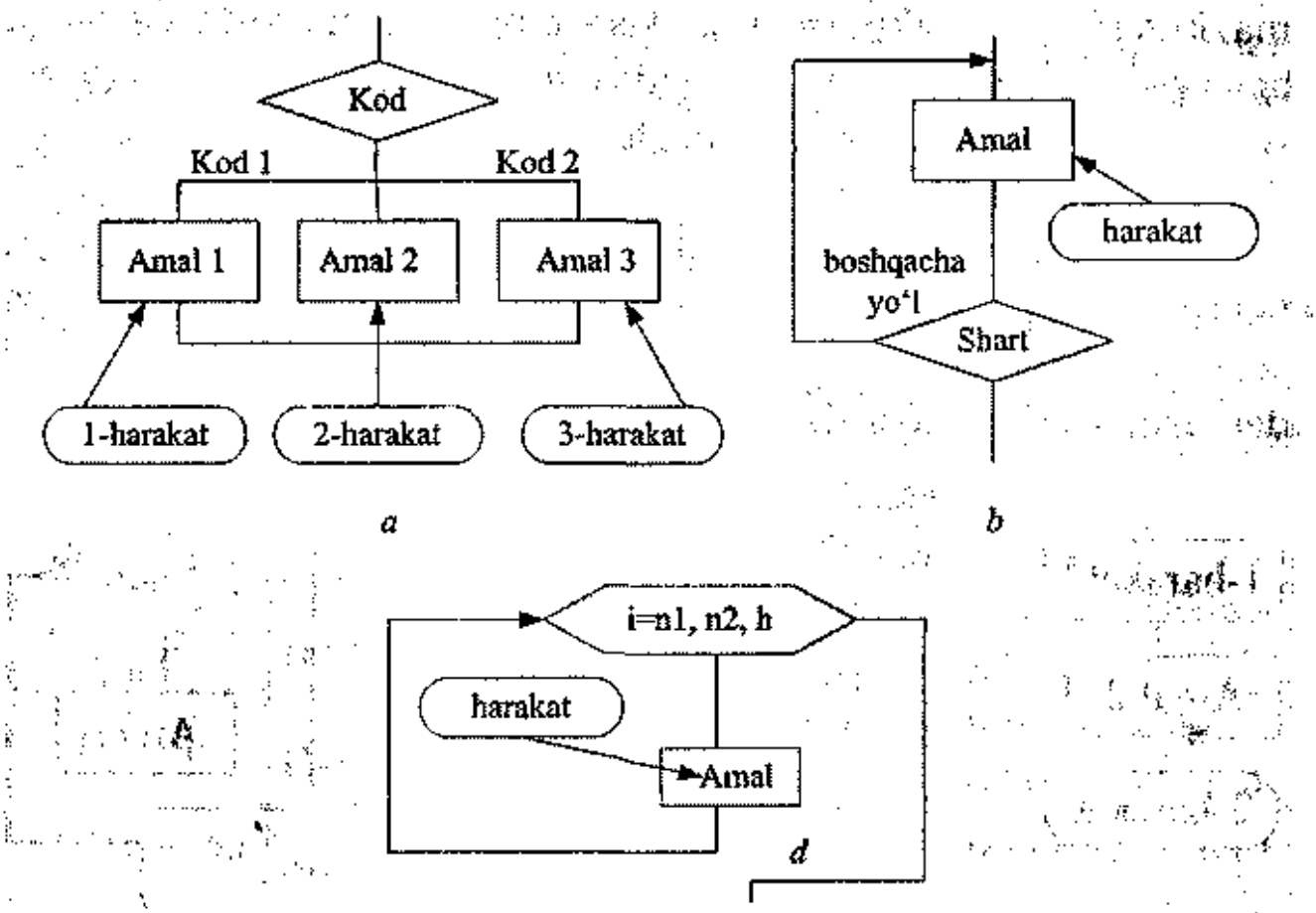
Hisoblash jarayonining talab qilinuvchi tipini tashkillashtirish uchun boshqaruvni berish buyruqlaridan foydalanishning o'ziga xosliklarini namoyon etuvchi misol sifatida Assembler tipidagi dasturni ko'rib chiqamiz.




**2.2-rasm.** Tayanch algoritmik tuzilmalar:
a – ergashish; b – tarmoqlanish; d – sikl-hozircha.

XX asrning 60-yillarida har qanday istalgan murakkab algoritmni uchta asosiy boshqaruvchi konstruksiyalar orqali taqdim etish mumkinligi isbotlangandan so'ng yuqori darajadagi dasturlash tillarida tegishli konstruksiyalarni amalga oshirish uchun boshqaruvchi operatorlar paydo bo'ldi. Mazkur uch konstruksiyani asos kontsruksiyalarda hisoblash qabul qilingan. Ularga quyidagi konstruksiyalarga mansub:

- *ergashish* – amallarning ketma-ket bajarilishini ifodalaydi (2.2, a-rasm);
- *tarmoqlanish* – amallarning ikki variantidan birini tanlashga muvofiq keladi (2.2, b-rasm);
- sikl – hozircha bajarilishi sikl boshlanishida tekshiriladigan ayrim shart hozircha buzilmaguncha amallar takrorlanishini belgilaydi (2.2, d-rasm).




**2.3-rasm.** Algoritmlarning qo'shimcha tuzilmalari:
a – tanlash; b – sikl – to; d – takrorlanishlar soni bilan berilgan sikl.

Asos konstruksiyalardan tashqari yuqori darajadagi dasturlashning protsedurali tillari asos konstruksiyalardan tarkib toptirish mumkin bo'lgan yana uchta konstruksiyadan foydalaniladi:

- *tanlash* – ayrim kattalikning ifodasiga bog'liqlikda bir necha variantdan bir variantni tanlashni bildiradi (2.3, a-rasm);
- *sikl – to* siklida amallar bajarilgandan so'ng tekshiruvi amalga oshiriladigan topshirilgan shart *to* bajarilguncha ayrim amallarning takrorlanishini bildiradi (2.3, b-rasm);
- *takrorlanishlarning berilgan soni bilan birgalikdagi sikl* (hisob sikli) – ayrim amallarning bir qancha miqdorida takrorlanishini bildiradi (2.3, d-rasm).

Ko'rsatilgan qo'shimcha konstruksiyalar har qanday asos konstruksiyalar orqali yengil ijobatlanadi.

Sanab o'tilgan olti konstruksiya tuzilmaviy dasturlash negiziga qo'yilgandi.

**Eslatma.** Mazkur qoidaga «tuzilmaviy dasturlash»da faqat sanab o'tilgan konstruksiyalar (tuzilmalar) qo'llanilganligi faktini ta'kidlaydi. «go to siz dasturlash» tushunchasi ham shundan.

Boshqaruvni berish faqat tuzilmaviy dasturlar deyiladi.

Assembler tegishli konstruksiyalarni nazarda tutmasligiga qaramay, unda ham «tuzilmali» dasturlash mumkin.

Eng so'ngining klassik misoli izlash siklini boshqaruvni notuzilmaviy berishdan foydalangan holda tashkillashtirish (2.4, a-rasm) hamda ekvivalent tuzilmaviy variant (2.4, b-rasm) bo'la oladi.

Sxemalardan tashqari, algoritmlar bayoni uchun Flow – shakllar va Nassi-Shneyderman diagrammalari psevdokodlaridan foydalanish mumkin. Barcha sanab o'tilgan notatsiyalar bir tomondan tuzilmaviy dasturlashdagi kabi aynan o'sha asosiy tuzilmalarga asoslanadi, boshqa tomondan esa detallashtirishning turlicha darajalariga yo'l qo'yadi.

**Psevdokodlar.** *Psevdokod* – algoritmning shakllashtirilgan matnli bayoni (matnli notatsiya). Adabiyotlarda psevdokodlarning bir necha variantlari taklif etilgandi. Ulardan biri 2.4-jadvalda keltirilgan.

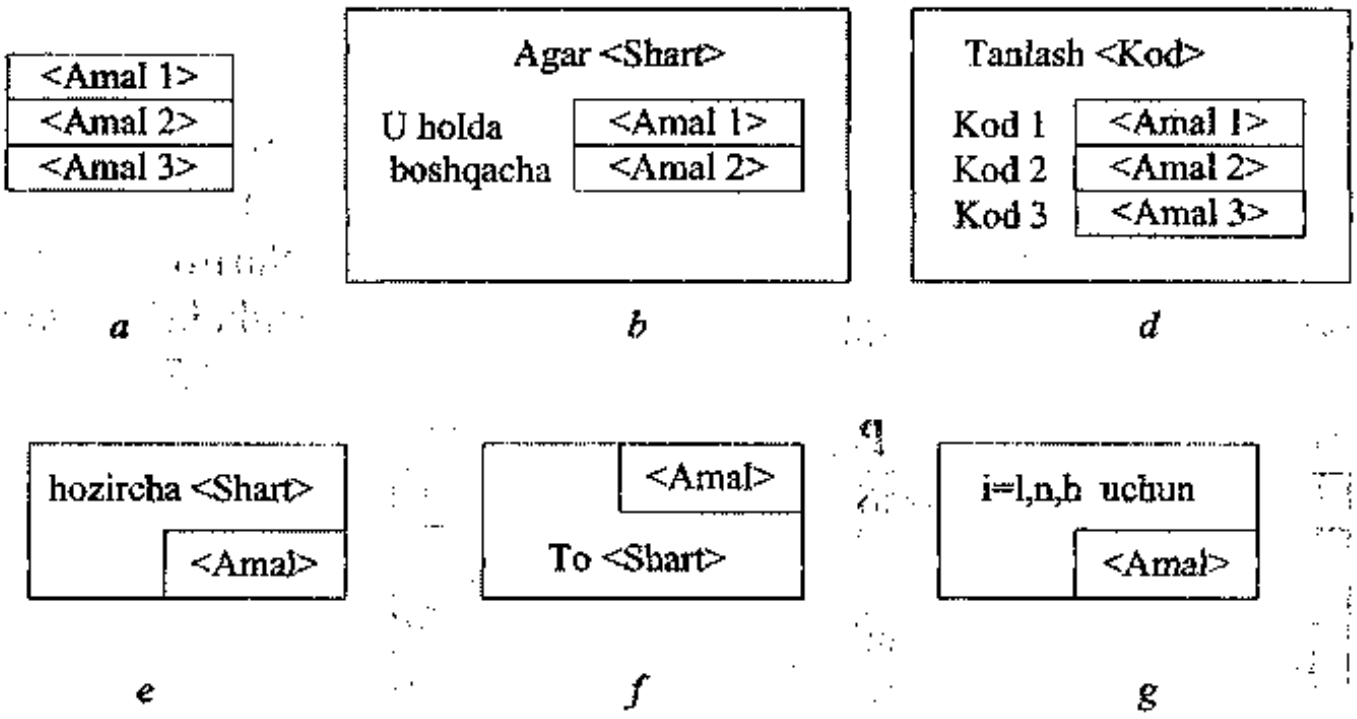
*2.4-jadval*

**Psevdokod variantlari**

| Tuzilma | Psevdokod | Tuzilma | Psevdokod |
|---|---|---|---|
| Ergashish | <Amal 1><br><Amal 2> | Tanlash | Tanlash <kod><br><kod 1>: <amal 1><br><kod 1>: <amal 1><br>...<br>hammasi tanlash |
| Tarmoq-lanish | Agar <shart><br>u holda <amal 1><br>boshqacha <amal 2><br>hammasi agar | Takror-lanishlar topshirilgan miqdorli sikl | uchun <indeks>=<br><n>,<k>, <h><br><amal> |
| Sikl hozir-cha | Sikl hozircha<br><shart><br><amal><br>hammasi sikl | Sikl – To | Bajarilsin<br><amal><br>To <shart> |

Psevdokodlar yordamida notuzilmaviy algoritmni bayon etish imkoni yo'q.

Psevdokodlardan foydalanish avval-boshdan loyihalovchini faqat boshqaruvni berishning tuzilmaviy usullariga yo'naltiradi, shu boisdan ishlab chiqilayotgan algoritmning yanada batafsil tahlilini talab etadi. Algoritmlar sxemalaridan farqli o'laroq, psevdokodlar loyihalanayotgan operatsiyalar detallashtirilishi pog'onasini chegaralamaydi. Ular amal detallashtirilishi pog'onasini ushbu amal ko'rib chiqiladigan abstraksiya darajasi bilan o'lchamlashtirishga imkon beradi hamda tuzilmaviy dasturlashning asosiy usuli qadam-baqadam detallashtirish usuli bilan yaxshi muvofiqlashadi.

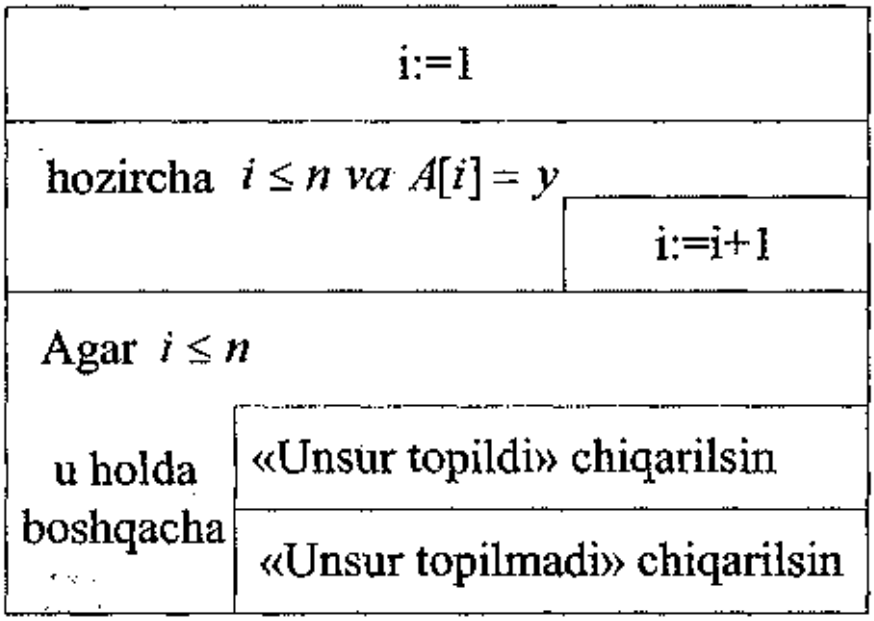



**2.4-rasm.** Asosiy konstruksiyalar uchun Flow-shakl shartli belgilari: a — ergashish; b — tarmoqlanish; d — tanlash; e — sikl-hozircha; f — sikl-to; g — hisobli sikl.

**Flow-shakllar.** Flow-shakllar tuzilmaviy algoritmlar bayonining tuzilmalar sarflanishini namoyon etuvchi grafik notatsiyasini o'zida ifoda etadi. Flow-shaklning har bir timsoli boshqaruvchi tuzilmaga mos keladi va to'g'riburchak tarzida tasvirlanadi. Tuzilmalar sarflanishini namoyish etish uchun Flow-shakllar tim-

soli har qanday boshqa timsolning tegishli to'g'riburchak shakliga yozilishi mumkin. Timsollarning to'g'riburchaklarida matn tabiiy tilda yoki matematik notatsiyada beriladi. To'g'riburchak o'lchami unga yozilgan matn uzunligi va sarflangan to'g'riburchaklar o'lchamlari bilan belgilanadi. Asosiy va qo'shimcha boshqaruvchi konstruksiyalarga mos keluvchi Flow-shakl timsollari 2.4-rasmda keltirilgan.

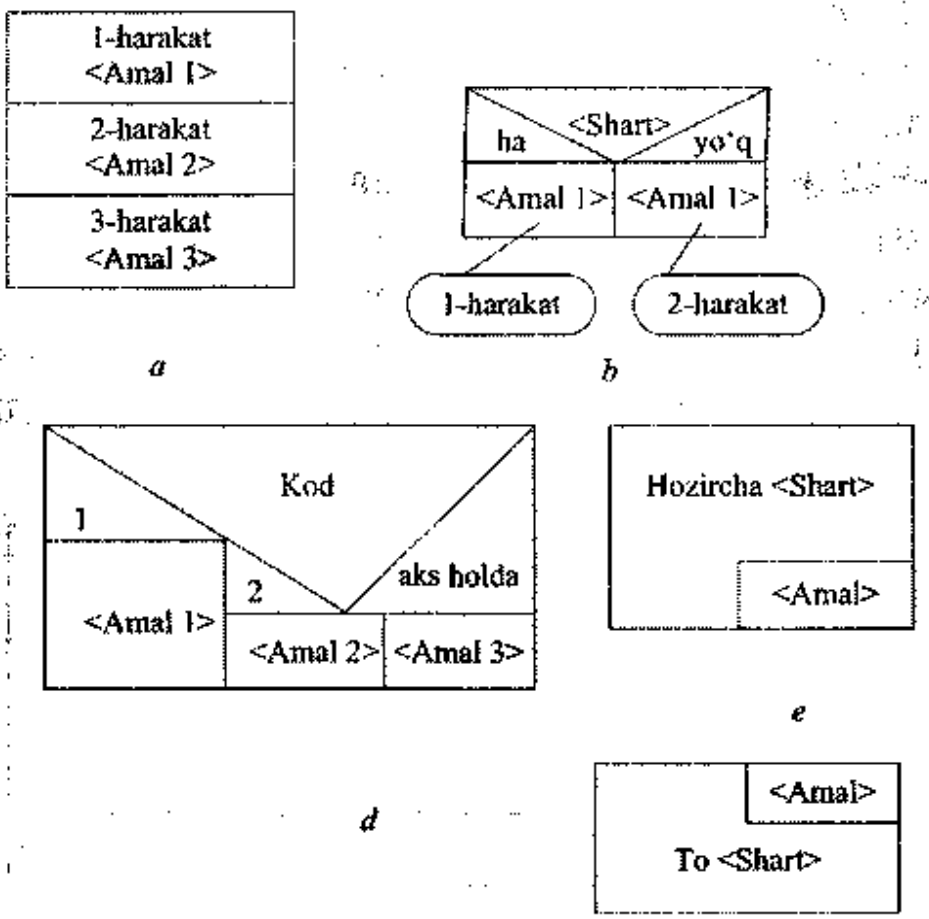
**Nassi-Shneyderman diagrammalari.** Nassi-Shneyderman diagrammalari Flow-shakllar rivojlanishidir. Ularning Flow-shakllardan asosiy farqi shundaki, tarmoqlanish shartlarini va variantlarini belgilash sohasi uchburchak tarzda tasvirlanadi (2.6-rasm). Bundan belgilash algoritm taqdirining katta ko'rgazmaliligini ta'minlaydi.




**2.5-rasm.** Izlash sikli algoritmi.

Shuningdek, psevdokodlardan foydalanishdagi kabi, Flow-shakllar yoki Nassi-Shneyderman diagrammalarini qo'llagan holda notuzilmaviy algoritmni bayon etishning imkoni yo'q (boshqaruvni notuzilmaviy berishlar uchun ushbu notatsiyalarda shartli belgilar mavjud emas). Shu bilan birga ushbu notatsiyalar grafik bo'lgani holda konstruksiyalar sarflanishini psevdokodlardan ko'ra yaxshiroq aks ettiradi.

Flow-shakllar va Nassi-Shneyderman diagrammalari uchun xos umumiy kamchilik timsollar tasvirlari tuzilishining murakkabligi bo'lib, bu ushbu notatsiyalarning katta algoritmlar bayoni uchun amaliy qo'llanishini murakkablashtiradi.




**2.9-rasm.** Nassi-Shneyderman diagrammalarining shartli belgilari: a – ergashish; b – tarmoqlanish; d – tanlash; e – sikl-to.

## 2.5. Dasturni rasmiylashtirish uslubi

Texnologiklik nuqtayi nazaridan dasturning ham muallif, ham uni tekshirishi yoki modifikatsiyalashi ehtimoli bor boshqa dasturchilar tomonidan qabul qilinishini yengillashtiruvchi dasturni rasmiylashtirish uslubi yaxshi hisoblanadi. «Yodda tuting, dasturlarni odamlar o'qishadi» deya uqtirgandi dasturlash muammolariga bag'ishlangan mashhur monografiyalardan birining muallifi D. Van Tissel.

Aynan har qanday dasturni bir necha marta qarab chiqishga to'g'ri kelishidan kelib chiqqan holda dasturlarni yozishning yaxshi uslubiga rioya qilish lozim.

Dasturni rasmiylashtirish uslubi quyidagilarni o'z ichiga oladi:

• dastur obyektlarini (o'zgaruvchanlarni, funksiyalarni, tiplar, ma'lumotlar va hokazolarni) nomlash qoidalari;

• modullarni rasmiylashtirish qoidalari;

• modullar matnlarini rasmiylashtirish uslubi.

**Dastur obyektlarini nomlash qoidalari.**

Dasturiy obyektlar nomlarini tanlashda quyidagi qoidalarga rioya qilish lozim:

• obyekt nomi uning mazmuniga muvofiq bo'lishi shart, masalan:

Maxitem – maksimal element;

Nextitem – keyingi element;

• agar dasturlash tili imkon bersa bir necha so'zdan iborat nomlarni vizual ajratish uchun «—« timsolidan foydalanish mumkin, masalan:

Max_item, Next_item

• yozilishi bo'yicha yaqin nomlardan qochish zarur, masalan:

Index va Indec

**Modullarni rasmiylashtirish qoidalari.** Har bir modul minimum quyidagilarni ichiga olgan sarlavha bilan shaylantirilishi shart:

• modul nomi;

• uning vazifasining qisqacha bayoni;

• o'lchov birliklari ko'rsatilgan holda kirish, chiqish parametrlarining qisqacha bayoni;

• foydalaniluvchi (chaqiriluvchi) modullar ro'yxati;

• algoritm (usul) yoki cheklovlar qisqacha bayoni;

• dastur muallifi ism-sharifi;

• identifikatsiyalovchi axborot (versiya raqami yoki so'nggi to'g'rilash sanasi). Masalan:

Funksiya: Length_Path(n:word; L; array of real): real;

Maqsad: kesimlar umumlashma uzunligini belgilash.

Boshlang'ich ma'lumotlar:

T – kesmalar miqdori;

D – kesmalar uzunliklari massivi (metrlarda).

Natija: uzunlik (metrlarda)
Chaqiriluvchi modullar: yo'q
Algoritm bayoni:

- kesimlar jamg'arish usuli bilan umumlashtiriladi, $n \geq 0$;
- ma'naviy qaram, ya'ni muayyan mashina tipiga yo'naltirilgan va skalyar kodlar optimallashtirishini mashinaviy buyruqlar darajasida bajariladi, masalan, ortiqcha yuborishlarni olib tanlash, yanada samarali buyruqlardan foydalanish va hokazo;
- mashinaviy-mustaqil vositalar optimallashtirishini kirish tili darajasida bajariladi, masalan sikldan konstant (sikl indeksiga bog'liq bo'lmagan) ifodalarning hisoblanishlarini chiqarish va hokazo.

Tabiiyki, kompilyator ishiga aralashish mumkin emas, biroq buyruqlar darajasida dasturni optimallashtirishning ko'plab imkoniyatlari mavjud.

**Xotirani tejash usullari.** Xotirani tejash bo'yicha choralar qabul qilish qandaydir holatlarda ushbu xotiradan tejamsiz foydalanilganini nazarda tutadi. Faqat samaradorlik tavsifiga salmoqli ta'sir ko'rsatuvchi ma'lumotlar joylashuvi operatsiyalarning tahlili ma'noga egaligini hisobga olgan holda xotirani tuzilmaviy tiplarning (massivlar, yozuvlar, obyektlar va hokazolarning) ma'lumotlarga ajratishga alohida e'tibor qaratish lozim.

Xotiradan foydalanishga cheklovlar mavjudligida eng avvalo qayta ishlash jarayonida tuzilmaviy tiplarning boshlang'ich ma'lumotlari dublashtirishni talab qilmaydigan qayta ishlash algoritmlarini tanlash lozim. Yuklatilgan massivda operatsiyani bajaruvchi massivlarni xillash algoritmlari, masalan yaxshi ma'lum «pufakcha» usuli bilan xillashtirish misol tariqasida xizmat qilishi mumkin.

Agar dasturda cheklangan vaqtdan foydalanuvchi katta massivlar zarur bo'lsa, bu holda ularni dinamik xotiraga joylashtirish va qayta ishlash yakunlangandan so'ng yo'qotib tashlash mumkin.

Yana shuni esda tutish lozimki, tuzilmaviy ma'lumotlarni tagdasturga «vazifasi bo'yicha» berishda ushbu ma'lumotlarning

nusxalari stekda joylashadi. Agar ma'lumotlarni «murojaat bo'yicha» biroq o'zgarmas (const bayonida) berilsa, ba'zan nusxalanishdan saqlanishga erishiladi. Oxirgi holatda stekda faqat ma'lumotlar manzili joylashadi. Masalan:

*Type Massiv=array[1...100] of real;*
*Function Yig'indi (Const a: Massiv;...)...*

**Bajarish vaqtini kamaytirish usullari.** Yuqorida avval eslatib o'tilganidek, bajarish vaqtini kamaytirish uchun birinchi navbatda dasturning takrorlanishlar ko'p miqdoriga ega siklik uchastkalarini tahlil qilish lozim. Ularni yozishda imkoniyatga ko'ra quyidagilar zarur:

- sikllardan konstant, ya'ni sikl parametrlariga bog'liq bo'lmagan ifodalarning hisoblanishini chiqarish;
- qo'shish, ayirish va surishlar bilan almashtirgan holda ko'paytirish hamda bo'lishning «uzun» operatsiyalaridan qochish;
- tiplarning ifodalarda qayta tuzilishini minimallashtirish;
- shartli ifodalar yozuvni optimallashtirish, ortiqcha tekshiruvlarni chiqarib tashlash;
- indekslar bo'yicha massivlar elementlariga ko'p martalik murojaatlarni (ayniqsa, ko'p o'lchamlarini, chunki element manzili hisoblab chiqarilishida indekslar qiymatiga ko'paytirish operatsiyalaridan foydalaniladi) chiqarib tashlash – xotiradan massiv elementini birinchi marta o'qib, uni skalyar o'zgaruvchanda eslab qolish va kerakli joylarda qo'llash lozim;
- ifodada turli tiplardan foydalanishdan saqlanish va hokazo.

Quyidagi misollarni ko'rib chiqamiz:

**2.1-misol.** Quyidagi tuzilma (Pascal) sikliga ega bo'la qolaylik:

```
for y:=0 to 99 do
  for x:=0 to 99 do
    a[320*x+y]:=S[k,l];
```

Ushbu siklda ko'paytirish va S(k) elementiga murojaat operatsiyalari 10000 marta bajariladi. 320=28+26 ekanligidan foydalanib, siklni optimallashtiramiz:

*skl:=S[k,l]* {sikldan massiv elementiga murojaatni chiqaramiz}

*for x:=0 to 90 do* {sikllarni joylari bilan almashtiramiz}

*begin*

*i:=x shl8+x shl 6;* {ko'paytirishni surishga almashtiramiz va sikldan chiqaramiz}

*for y = 0 to 99 do*

*a[i+y]:=skl;*

*end;...*

Natijada 10000 ko'paytirish operatsiyasi o'rniga 200 surish operatsiyasi bajariladigan bo'ladi, ularning vaqti esa qo'shish operatsiyasini bajarish vaqti bilan taxminan qiyosli. S(K) massivi elementiga murojaat bir marta bajariladi.

**2.2-misol.** Jismida murakkab shart amalga oshirilgan siklga ega bo'la qolaylik:

*for k: = 2 to n do*

*begin*

*if x[k]>yk then S:=S+y[k]-x[k];*

*if (x[k]<=yk) and (y[k]<yk) then S:=S+yk-x[k];*

*end;...*

Ushbu siklda ortiqcha tekshiruvlarni olib tashlash mumkin:

*for k: = 2 to n do*

*begin*

*if x[k]>yk then S:=S+y[k]-x[k]*

*else*

*if y[k]<yk then S:=S+yk-x[k];*

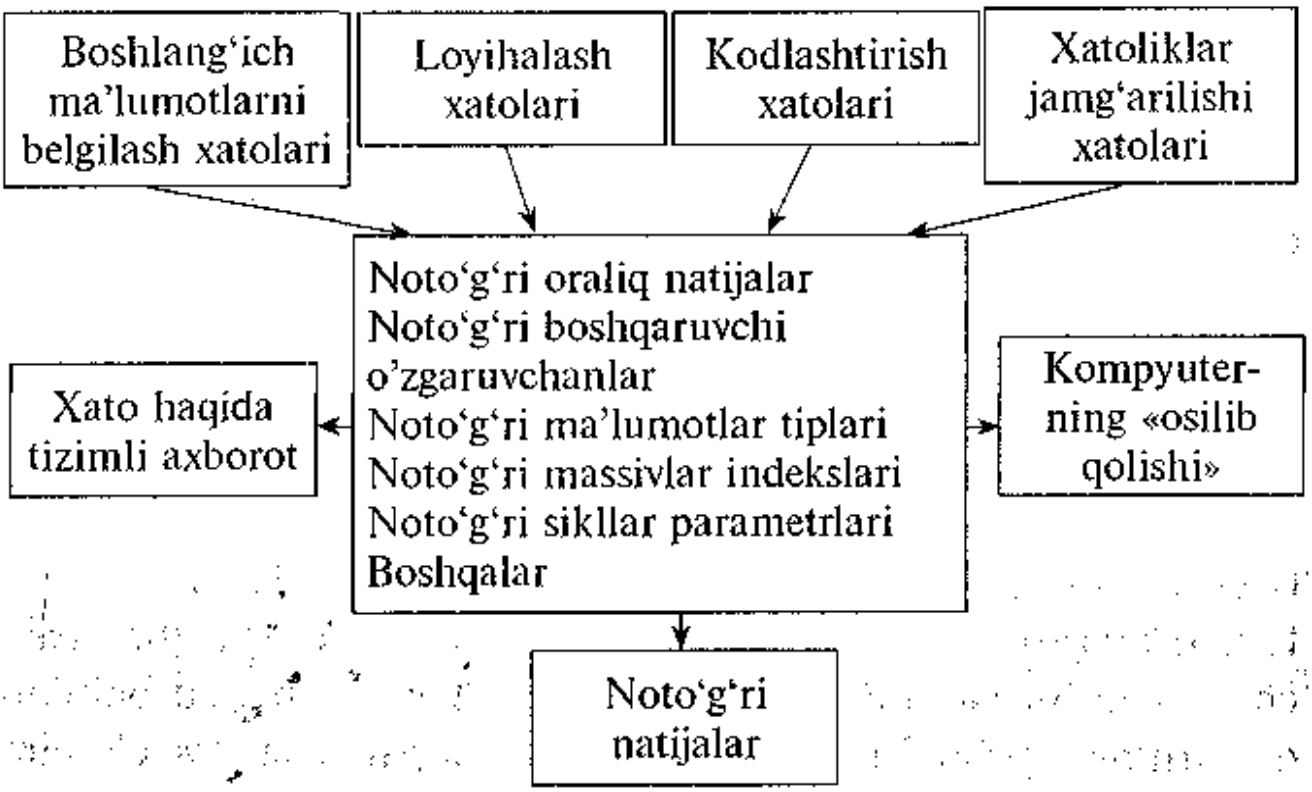
*end;...*

**2.1-misol**da dastur nima qilayotganini tushunish murakkabroq bo'lib qolganiga, 2.2-misolda esa murakkablik amalda yo'qligiga e'tibor bering. Binobarin, birinchi holatda bajarilgan optimallashtirish dastur texnologikligini yomonlashtirishi mumkin, shu boisdan juda xohlanishli emas.

## 2.6. «Xatolardan himoyali» dasturlash

Dastur kompilyatsiyasi va jamlashtirilishi bosqichlarida aniqlanilmaydigan dasturlash xatolaridan istalgani oxir-oqibatda uch usul bilan namoyon bo'lishi mumkin: xato haqida tizimiy axborot berilishiga, kompyuterning osilib qolishiga va noto'g'ri natijalar olinishiga olib keladi.

Biroq dastur ishining natijasi muhim bo'lgunga qadar xatolar odatda ko'p marotaba noto'g'ri oraliq natijalar, noto'g'ri boshqaruvchi o'zgaruvchanlar, noto'g'ri ma'lumotlar tiplari, ma'lumotlar tuzilmalarining indekslari va hokazolar tarzida namoyon bo'ladi (2.7-rasm), bu esa xatolarni ular og'ir oqibatlarga olib kelgunga qadar aniqlanishga va zararlantirishga urinib ko'rish mumkinligini bildiradi.




**2.7-rasm.** Xatolar namoyon bo'lish usullari.
Kiritish-chiqarish operatsiyalari bajarilishing to'g'riligi tekshiruvi.

Xatolarni erta aniqlash va zararsizlantirishning maxsus yo'llari qo'llaniladigan dasturlashni himoyali yoki xatolardan himoyali dasturlash deb nomlanadi. Undan foydalanilganda noto'g'ri natijalar olish ehtimoli salmoqli ravishda kamayadi.

Xatolarni va ularning mumkin bo'lgan erta ko'rinishlarini batafsil tahlili shuni ko'rsatadiki, quyidagilar tekshirilishi maqsadga muvofiqdir:

- kiritish-chiqarish operatsiyalari bajarilishining to'g'riligi;
- oraliq natijalarning (boshqaruvchi o'zgaruvchanlar ifodalarining, ma'lumotlar indekslari, tiplari ifodalarining, raqamli argumentlar ifodalarining va hokazolarning) imkonliligi.

Boshlang'ich ma'lumotlarni noto'g'ri belgilashning sabablari ham ichki xatolar – kiritish-chiqarish yoki dasturiy ta'minlash qurilmalarining xatolari, ham tashqi xatolar – foydalanuvchining xatolari bo'lishi mumkin. Bunda quyidagilarni farqlash qabul qilingan:

- *berish xatolari* – apparat vositalar, masalan nosozlik oqibatida ma'lumotlarni buzadi;
- *qayta tuzish xatolari* – dastur boshlang'ich ma'lumotlarni kirish formatidan ichki formatga noto'g'ri qayta tuzadi;
- *qayta yozuv xatolari* – foydalanuvchi ma'lumotlarni kiritishda yanglishadi, masalan ortiqcha yoki boshqa timsolni kiritadi;
- *ma'lumotlar xatolari* – foydalanuvchi noto'g'ri ma'lumotlarni kiritadi.

Uzatish xatolari odatda apparatli nazorat qilinadi.

Qayta tizimdagi xatolardan himoyalanish uchun kiritishdan so'ng ma'lumotlar odatda darhol foydalanuvchiga namoyish qilinadi («aks-sado»). Bunda dastlab ichki formatga qayta tuzish, so'ngra teskarisi bajariladi. Biroq mazkur bosqichda qayta tuzishning barcha xatolarini bartaraf etish odatda o'ta murakkab, shu boisdan dasturning tegishli fragmentlarini ekvivalent parchalash va chegara ifodalari usullaridan foydalanilgan holda batafsil testlashtiriladi.

Noto'g'ri ma'lumotlarni odatda faqat foydalanuvchi aniqlashi mumkin.

**Oraliq natijalarning imkonliligi tekshiruvi.** Oraliq natijalarning tekshiruvlari nafaqat ma'lumotlarni noto'g'ri belgilash xatolarining, balki kodlashtirish va loyihalashdagi ayrim xatolarning kechikkan namoyoni ehtimolini kamaytirishga ham imkon beradi. Bunday tekshiruv mumkin bo'lishi uchun dasturda har qanday kelib chiq-

ishga ega, masalan modellashtirilayotgan jarayonlar mohiyati bilan bog'liq cheklovlar mavjud o'zgaruvchanlardan foydalanilishi zarur.

Biroq shuni nazarda tutish lozimki, dasturda har qanday qo'shimcha operatsiyalar qo'shimcha resurslardan (vaqt, xotira va hokazo resurslardan) foydalinishni talab qiladi hamda ularda ham xatolar bo'lishi mumkin. Shu boisdan barcha oraliq natijalarni emas, faqat tekshiruv maqsadga muvofiqlarini, ya'ni xatoni aniqlashga imkon berish ehtimoli bor va murakkab bo'lmaganlarini tekshirish ma'no kasb etadi. Masalan:

- agar massiv elementining indeksi qandaydir tarzda hisoblanilsa, bu indeks imkonli ekanligini tekshirish lozim;
- takrorlanishlari miqdori o'zgaruvchan ifodasi bilan belgilanuvchi sikl barpo etilayotgan bo'lsa, ushbu o'zgaruvchan ifodasi salbiy emasligiga ishonch hosil qilish maqsadga muvofiqdir;
- agar qandaydir hodisaning ehtimolligi belgilanayotgan bo'lsa, olingan ifoda 1 dan ko'p emasligini, barcha ehtimolli mustaqil hodisalarning ehtimolligi umumlashmasi 1 ga tengligini va hokazolarni tekshirish maqsadga muvofiq.

**Xatoliklar jamg'arilishini bartaraf etish.** Hisoblashlar natijalarining xatolarini kamaytirish uchun quyidagi tavsiyalarga rioya qilish zarur:

- yaqin raqamlarni ayirishdan saqlanish lozim (mashinali nol);
- katta raqamlarni kichiklariga bo'lishdan saqlanish kerak;
- davomiyligi uzun raqamlarni qo'shishni mutlaq kattalik bo'yicha kichiklaridan boshlash zarur;
- imkon qadar operatsiyalar miqdorini kamaytirishga intilish darkor;
- qusurlar baholanishlari ma'lum usullardan foydalanishi joiz;
- haqiqiy raqamlar tengligi shartidan foydalanmaslik lozim;
- hisoblashlarni ikki yoqlama aniqlik bilan amalga oshirish, natijalarni esa yagona aniqlikda berish shart.

**Istisnolarning qayta ishlanishi.** Ma'lumotlarni kiritishda va hisoblashlar jarayonida to'liq nazorat qilish, qoidaga ko'ra, imkondan hosil bo'lganligi bois avariyaviy vaziyatlar qayta ishlanishini tutib olish ko'zda tutilishi lozim.

Apparatli va dasturli qayd etiluvchi xatolarni tutish hamda qayta ishlash uchun ayrim dasturlash tillarida, masalan Delphi, Pascal, C++, Java da istisnolar qayta ishlanishi vositalari ko'zda tutilgan. Ushbu vositalardan foydalanish foydalanuvchiga hech narsani anglatmaydigan, dasturning avariyali yakunlanishi haqidagi axborotning unga berilishiga yo'l qo'ymaslikka imkon yaratadi. Buning o'rniga dasturchi mazkur xatoni tuzatishga imkon beruvchi amallarni ko'zda tutish imkoniyatini yoki, agar buning iloji yo'q bo'lsa, foydalanuvchiga vaziyatning aniq bayoni bor axborotni berish hamda ishni davom ettirish imkoniyatini oladi.

## 2.7. Oralama tuzilmaviy nazorat

Oralama tuzilmaviy nazorat ishlab chiqish jarayonida xatolarni imkon qadar yanada erta aniqlashni ta'minlashga imkon beruvchi nazorat texnologik operatsiyalarining jamlanmasini ifoda etadi. Nomdagi «oralama» atamasi ishlab chiqishning barcha bosqichlarida nazorat bajarilishini aks ettiradi. «Tuzilmaviy» atamasi har bir bosqichda nazorat qiluvchi operatsiyalar bajarilishi bo'yicha aniq tavsiyalar mavjudligini anglatadi.

Oralama tuzilmaviy nazorat maxsus nazorat sessiyalarida bajarilishi shart, zero ularda ishlab chiquvchilardan tashqari maxsus taklif etilgan ekspertlar qatnashishlari mumkin. Sessiyalar o'rtasidagi vaqt sessiyaga chiqariladigan material hajmini belgilaydi: xususiy sessiyalarda materialni o'rtacha porsiyalarda, kamyob sessiyalarda esa salmoqli fragmentlarda qarab chiqiladi. Navbatdagi sessiya uchun materiallar qatnashchilarga ularni oldindan o'ylab chiqishlari uchun avvalroq berilishi shart.

Dastlabki sessiyalardan biri ixtisoslashtirishlar belgilanishi bosqichida tashkil etilishi darkor. Ushbu sessiyada ixtisoslashtirishlarning to'liqligi va aniqligi tekshiriladi, bunda dasturiy ta'minot ixtisoslashtirishlari qanchalik to'g'ri va to'liq belgilanganligining aniqlashga qodir buyurtmachi yoki predmetli soha bo'yicha mutaxassis qatnashishi maqsadga muvofiq.

Qismlar bo'yicha qo'lda loyihalash bosqichida ishlab chiqilayotgan dasturiy ta'minot algoritmlari ma'lumotlarning muayyan majmualarida tekshiriladi va olingan natijalar tegishli ixtisoslashtirishlar bilan solishtiriladi. Asosiy vazifa – ixtisoslashtirishlar tushunilishi to'g'riligiga ishonch hosil qilish hamda loyihaga asos qilinayotgan konseptual qarorlarning afzalliklari va kamchiliklarini tahlil qilish.

Amalga oshirish bosqichida modullarning ijobatlanish, testlarning terilish rejasi (davomiyligi), shuningdek alohida modullar matnlari tekshiriladi.

Barcha bosqichlar uchun adabiyot manbalari bo'yicha va avvalgi ishlab chiqishlar tajribasidan kelib chiqqan holda shakllantirilgan eng ko'p uchrovchi xatolar ro'yxatiga ega bo'lish maqsadga muvofiq. Bunday ro'yxatlar hammasini birvarakayiga ketma-ket tekshirishga emas, muayyan holatlarda kuch-g'ayratga markazlashtirishga imkon beradi. Bunda barcha topilgan xatolar maxsus hujjatda qayd etiladi, biroq ular tuzatilmaydi.

Oralama tuzilmaviy nazorat xatolarni erta aniqlash bilan birga loyiha bo'yicha sifatli hujjatlar o'z vaqtida tayyorlanishini ham ta'minlaydi.

**Nazorat savollari:**

1. Dasturiy ta'minot texnologikligi deganda nima tushuniladi? Nima uchun?

2. Modul ta'rifini ayting. Ushbu tushunchaning o'zgarishlari nima sababli yuz bergan? Hozirgi paytda modullarga talablar qanday o'zgardi va nima uchun?

3. Modullar aloqadorligi va ulashuvi deganda nima tushuniladi? Aloqadorlikning va ulashuvning qanday tiplari imkonli hisoblanadi va nima uchun? Resurslar kutubxonalarining o'ziga xosligi nimada?

4. Ishlab chiqishga pastlashuvchi yondashuv ko'tariluvchi yondashuvdan nimasi bilan farqlanadi? Mazkur yondashuvlarning afzalliklari va kamchiliklarini sanab ko'rsating.

5. Nima tuzilmaviy dasturlash deb nomlanadi va nima uchun? Asosiy va qo'shimcha tuzilmalarni ayting. Tuzilmaviy dasturlarni loyihalashda algoritmlar sxemalaridan foydalanishning murakkabligi nimadaligini tushuntiring. Tuzilmaviy algoritmlar bayonining qanday usullari mavjud?

6. Raqamlarni 16 lik hisob sanoq tizimiga o'tkazishning tuzilmaviy algoritmini taklif eting. Uni algoritm sxemalaridan, psevdokoddan, Nassi-Shneyderman diagrammalari va Flowshakllar-dan foydalangan holda bayon eting. Sizningcha, keyingi ikki notatsiyaning keng qo'llanishiga to'sqinlik qilayotgan umumiy asosiy kamchilik nimada?

7. Nima dasturni rasmiylashtirishning «yaxshi uslubi» deb nomlanadi va nima uchun? Avvalgi topshiriq yechimini istalgan dasturlash tilida ijobatlang. O'zgaruvchanlarni qanday nomlash lozimligi va qanday izohlar zarurligi haqida o'ylang.

8. «Xatolardan himoyali dasturlash» qanday xatolardan himoyalaydi va nima uchun? «Istisno» atamasi ostida nimani tushunish kerak? Qanday holatlarda «istisnolar» qo'llaniladi?

9. «Oralama tuzilmaviy nazorat» nima uchun shunday nomlangan? «Oralama» nazorat nimani anglatadi? Uning tuzilmaviyligi nimadan iborat?

# 3. DASTURIY TA'MINOTGA VA UNI LOYIHALASH UCHUN BOSHLANG'ICH MA'LUMOTLARGA TALABLARNI BELGILASH

*Vazipredmeting (masalaning) qo'yilish bosqichi* dasturiy mahsulot yaratishning eng mas'uliyatli bosqichlaridan biridir. Ushbu bosqichda ishlab chiqilayotgan dasturiy ta'minlashga asosiy talablar shakllantiriladi. Funksiyalar va tasarrufiy talablar qanchalik to'liq belgilanganiga hamda loyihalash jarayonini belgilovchi prinsipial qarorlar qanchalik to'g'ri qabul qilinganiga ishlab chiqish qiymati va uning sifati ko'p jihatdan bog'liq.

## 3.1. Dasturiy mahsulotlarning funksional alomat bo'yicha tasnifi

Har bir dasturiy mahsulot muayyan funksiyalarni bajarish uchun mo'ljallangan. Barcha dasturiy mahsulotlarni ahamiyatiga ko'ra uch guruhga: tizimli, amaliy va gibrid guruhlarga ajratish mumkin (3.1-rasm).

Tizimli guruhga odatda hisoblash tizimlarining (ham alohida kompyuterlarning, ham tarmoqlarning) funksiyalar bajarilishini ta'minlovchi dasturiy mahsulotlar mansub bo'ladi. Bular operatsion tizimlar, qobiqlar va boshqa dasturlar (utilitlar).

*Operatsion tizimlar,* qoidaga ko'ra, resurslarni (protsessor va xotira bilan), jarayonlarni (vazifalar va oqimlar bilan) va qurilmalarni boshqaradi. Operatsion tizimlarni tashkillashtirishning murakkabligi avtomatlashtirish darajasi va boshqaruv jarayonlarining erishuvchi samaradorligi bilan shartlaniladi. Jumladan, multidasturiy operatsion tizimlar yakkadasturiy tizimlardan ancha murakkabroq, bu MS DOS hamda WINDOWS misolida yaxshi ko'rinadi.

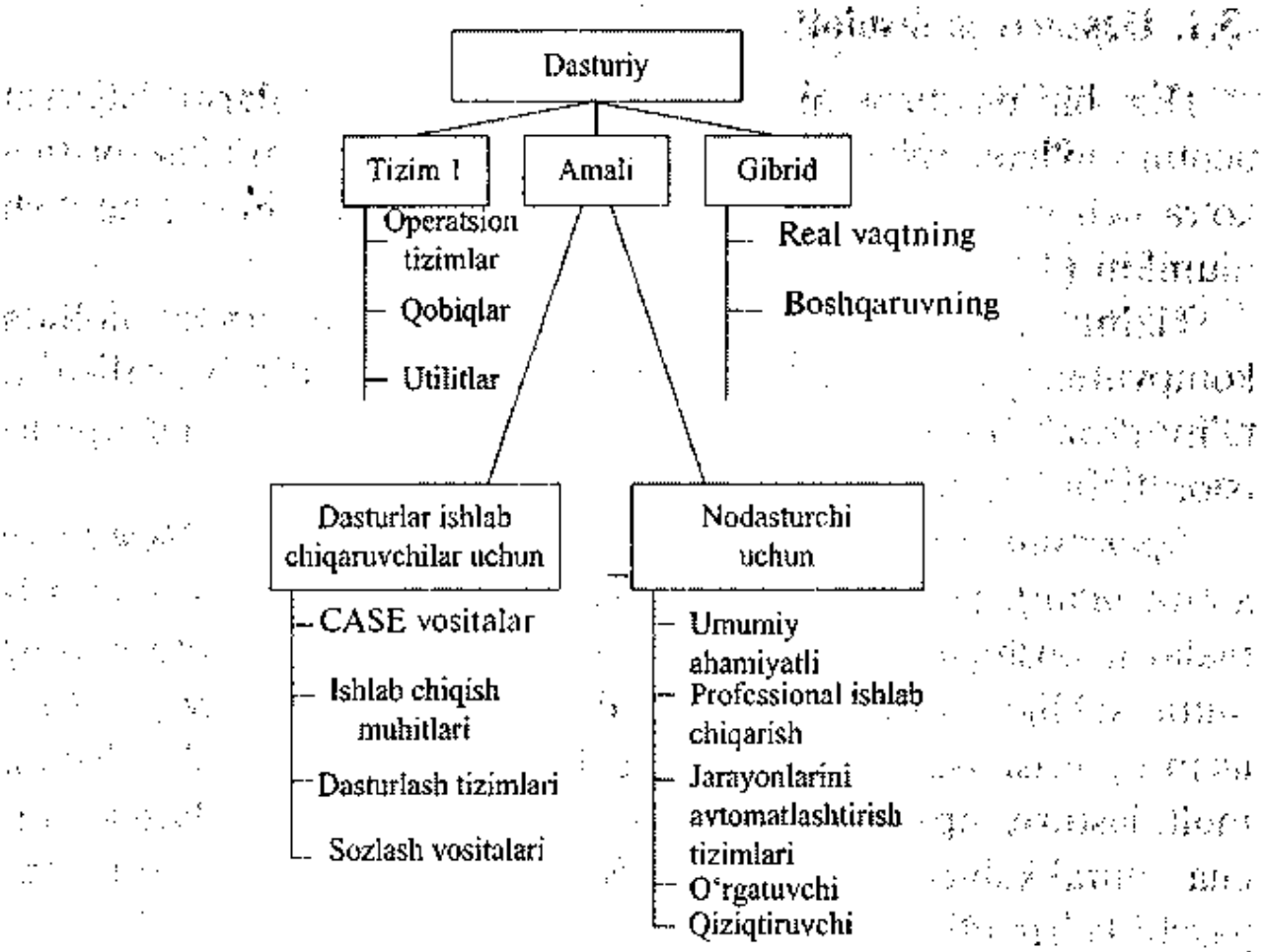
*Qobiqlar* (masalan, NORTON COMANDER) o'z vaqtida tizimli MS DOS fayllaridan foydalanuvchining yanada qulay interfeysini tashkillashtirish uchun paydo bo'lgandi. FAR kabi zamonaviy qobiqlar faylli tizim bilan ishlashda foydalanuvchiga odatiy muhitni ta'minlash uchun qo'llaniladi.

Muayyan funksiyalar, jumladan fayllarni arxivlashtirish, kompyuterni viruslar yuqishidan tekshirish, axborotga uzoqlashgan daxlni amalga oshirish va boshqa funksiyalar bajarilishini ta'minlovchi dasturlar hamda tizimlarni utilitlarga kiritish qabul qilingan.

Amaliy dasturlar va tizimlar muayyan foydalanish vazifalarini hal etishga yo'naltirilgan.

Foydalanuvchilarni ikki xilga ajratiladi:

- dastur ishlab chiquvchilar;
- kompyuter tizimlardan o'z maqsadlariga erishish uchun foydalanuvchi nodasturchilar.




**3.1-rasm.** Dasturiy mahsulotlarning ahamiyati bo'yicha tasnifi.

Utilitlar deganda to'g'ridan to'g'ri operatsiyalar tizim tarkibiga kirmaydigan, lekin fayllarni arxivlash, kompyuterni virus bilan zararlanmaganligini tekshirish, axborotlarga masofadan turib

ruxsatli foydalanish kabi boshqa ishlarni bajarish funksiyalarini ta'minlovchi dasturlar va tizimlar tushuniladi.

Amaliy dasturlar va tizimlar muayyan foydalanuvchining aniq masalalarini hal qilishga mo'ljallangan bo'ladi.

Dasturlar ishlab chiquvchilar keyingi paytda, odatda dasturlash tizimlariga va ishlab chiqish muhitlariga integratsiya qilinayotgan kompilyatorlar, jamlovchilar, sozlovchilar kabi maxsus instrumental vositalaridan foydalanishadi. Zamonaviy dasturlash, muhitlari, masalan Delphi, Visual C++ dasturiy mahsulotlar ishlab chiqishning vizual texnologiyasini ishlatadi va dasturchilar uchun o'z ishlanmalariga kiritishlari mumkin bo'lgan komponentlarning boy kutubxonalarini taqdim etishga imkon beradi. Assess, FoxPro, Oracle kabi ma'lumotlar bazalarini yaratishning instrumental komplekslari, intellektual tizimlar, masalan ekspert, ta'limiy tizimlar, bilimlar nazorati tizimlari va shu kabilarni yaratish vositalari ham ushbu guruhga mansubdir. Ushbu yo'nalishdagi so'nggi yutuq – CASE dasturiy ta'minotni ishlab chiqishning ERwin, BPwin, Paradigm Plus, Rational Rose kabi vositalaridir.

Foydalanuvchi-nodasturchilar zamonaviy talablarga muvofiq dasturiy mahsulotlar yaratish va ularning operatsion tizim bilan o'zaro harakatlanishi spetsifikasi muammolarida professionallar bo'lmasligi shart. Ular uchun aniq predmet sohaga mo'ljallangan maxsus dasturiy mahsulotlar ishlab chiqiladi. Bunday mahsulotlarni shartli ravishda umumiy ahamiyatli mahsulotlarga, professional muhitlar yoki paketlarga, o'rgatuvchi tizimlarga, ko'ngilochar dasturlarga va hokazolarga ajratish mumkin.

Umumiy ahamiyatli dasturlardan foydalanuvchilarning turli guruhlari foydalanadi. Matnli muharrirlar, masalan WinWord, shuningdek, Excel tipidagi elektron jadvallar, grafik muharrirlar, umumiy ahamiyatli axborot tizimlari, masalan Moskva yoki Toshkent xaritalari, tarjimon-dasturlar va hokazolar ularga mansubdir.

*Professional mahsulotlar* turli sohalardagi mutaxassislar uchun mo'ljallangan va ularga quyidagilar mansub:

• turli texnik sohalarga yo'naltirilgan loyihalashni avtomatlashtirish tizimlari;

• trenajer (mashq) – tizimlar, masalan avariyali vaziyatda uchuvchilar harakatlari ishlanishi uchun trenajer;

• hisoblagich tizimlari, masalan 1C;

• noshirlik tizimlari, masalan PageMaker, QuarkXpress;

• professional grafik tizimlar, masalan Adobe Illustrator, PhotoShop, CorelDraw va hokazo;

• ekspert tizimlar va boshqalar.

*Ishlab chiqarish jarayonlarini avtomatlashtirish tizimlari* professional tizimlardan shunisi bilan farqlanadiki, ular yagona ishlab chiqarish jarayoni bilan bog'liq turli kasbdagi foydalanuvchilarga yo'naltirilgan.

*Ta'limiy dasturlar va tizimlar* o'z nomiga muvofiq ravishda ta'lim, masalan xorijiy tillar, yo'l harakati qoidalari va hokazolar ta'limi uchun mo'ljallangan.

*Ko'ngilochar tizimlarga* o'yinlar dasturlari, musiqiy dasturlar, axborot, biror ko'ngilochar mazmundagi testlari bor tizimlar, masalan munajjimlar bashorati va hokazolar mansubdir.

Gibrid tizimlar tizimli va amaliy dasturiy ta'minot alomatlarini o'zida biriktiradi. Qoidaga ko'ra, bular real vaqt tartibida turlicha tiplardagi texnologik jarayonlarni boshqarish uchun mo'ljallangan katta, lekin tor ixtisosli tizimlardir. Ishonchlilikni oshirish va ishlanish vaqtini kamaytirish uchun bunday tizimlarga, odatda operatsion tizimlar funksiyalari bajarilishini ta'minlovchi dasturlar kiritiladi.

Yuqorida sanalgan dasturiy ta'minot tiplarining har biriga ishlab chiqishdagi funksional talablar bilan birga, odatda muayyan tasarrufiy talablar ham qo'yiladi.

### 3.2. Dasturiy mahsulotlarga asosiy tasarrufiy talablar

1.6-mavzuda eslatib o'tilganidek, tasarrufiy talablar ishlab chiqilayotgan dasturiy ta'minotning bajarish jarayonida namoyon bo'luvchi ayrim tavsiflarni belgilaydi. Bunday tavsiflarga quyidagilar mansub:

• to‘g‘ri ishlashlilik – texnik topshiriq bilan muvofiqlikda ishlab turishi;

• universallik – har qanday imkonli ma’lumotlarda to‘g‘ri ishlashni va noto‘g‘ri ma’lumotlardan himoyani ta’minlash;

• ishonchlilik (to‘siqlardan himoyalanganlik) – natijalarning to‘liq takrorlanuvchanligini ta’minlash, ya’ni turli xildagi izdan chiqishlar mavjudligida ularning to‘g‘riligini ta’minlash;

• tekshiriluvchanlik – olingan natijalarni tekshirish imkoniyati;

• natijalar aniqligi – berilgandan yuqori bo‘lmagan natijalar xatoligini ta’minlash;

• himoyalanganlik – axborot konfidensialligini ta’minlash;

• dasturiy moslik – boshqa dasturiy ta’minot bilan birgalikda ishlash imkoniyati;

• samaradorlik – texnik vositalar resurslari, masalan mikroprotsessor vaqti yoki operativ xotira hajmidan mumkin bo‘lgan darajadagi minimal miqdoridan foydalanish;

• moslashuvchanlik – funksiyalar bajarishning o‘zgarayotgan shart-sharoitlariga moslashish maqsadida jadal modifikatsiyalanish imkoniyati;

• takroriy kirishlilik – diskdan qayta yuklatishsiz takroriy bajarish imkoniyati;

• rentabellik (reyentabellik) – bir necha jarayonlardan «parallel» foydalanish imkoniyati.

*To‘g‘rilik* har qanday dasturiy ta’minlash uchun majburiy talab hisoblanadi: texnik topshiriqda ko‘rsatilgan hamma narsa muqarrar ijobatlanishi shart. Biroq testlash ham, verifikatsiyalash ham yaratilgan dasturiy mahsulot to‘g‘riligini isbotlay olmasligini tushunish lozim. Shu munosabat bilan odatda muayyan xatolar mavjudligi ehtimoli to‘g‘risida gapiriladi. Tabiiyki, kompyuter tizimiga qancha ko‘p mas’uliyat yuklatilsa ham dasturiy, ham apparatli izdan chiqishlar ehtimoli shuncha kam bo‘lishi darkor. Masalan, atom elektrostansiyasini boshqarish tizimi uchun noto‘g‘ri ish ehtimoli nolga yaqin bo‘lishi shartligi ochiq-ravshan.

*Universallik* talablari ham odatda majburiy talablar guruhiga kiradi. Ishlab chiqilgan tizim nuqsonli (nokorrekt) ma'lumotlar uchun natija chiqarishining yoki ma'lumotlarning ayrim termasida o'z ishini buzilishli (avariyali) yakunlanishining hech qanday yaxshi tomoni yo'q. Biroq yuqorida eslatib o'tilganidek, nisbatan murakkab dasturning universalligini ham xuddi uning to'g'riligi kabi isbotlash mumkin emas, shu boisdan dasturning universalligi darajasi haqida so'z yuritish mohiyatan to'g'ri bo'ladi.

Amalda dasturiy ta'minotning to'g'riligi va universalligiga talablar qancha yuqori bo'lsa, uning ishonchliligiga talablar ham shuncha yuqori bo'ladi. Hisoblash jarayonining barcha qatnashchilari: texnik vositalar, dasturiy vositalar va odamlar to'siqlarning manbalari bo'lishlari mumkin. Texnik vositalar, masalan elektr kuchlanishining keskin sakrashlari yoki tarmoqlar bo'yicha axborotni berishdan to'siqlar tufayli izdan chiqishlarga moyil. Dasturiy ta'minotda xatolar bo'lishi mumkin. Odamlar esa boshlang'ich ma'lumotlarni kiritishda yangilanishlari mumkin.

Zamonaviy hisoblash qurilmalari hozirda ishonchli. Texnik vositalarning izdan chiqishlari, qoidaga ko'ra, apparatlarda qayd etiladi va ushbu holatda tegishli ravishda hisoblash natijalari tiklanadi. Uzoq muddatli hisoblarda, qoidaga ko'ra, oraliq natijalar saqlanadi (bu yo'l «nazorat nuqtalarini yaratish» nomini olgan), bu esa izdan chiqish sodir bo'lganda hisoblashni oxirgi nazorat nuqtasida yozilgan ma'lumotlar bilan davom ettirishga imkon beradi.

Tarmoqlar bo'yicha axborotni berish ham apparatlar orqali nazorat qilinadi, bundan tashqari ma'lumotlarni berishdagi xatolarni topish va tuzatishga imkon beruvchi maxsus to'siqdan himoyalovchi kodlashtirish ham odatda qo'llaniladi. Biroq texnik vositalarning xatolarini butunlay istisno qilish mumkin emas, shu boisdan ishonchlilikka talablar yuqori bo'lgan holatlarda, odatda tizimlar dublashtirishdan foydalaniladiki, bunda ikki tizim vaqti-vaqti bilan olingan natijalarni solishtirgan holda aynan bitta vazifani bajaradi.

Ko'pincha zamonaviy tizimlarning eng «ishonchsiz elementi» odamlar bo'ladi. Hozirda yaxshi ma'lumki, hisoblash moslamasi pultdagi monoton ish sharoitlarida operatorlar juda ko'p xatolarga yo'l qo'yishadi. Muayyan vaziyatlarda xatolar miqdorini kamaytirishga imkon beruvchi vositalar ham ma'lum. Jumladan, imkoni bo'lgan joyda ziyoda axborot kiritishdan foydalaniladi, bu kiritiluvchi ma'lumotlarning to'g'riligini tekshirish imkonini beradi. Bundan tashqari, axborotni kiritish emas, ayrim ro'yxatdan tanlash zarur bo'lganda turli xildagi ko'maklashuvlardan keng foydalaniladi va hokazo.

Hisoblashlar texnologik jarayonlar bilan parallel bajariladigan real vaqt tartibida funksiyalarni bajaruvchi boshqaruv tizimlari ishlab chiqilishida ishonchlilikka yuqori talablar qo'yiladi. Mazkur talab ilmiy-texnik tizimlar va ma'lumotlar bazalari uchun ham muhimdir.

Tekshiriluvchanlikni ta'minlash uchun olingan natijalarga ta'sir qiluvchi boshlang'ich ma'lumotlarni, o'rnatilgan tartiblar va boshqa axborotni hujjatli qayd etish lozim. Bu, ayniqsa, ma'lumotlar bevosita datchiklardan kelib tushadigan holatlarda g'oyatda ahamiyatlidir. Agar bunday ma'lumotlar natijalar bilan birga chiqarilmasa, natijalarni tekshirish mumkin bo'lmay qoladi.

Aniqlik yoki natijalar xatoligi kattaligi boshlang'ich ma'lumotlar aniqligiga, foydalanilayotgan model adekvatligi darajasi va tanlangan usul aniqligiga, shuningdek, kompyuterda operatsiyalar bajarishning xatoligiga bog'liq. Odatda natijalar aniqligiga talablar texnologik (masalan, kimyoviy) jarayonlarni boshqaruv tizimlari hamda navigatsiya tizimlari (masalan, kosmik apparatlarning ulashuvini boshqaruv tizimi) uchun eng qattiq bo'ladi.

Loyihalanayotgan tizim foydalaniladigan axborotning himoyalanganligini (konfidensialligini) ta'minlash alohida bo'lib, tarmoqlar mavjudligi sharoitlarida yetarlicha murakkabdir. Himoyalanganlikni ta'minlash uchun axborotni kodlashtirish va foydalanuvchi identifikatsiyalash kabi sof dasturiy himoya vosi-

talari bilan birga maxsus tashkiliy usullardan ham foydalaniladi. Davlat va tijorat sirlari bilan bog'liq axborot saqlanadigan tizimlarga eng qattiq talablar qo'yiladi.

Dasturiy mutanosiblikning talablari ko'rsatilgan dasturiy ta'minotni birgalikda o'rnatish imkoniyatidan tortib, to u bilan o'zaro harakatini ta'minlashgacha bo'lgan o'zgarishni tushunish mumkin, masalan ma'lumotlar almashinuvi va h.k. Hammasidan ko'ra ko'proq dasturiy ta'minotning ishlashi mazkur operatsion tizim boshqaruvi ostida bo'lishini ta'minlashga to'g'ri keladi. Biroq qandaydir dasturdan ma'lumotlar olishni yoki unga ayrim ma'lumotlarning berilishini ko'zda tutish talab qilinishi mumkin. Ushbu holatda berilayotgan ma'lumotlar formatlarini aniq ta'kidlab ko'rsatish zarur.

Apparatli mutanosiblik talablari asosan dasturiy ta'minot ishlaydigan uskunaning minimal konfiguratsiyasi tarzida shakllantiriladi. Agar nostandart uskunadan foydalanish nazarda tutilayotgan bo'lsa, uning uchun axborot bilan almashinuv interfeyslari yoki protokollari ko'rsatilishi shart. Bu borada Windows sinfidagi operatsion tizimlar uchun qurilmaning operatsion tizim bilan o'zaro harakatlanishini ta'minlovchi drayver-dasturlar mavjud bo'lmagan qurilmalar nostandart hisoblanadi.

Tizim samaradorligi hisoblash moslamasining har bir resursi bo'yicha alohida baholanadi. Ko'pincha quyidagi mezonlardan foydalaniladi:

- tizim javobi vaqti (foydalanilayotgan uskuna tez hakarakatlanishiga mansub bo'luvchi) – foydalanuvchi bilan interaktiv tartibda o'zaro ishlovchi tizimlar va real vaqt tizimlari uchun;
- operativ xotira hajmi – operativ xotiraning hajmi cheklangan tizimlarda ishlovchi mahsulotlar uchun masalan MS DOS;
- tashqi xotira hajmi – tashqi xotiradan, masalan ma'lumotlar bazalaridan intensiv foydalanadigan mahsulotlar uchun;
- xizmat ko'rsatiluvchi tashqi qurilmalar miqdori – tashqi qurilmalar, masalan datchiklar bilan intensiv o'zaro harakatlanishni amalga oshiruvchi mahsulotlar uchun.

Samaradorlik talablari bir-biriga qarama-qarshi bo'lishi mumkin. Masalan, dasturning ayrim fragmenti bajarilish vaqtini kamaytirish uchun operativ xotiraning qo'shimcha hajmi talab qilinishi mumkin.

Moslashuvchanlik, mohiyatga ko'ra, dasturiy ta'minotning texnologik sifatini baholaydi, shu boisdan ushbu tavsifni miqdoriy baholash amalda mumkin emas. Mahsulotni yaratishda uning modellashtirilishini yengillashtiruvchi texnologiyalardan va maxsus usullardan foydalanilganini qayd etish mumkin.

Takroriy kirishlilik talablari operativ xotiraga rezidentli yuklatilgan dasturiy ta'minotga, masalan drayverlarga qo'yiladi. Mazkur talabni ta'minlash uchun dasturni shunday tashkillashtirish kerakki, uning boshlang'ich ma'lumotlari bajarish jarayonida o'chirilib ketmasligi yoki har bir chaqiruv boshlanishida yoxud tugallanishida tiklanishi kerak.

Rentabellik talablari takroriy kirishlik talablariga ko'ra yanada qattiqroq, zero ushbu holatda bajarish jarayonida dastur o'zgartiruvchi barcha ma'lumotlar maxsus blokka ajratilishi shart va uning nusxasi dasturni chaqirishidagi har bir jarayon uchun yaratiladi.

Ko'plab dasturiy tizimlarning murakkabligi ularga bo'lgan aniq talablarni darhol shakllantirishga imkon bermaydi. Odatda qandaydir dasturiy ta'minotni yaratish g'oyasidan texnik topshiriqqa kiritilishi mumkin bo'lgan talablarning aniq shakllantirilishga o'tishi uchun ishlab chiqish sohasida loyihadan oldingi tadqiqotlarni bajarish zarur bo'ladi.

### 3.3. Predmetli sohada loyihadan oldingi tadqiqotlar

Loyiha oldi tadqiqotlarning maqsadi bo'lg'usi biror narsaga mo'ljallangan dasturiy ta'minot haqidagi umumiy noravshan bilimlarni unga nisbatan aniq talablarga aylantirishdan iborat.

Noaniqlikning ikki varianti mavjud:

- shakllantirilayotgan vazipredmeti hal etish usullari noma'lum, odatda ilmiy-texnik vazifalarni hal etishda shunday tipdagi noaniqliklar yuzaga keladi;

• avtomatlashtiriluvchi axborot jarayonlarning tuzilmasi noma'lum, odatda korxonalar boshqaruvining avtomatlashgan tizimlarini barpo etishda uchraydi.

Birinchi holatda loyihadan oldingi tadqiqotlar vaqtida qo'yilgan vazipredmeti hal etish imkoniyati va talab qilinuvchi natijani olishga imkon beradigan usullar belgilanadi, bu esa ham fundamental, ham amaliy mazmundagi tegishli ilmiy-tadqiqotlarni, real dunyo obyektlari yangi modellarining ishlab chiqilishi hamda tadqiq qilinishini talab etishi mumkin.

Ikkinchi holatda quyidagilar belgilanadi:

• avtomatlashtiruvchi va axborot jarayonlarining tuzilmasi hamda o'zaro aloqasi;

• inson va tizim o'rtasidagi, shuningdek apparatlar va dasturiy ta'minot o'rtasidagi funksiyalar taqsimoti;

• dasturiy ta'minot funksiyalari: uning ham foydalanuvchilar bilan, ham zaruratga ko'ra apparatlar bilan birga ishlashining tashqi shart-sharoitlari va uni interfeysining xususiyatlari;

• dasturiy va axborot komponentlariga talablar, zarur apparat resurslari, ma'lumotlar bazalariga talablar hamda dasturiy komponentlarning fizik tavsifnomalari.

Predmetli sohaning loyihadan oldingi tadqiqotlari natijalaridan texnik topshiriqni ishlab chiqish jarayonida foydalaniladi.

### 3.4. Texnik topshiriqni ishlab chiqish

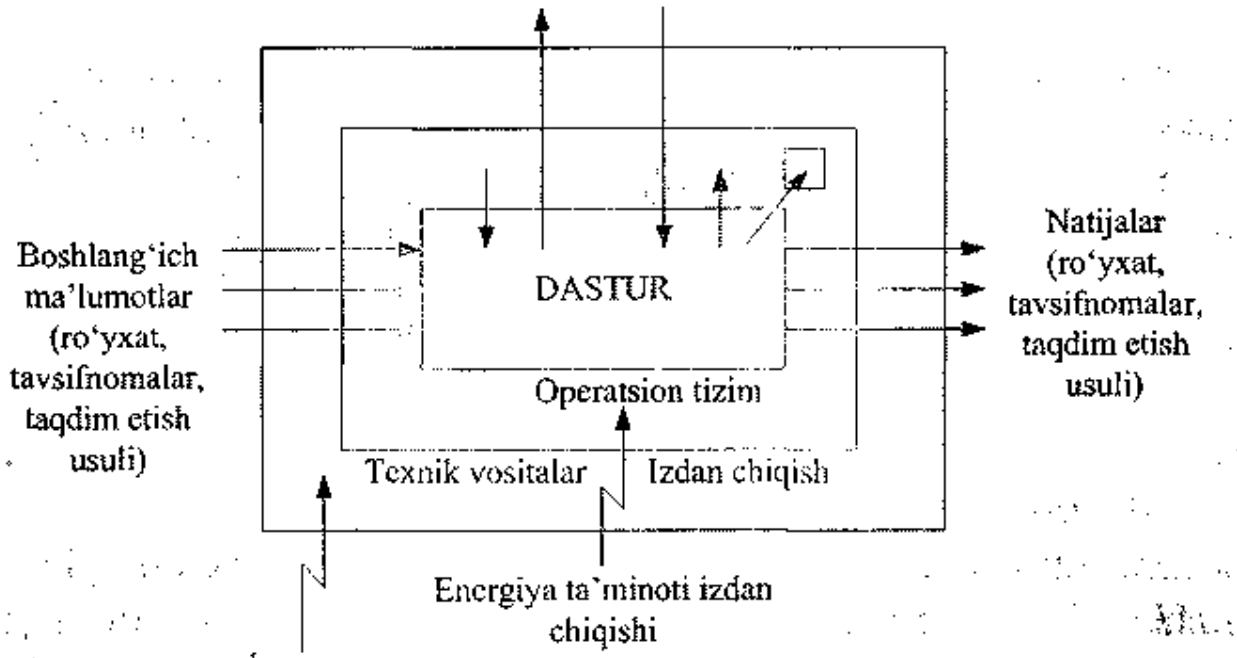
Texnik topshiriq ishlab chiqishning asosiy maqsadlari, dasturiy mahsulotga talablar shakllantirilgan, ishlab chiqish muddatlari va bosqichlari belgilangan hamda qabul, topshirish sinovlari jarayoni reglamentlangan hujjatni o'zida ifoda etadi. Texnik topshiriqni ishlab chiqishda buyurtmachi vakillari ham, ijrochi vakillari ham ishtirok etishadi. Ushbu hujjatning asosida buyurtmachining boshlang'ich talablari, texnika ilg'or yutuqlarining tahlili, ilmiy-tadqiqot ishlarining, loyihadan oldingi tadqiqotlarning, ilmiy bashorat (prognozlash)ning natijalari va hokazolar turadi.

3.2-rasmda ishlab chiqiluvchi dasturiy ta'minot tavsiflarini belgilovchi omillar sxematik ravishda ko'rsatilgan. Bunday omillar quyidagilardir:

• dastur yoki tizim funksiyalarini belgilaydigan boshlang'ich ma'lumotlar va talab qilinuvchi natijalar;

• funksiyalar bajarish muhiti (dasturiy va apparatli) yuklatilishi ham, texnik topshiriqda ko'rsatilgan parametrlarni ta'minlash uchun tanlanishi ham mumkin;

• boshqa dasturiy ta'minot yoki maxsus texnik vositalar bilan mumkin bo'lgan o'zaro harakatlanishi ham, bajariluvchi funksiyalar majmuidan kelib chiqqan holda tanlanishi ham mumkin.




**3.2-rasm.** Ishlab chiqilayotgan dasturiy ta'minot parametrlarini belgilovchi omillar.

Texnik topshiriqni ishlab chiqish quyidagicha davomiylikda bajariladi. Eng avvalo, bajariluvchi funksiyalar majmui tayinlanadi, so'ngra natijalar ro'yxati, ularning tavsifnomalari va taqdim etish usullari belgilanadi. Keyin esa dasturiy ta'minot funksiyalari bajarilishining muhiti aniqlanadi; texnik vositalarning muayyan komplektlashuvi va parametrlari, foydalaniluvchi operatsion tizim versiyasi hamda bo'lg'usi dasturiy mahsulot o'zaro harakatlanishi

joiz bo'lgan boshqa o'rnatilgan dasturiy ta'minlashning versiyalari va parametrlari belgilanadi.

Ishlab chiqilayotgan dasturiy ta'minot ayrim axborotni to'playdigan va saqlaydigan yoki boshqaruvga qandaydir texnik jarayon bilan qo'yiladigan hollarda ham uskuna, ham energiya ta'minoti izdan chiqishlari holatlarida dastur ishlashini aniq reglamentlash zarur.

Texnik topshiriq uchun GOST 19.201-78 «Texnik topshiriq. Mazmun va rasmiylashtirishga talablar» standarti mavjud. Ushbu standartga muvofiq texnik topshiriq quyidagi bo'limlardan iborat bo'lishi shart:

- kirish;
- ishlab chiqish uchun asos;
- ishlab chiqish ahamiyati;
- dasturga yoki dasturiy jihozga talablar;
- dasturiy hujjatlarga talablar;
- texnik-iqtisodiy ko'rsatkichlar;
- ishlab chiqish davrlari (stadiya) va bosqichlari;
- nazorat va qabul tartibi.

Zaruratga ko'ra texnik topshiriqqa ilovalar qo'shishga yo'l qo'yiladi.

Har bir bo'lim mazmunini yanada batafsil qarab chiqamiz.

*Kirish* dastur yoki dasturiy mahsulot nomini va ularning qo'llanish sohasi qisqacha tavsifnomasini, shuningdek ulardan foydalanish ko'zda tutilayotgan obyekt (masalan tizim) nomi hamda tavsifnomasini qamrab olishi shart. Kirishning asosiy ahamiyati tegishli ishlab chiqish dolzarbligini namoyish etish va ushbu ishlab chiqish, o'xshash ishlab chiqishlar qatorida qanday o'rin egallashini ko'rsatish.

Ishlab chiqish uchun asos bo'limi ishlab chiqish yuritilishiga asos hujjat va ushbu hujjatni tasdiqlagan tashkilot nomini hamda ishlab chiqish mavzusining nomini yoki shartli ifodasini o'z ichiga olishi darkor. Reja, buyruq, shartnoma va hokazolar asos hujjat tariqasida xizmat qilishi mumkin.

Ishlab chiqish ahamiyati foydalanuvchilar toifalari ko'rsatilgan holda dasturiy mahsulot funksional va tasarrufiy ahamiyatining bayonini qamrab olishi shart.

Dasturga yoki dasturiy jihozga talablar bo'lishi quyidagi tag bo'limlarni o'z ichiga olishi darkor:

- funksional tavsiflarga talablar;
- ishonchlilikka talablar;
- ishlatish sharoitlari;
- texnik vositalar tarkibiga va parametrlariga talablar;
- axborot va dasturiy mutanosiblikka talablar;
- rusumlashtirish va o'ramlashtirishga talablar;
- transportlashtirish va saqlashga talablar;
- maxsus talablar.

Yuqorida sanalganlar qatorida funksional tavsiflarga talablar tag bo'limi eng muhimdir. Ushbu bo'limda bajariluvchi funksiyalar sanalishi va boshlang'ich ma'lumotlar hamda natijalaning tarkibi, tavsifnomalari, taqdim etish shakllari bayon etilishi shart. Xuddi shu bo'limda zaruratga ko'ra samaradorlik mezonlari: tizim javobining maksimal mumkin bo'lgan vaqti, foydalaniluvchi operativ yoki tashqi xotira hajmi va boshqalar ko'rsatiladi.

**Eslatma.** Agar ishlab chiqilgan dasturiy ta'minlash texnik topshiriqda ko'rsatilgan funksiyalarni bajara olmasa, u texnik topshiriqqa nomuvofiq, ya'ni sifat mezonlari nuqtayi nazaridan noto'g'ri hisoblanadi. Bo'lg'usi loyihaning universalligi ham odatda maxsus ta'kidlab o'tilmaydi, lekin ko'zda tutiladi.

Ishonchlilikka talablar tag bo'limida ishlab chiqilayotgan tizim ta'minlashi shart bo'lgan ishonchlilik darajasi va izdan chiqishdan so'ng tizimning tiklanish vaqti ko'rsatiladi. Tizimlar uchun ishonchlilikka odatiy talablar bilan birga ushbu bo'limda ba'zan ishlab chiqilayotgan mahsulotning natijalar ishonchliligini orttirish bo'yicha harakatlari (kirish va chiqish axboroti nazorati, oraliq natijalarning zaxira nusxalarini yaratish va hokazo) reglamentlanadi.

Ishlatish sharti tag bo'limda ishlatish shartlariga qo'yiladigan maxsus talablar: atrof-muhit haroratiga, havoning nisbiy namli-

giga va hokazolarga alohida talablar ko'rsatiladi. Qoidaga ko'ra, agar ishlab chiqilayotgan tizim nostandart shart-sharoitlarda tasarruf qilinadigan yoki maxsus tashqi qurilmalardan, masalan axborotni saqlash uchun qurilmadan foydalaniladigan bo'lsa, shunday talablar qo'yiladi. Xuddi shu joyda xizmat ko'rsatish turi, xodimlarning zaruriy miqdori va malakasi ko'rsatiladi. Aks holda talablar qo'yilmayotganligini ko'rsatishga ruxsat etiladi.

**Texnik vositalarning tarkibi va parametrlariga talablar** tag bo'limida texnik vositalarning zaruriy tarkibi ularning quyidagi asosiy texnik tavsiflari aniqlanadi: mikroprotsessor tipi, xotira hajmi, tashqi qurilmalar mavjduligi va boshqalar qayd etilgan holda ko'rsatiladi, bunda ko'pincha konfiguratsiyaning ikki varianti: minimal hamda tavsiya etiluvchi konfiguratsiya ko'rsatiladi.

Axborot va dasturiy mutanosiblikka talablar tag bo'limida zaruratga ko'ra hal etish usullarini berish, ishlab chiqish uchun dasturlash tilini yoki muhitni belgilash, shuningdek foydalaniluvchi operatsion tizimni hamda ishlab chiqilayotgan dasturiy ta'minot o'zaro harakatlanishi shart bo'lgan boshqa tizimiy va foydalanuvchilik dasturiy vositalarini ham belgilash mumkin. Xuddi shu bo'limda zaruratga ko'ra axborot himoyasining qanday darajasini ko'zda tutish zarurligi ham ko'rsatiladi.

Dasturiy hujjatlarga talablar bo'limida dasturchi qo'llanmasi, tushuntirish yozuvnomasi va boshqalar mavjudligining zarurligi ko'rsatiladi. Hujjatlarning ushbu barcha tiplariga ham GOSTlar mavjud. Ularni tuzish qoidalari 15-bobda ko'rib chiqilgan.

**Texnik-iqtisodiy ko'rsatkichlar** bo'limida taxminiy iqtisodiy samaradorlik, ko'zda tutilayotgan yillik ehtiyoj va mavjud analoglarga qiyosan iqtisodiy afzalliklar ko'rsatilishi tavsiya etiladi.

Ishlab chiqish stadiyalari va bosqichlari bo'limida ishlab chiqish stadiyalari, bosqichlari va yillar mazmuni ishlab chiqish muddatlari hamda ijrochilar ko'rsatilgan holda bo'ladi.

Nazorat va qabul tartibi bo'limida sinovlar turlari, ish qabuliga umumiy talablar ko'rsatiladi.

Ilovalarda zaruriyatga ko'ra quyidagilar keltiriladi: ishlab chiqishni asoslovchi ilmiy-tadqiqot ishlari ro'yxati; ishlab chiqishda foydalanish lozim bo'lgan algoritmlar sxemalari, jadvallar, bayonlar, asosnomalar, hisob-kitoblar va boshqa hujjatlar.

Ishlab chiqilayotgan mahsulot o'ziga xosliklariga bog'liq ravishda bo'lishlar mazmunini aniqlashtirishga, ya'ni tag bo'limlardan foydalanishga, yangi bo'limlar kiritishga yoki ularni birlashtirishga ruxsat etiladi.

Texnik topshiriqda nazarda tutilgan qandaydir talablarni buyurtmachi qo'ymagan holatlarda tegishli joyda «Talablar qo'yilmadi» deb ko'rsatish lozim.

Texnik topshiriqni ishlab chiqish – muayyan ko'nikmalarni talab qiluvchi sermehnat jarayon qoidaga ko'ra, asosiy bo'limlarni: kirish, dasturiy mahsulot ahamiyati va dasturiy mahsulotga talablarni aniq shakllantirish eng murakkabdir. Misol tariqasida qisqartirilgan sxema bo'yicha tuzilgan kurs loyihalashini bajarishga ikkita texnik topshiriqni hamda davlat budjetidagi ilmiy-tadqiqot ishini bajarishga nisbatan to'liq texnik topshiriqni ko'rib chiqamiz.

### 3.4.1. Texnik topshiriqni ishlab chiqish bo'yicha misollar

**3.1-misol.** Bitta $y = f(x)$ argumenti funksiyalarining grafiklarini o'quvchilarga ko'rgazmali namoyish etish uchun mo'ljallangan dasturiy mahsulotga texnik topshiriq ishlab chiqilsin. Ishlab chiqiluvchi dastur ifodalar jadvalini hisoblashi, berilgan formula bo'yicha berilgan bo'lakda funksiyalar grafigini tuzishi hamda argument qadami va bo'lak chegaralarini o'zgartirishi shart. Bundan tashqari, dastur kiritilgan formulalarni eslab qolishi darkor.

3.3-rasmda o'quv dasturiy mahsulotga texnik topshiriqning titul varag'i namunasi taqdim etilgan. Quyida uning matni keltirildi.

O‘zbekiston Respublikasi aloqa, axborotlashtirish va telekommunikatsiya texnologiyalari Davlat Qo‘mitasi
Toshkent Axborot Texnologiyalari Universiteti
Kompyuter injiniring fakulteti
Informatika asoslari kafedrasi

f-m.f.d., prof. __SH.A. Nazirov

TASDIQLAYMAN
IA kafedrasi mudiri
«__»_____2014-y.

# FUNKSIYALAR JADVALLARI VA GRAFIKLARINI TUZISH DASTURI

*Kurs ishiga texnik vazifalar*
*5 varaq*

Rahbar:
t.f.nomzodi, dasent_____Rahmanov Q.S.

Ijrochi:
№ guruh talabasi_____

2014-yil

**3.3-rasm.** O‘quv dasturiy mahsulotga texnik vazifalarning titul varag‘ini to‘ldirish namunasi.

1. KIRISH

Mazkur texnik topshiriq bitta o'zgaruvchanning funksiyalari grafiklarini va ifodalari jadvallarini tuzishning yuqori sinf o'quvchilari tomonidan foydalanish uchun mo'ljallangan dasturini ishlab chiqishga joriy qilinadi.

Elementar algebra bo'yicha maktab kursida funksiyalar tahlili mavzusi eng murakkab mavzulardan biridir. Mazkur mavzuni o'rganishda o'quvchilar funksiyaning barcha ma'lum tavsifiy nuqtalaridan ildizlarni, birinchi va ikkinchi xildagi uzilish nuqtalarini qo'shgan holda foydalanib, bitta o'zgaruvchanning funksiyalari grafiklarini tadqiq qilishga hamda tuzishga o'rganishlari shart.

Bunday vazifalarni hal eta olishi mumkin bo'lgan mavjud dasturiy ta'minlash, masalan Eurica yoki MathCad universal hisoblanadi. U minimum oliy matematikaning institut kursini tinglagan foydalanuvchilarga mo'ljallangan, nisbatan murakkab foydalanuvchilik interfeysiga ega, bu esa shunday vositalardan maktab o'quvchilari foydalanishini imkonsiz qilib qo'yadi.

Ishlab chiqilayotgan dastur maktab o'quvchilariga ko'rsatilgan mavzuni o'rganishda o'z bilimlarini tekshirish uchun imkon beradi.

2. ISHLAB CHIQISH UCHUN ASOS

Dastur «Kompyuter tizimlari va tarmoqlari» kafedrasining o'quv rejasiga asosan va kafedraning ... -sonli maktab bilan 2009.5.09 dagi shartnomasiga muvofiq ishlab chiqilmoqda.

Dasturning asosiy vazifasi elementar algebra maktab kursining «Bitta argument funksiyasini tadqiq etish» bo'limini o'rganishda o'quvchilarga yordam berishdan iborat.

### 3.5. Dastur yoki dasturiy mahsulotga nisbatan talablar

#### 3.5.1. Funksional tavsiflarga talablar

3.1.1. Dastur quyidagi funksiyalarni bajarish imkoniyatini ta'minlashi lozim:

- bitta o'zgaruvchan va uzoq saqlash funksiyasining tahliliy tasavvurini kiritish;

• intervalni kiritish va funksiyani aniqlash intervalini o‘zgartirish;

• argumentlarni kiritish va tuzatish;

• berilgan intervalda funksiyalar ahamiyatining jadvalini yoki funksiyalar grafigini tuzish.

3.1.2. Dastlabki ma’lumotlar:

• funsiyalarning tahliliy vazifalari;

• funksiyalar topshiriqlarining intervali;

• intervaldagi nuqtalar sonini belgilovchi argumentlar o‘zgarishining qadami.

### 3.5.2. Ishonchlilikka talablar

3.2.1. Kiritiladigan axborot nazoratini ko‘rib chiqish.

3.2.2. Tizim bilan ishlashda foydalanuvchining noto‘g‘ri xatti-harakatlarini blokirovkalashni ko‘rib chiqish.

### 3.5.3. Texnik vositalar tarkibi va parametrlariga talablar

3.3.1. Tizim personal kompyuterlarga joylashtiriladigan IBM da ishlashi lozim.

3.3.2. Minimal konfiguratsiya:

• protsessor turi _____Pentium va undan yuqori;

• tezkor xotiraga ega qurilma hajmi ___32Mb va undan ko‘p.

### 3.5.4. Axborot va dasturiy muvofiqlikka nisbatan talablar

Tizim Win 32 (Windows 95, Windows 98, Windows 2000, Windows NT, Windows XP, Windows 7 va hokazo) operatsion tizim oilasining boshqaruvida ishlashi kerak.

### 3.5.5. Dasturiy hujjatlashtirishga nisbatan talablar

3.1. Ishlab chiqiladigan dasturiy modullar o‘zini-o‘zi hujjatlashtirish, ya’ni dastur matnlari barcha kerakli izohlarni o‘zida saqlashi lozim.

3.2. Ishlab chiqiladigan dastur matematika bo‘limiga mos keluvchi asosiy atamalar to‘g‘risidagi ma’lumotnoma axborotini o‘z ichiga olishi kerak.

3.3. Hujjatlashtirish tarkibiga quyidagilar kirishi kerak:

3.3.1. Ishlanma bayon etilgan 25–30 varaqdan iborat izohlovchi matn.
3.5.3.2. Foydalanuvchi uchun qo'llanma.

### 3.5.6. Loyihalashtirish boshlang'ich bosqichlarining prinsipial yechimlari

Loyihalashtirishning boshlang'ich bosqichlarida ushbu jarayonni, ishlanma sifati va murakkabligini belgilovchi muhim qarorlar qabul qilinishi lozim. Bu quyidagi qarorlardir:

- dasturiy ta'minot arxitekturasini tanlash;
- foydalanuvchining interfeys turini va hujjatlar bilan ishlash texnologiyasini tanlash;
- ishlanmaga tuzilmaviy va obyektiv yondashuvni tanlash;
- dasturlash tili va muhitini tanlash;

Boshqacha aytganda, bu qarorlar qanday iste'mol tavsifi va qanday vositalar bilan loyihalanishini belgilab beradi.

Qarorlarning bir qismi texnik vazifalarda, texnologik talablar guruhini tashkil etgan holda belgilanishi, qolganlari esa loyihalashtirish jarayoni uchun boshlang'ich ma'lumotlar bo'lgani sababli oldinroq qabul qilinishi lozim.

**Dasturiy ta'minot arxitekturasini tanlash.** *Dasturiy ta'minot arxitekturasi* deb, uning tuzilishining bazaviy konsepsiyalari majmuiga aytiladi. U yengiladigan vazifalar murakkabligi, ishlab chiqiladigan dasturiy ta'minot universalligi darajasi va uning bir nusxasi bilan ishlovchi foydalanuvchilar soni bilan belgilanadi va quyidagilarni farqlaydi:

- bitta foydalanuvchi arxitekturasini.

Bunda dasturiy ta'minot personal kompyuterda ishlovchi bitta foydalanuvchiga mo'ljallangan:

- lokal yoki global tarmoqda ishlashga mo'ljallangan ko'p foydalanuvchi arxitekturasi.

Bundan tashqari bitta foydalanuvchi arxitekturasi quyidagilarni farqlaydi:

- dasturlar;
- dasturlar paketi;

- dasturiy komplekslar;
- dasturiy tizimlar.

Ko'p foydalanuvchili arxitektura «mijoz-server» tamoyili bo'yicha tuzilgan tizimlarni amalga oshiradi.

*Dastur* deb aniq bir masalani yechish uchun bajarilish ketma-ketligi kompyuterda aniq va ravshan bayon etilgan instruksiya (yo'riqnoma)lar to'plamiga aytiladi. Dastur tuzilmali (strukturali) yondashuvda aniq bir masalani yechish jarayonida bir-birini chaqiruvchi dastur ostilari (tag dastur (dastur qismi)lari) iyerarxiyasini namoyon etadi, obyektli yondashuvda esa, realizatsiya uchun maxsus yaratilgan sinflardan iborat obyektlarning xabarlar bilan almashinuvi majmuasidan iborat. Dastur bunday holatda alohida jamlanuvchi dasturiy birlikni namoyon etadi. U standart kutubxona dastur ostisidan foydalanishi mumkin, ammo o'zinikini tashkil etmaydi. Bu eng sodda arxitektura turi bo'lib, odatda kichik vazifalarni yechishda foydalaniladi.

*Dasturlar paketi* ayrim amaliy sohada vazifalarni hal etuvchi dasturlar majmuidan iborat. Masalan, grafik dasturlar paketi, matematik dasturlar paketi. Bunday paketlar dasturi bir-biri bilan ma'lum bir amaliy sohaga mansubligi bilan bog'liq bo'ladi. Dasturlar paketi alohida dasturlar to'plami sifatida amalga oshiriladi. Ularning har biri kerakli ma'lumotlarni kiritadi va natijalarni chiqaradi. Mohiyatan, dasturlar paketi bu dasturning o'ziga xos kutubxonasidir.

*Dasturiy komplekslar* dasturlar majmuidan iborat bo'lib, birgalikda biror amaliy sohada murakkab vazifalarning kichikroq sinfini hal etishni ta'minlaydi.

Bunday masalalarni hal etish uchun ketma-ket komplekslar dasturini chaqirgan hamda bir nechta vazipredmetini bajarishga to'g'ri kelishi mumkin. Dasturiy kompleksda dasturni chaqirish maxsus dastur — dispetcher orqali amalga oshiriladi. U foydalanuvchi bilan murakkab bo'lmagan interfeys orqali bog'lanadi va ayrim ma'lumotnomalarni taqdim etishi mumkin. Dasturiy kompleks dasturlar paketidan yana shunisi bilan farqlanadiki, bir

nechta dastur navbati bilan yoki davriy shaklda bitta vazipredmeti hal etish uchun chaqirilishi mumkin.

*Dasturiy tizimlar* ayrim amaliy sohada keng ko'lamdagi vazifalarni hal etish imkonini beruvchi dasturlar (tizim osti)ning uyushgan majmuini ifodalaydi. Dasturiy komplekslardan farqli ravishda dasturiy tizimga kiruvchi dasturlar umumiy ma'lumotlar orqali o'zaro harakat qiladi. Dasturiy tizimlar odatda, rivojlangan foydalanuvchi va ichki interfeysga ega bo'ladi. Bu ularni puxta loyihalashni talab etadi.

*Ko'p foydalanuvchi dasturlar tizimi* odatdagi dasturiy tizimlardan farqli ravishda dasturiy ta'minotning alohida komponentlari tarmoq ta'sirini tashkil etishi lozim. Bu uni ishlab chiqish jarayonini yanada murakkablashtiradi. Bunday dasturiy ta'minotni ishlab chiqish uchun maxsus texnologiya yoki CORBA, COM, Java va hokazo texnologiyalar platformasidan foydalaniladi.

**Foydalanuvchi interfeysi turini tanlash.** Foydalanuvchi interfeysining to'rtta turi farqlanadi. Bular:

- *primitiv (sodda)* – yagona ish ssenariysini amalga oshiradi, masalan, ma'lumotlarni kiritish – natijalarni chiqarish;
- *menyu* – ko'plab ish ssenariylarini amalga oshiradi. Ularning operatsiyalari iyerarxik strukturada tashkil etilgan. Masalan, «qo'yish», «faylni qo'yish», «simvolni qo'yish» va hokazo.
- *erkin navigatsiyali* – ko'plab ssenariylarni amalga oshiradi. Ularning operatsiyalari iyerarxiya darajalariga bog'liq va ma'lum bir ish bosqichida ko'plab operatsiyalarni belgilab beradi. Bunday shakl interfeyslari asosan Windows-ilovadan foydalanadi:
- *bevosita manipulyatsiyalash* – obyektlar ustidagi operatsiyalarda taqdim etilgan ko'plab ssenariylarni amalga oshiradi. Asosiy operatsiyalar «sichqon» obyektlari piktogrammasi ko'chishi bilan amalga oshiriladi. Bu shakl Windows operatsiya tizimining interfeysida, erkin navigatsiyali interfeysga muqobil ravishda amalga oshirilgan.

Foydalanuvchi interfeys turi ishlanmaning murakkabligi va ko'p mehnat talab etishini belgilab beradi. Oxirgi ma'lumotlarga ko'ra, dasturiy kodning 80 foizigacha aynan foydalanuvchining interfeysi amalga oshirilishi mumkin. Shu sababli dasturlashni

o‘rganishning boshlang‘ich bosqichida, asosan, sodda interfeys va menyulardan foydalaniladi, garchi ular foydalanuvchi uchun noqulay bo‘lsa ham. Dasturiy vositalarni ishlab chiqishning obyektga yo‘naltirilgan vizual muhitini paydo bo‘lishi muhim ahamiyatga ega bo‘ladi. Bunday yo‘l dasturlashda hodisali yondashuvga asoslangan va erkin navigatsiyali interfeyslarni yaratishga mo‘ljallangan bo‘lib, monand interfeyslarni yaratish qiyinchiliklarini yetarlicha kamaytirdi hamda to‘g‘ridan to‘g‘ri manipulyatsiya qilish interfeyslarini realizatsiyasini soddalashtirdi.

Shunday qilib, oxirgi ikkita interfeys turini tanlash dasturiy ta’minotni ishlab chiqishning vizual muhitlaridan biridan foydalanishni nazarda tutadi.

Bundan tashqari, interfeys turini tanlash o‘z ichiga hujjatlar bilan ishlash texnologiyasini tanlashni oladi. Ikkita texnologiya farqlanadi:

- bitta hujjatli interfeys (SDI – Single Document Interface) ni mo‘ljallaydi;
- ko‘p hujjatli – ko‘p hujjatli interfeysga (MDI – Multiple Document Interface) mo‘ljallangan.

Ko‘p hujjatli texnologiya agar, dasturiy ta’minot bir vaqtning o‘zida bir necha hujjatlar bilan ishlashi shart bo‘lsa (masalan, bir necha matn yoki tasvir bilan) foydalaniladi. Bitta hujjatli texnologiyada esa bir vaqtning o‘zida bir necha hujjatlar bilan ishlash shart emas.

Zamonaviy kutubxonadan foydalangan holda ko‘p hujjatli interfeysni amalga oshirish ish hajmi birinchisiga qaraganda 3–5% ko‘p. Interfeys turini tanlash ish hajmiga jiddiy ta’sir ko‘rsatadi.

**Ishlanmaga yondashuvni tanlash.** Agar erkin navigatsiyali yoki bevosita manipulyatsiyalovchi interfeys tanlangan bo‘lsa, yuqorida qayd etilganidek, bu vaziyatga ko‘ra dasturlash va obyektiv yondashuvdan foydalanishni nazarda tutadi. Chunki Visual C++, Delphi, Bulder C++ kabi zamonaviy vizual dasturlash muhiti aynan kutubxona sinfi obyektlari ko‘rinishidagi interfeys komponentlarini taqdim etadi. Bunda predmet sohaning murakkabligiga bog‘liq ravishda dasturiy ta’minot obyektlar (mos ravishda sinflar)

dan foydalanilgan holda yoki faqat protsedurali holdan foydalanib realizatsiya qilinishi mumkin. Istisno tariqasida Perl ga o'xshash Internet-qo'llanmani maxsus tillarini yaratishning natijasini ko'rsatish mumkin. Bu til umuman olganda boshqacha tamoyilga asosan qurilgan.

Sodda interfeys va menyuga o'xshash interfeys ham strukturali, ham ishlanmaga obyektli yondashuvga muvofiq keladi. Shu bois yondashuvni tanlash qo'shimcha axborotdan foydalanish orqali amalga oshiriladi.

Tajriba shuni ko'rsatadiki, obyektiv yondashuv juda katta dasturiy tizimni (dasturlash universal tilining 100 000 dan ortiq operatori) ishlab chiqishda hamda predmet sohasining obyekt strukturasi yorqin ifodalanganda samaralidir.

Shuni ham hisobga olish kerakki, ishlab chiqiladigan dasturiy ta'minot samarasi qat'iy cheklanganda obyektli yondashuvdan ehtiyotkorona yondashish kerak.

Boshqa hamma holatlarda yondashuvni tanlash foydalanuvchi ixtiyoriga bog'liq.

**Dasturlash tilini tanlash.** Ko'pgina hollarda dasturlash tilini tanlash yuzasidan hech bir muammo mavjud emas. Til quyidagilar tomonidan belgilanishi mumkin:

- ishlanmani olib boruvchi tashkilot tomonidan. Masalan, agar firma C++Builder litsenziya variantiga ega bo'lsa, u ko'proq mazkur muhitda ishlanmani olib boradi;
- har doim imkoniyatga qarab, yaxshi tildan foydalanuvchi dasturchi tomonidan;
- qat'iy fikr asosida («barcha bunday ishlanmalar C++ da yoki Java yoki ... da bajarilishi kerak») va hokazo.

Agar, til tanlash imkoni bo'lsa dasturlashning barcha mavjud tillarini quyidagi guruhlarga bo'lish mumkinligini hisobga olish kerak:

- yuqori darajadagi universal til;
- dasturiy ta'minotni ishlab chiquvchining ixtisoslashgan tillari;
- foydalanuvchining ixtisoslashgan tillari;
- quyi darajadagi tillar.

Universal tillarning yuqori darajasi guruhida so'zsiz ravishda bugungi kunda C (C++ bilan birga) dasturlash tili yetakchi sanaladi. Darhaqiqat C va C++ning turli versiyalari juda muhim afzalliklarga ega:

- ko'p platformali – bugungi kunda barcha foydalanuvchilar uchun C va C++ tilli kompilyatorlar mavjud;
- asosiy tuzilmaviy algoritmli konstruksiyalarni (shartli qayta ishlash, sikllarning barcha turi) amalga oshiruvchi operatorlar mavjudligi;
- tezkor xotira manzillaridan foydalangan holda quyi (tizimli) darajada dasturlash imkoniyatini;
- dastur osti va sinflarga oid katta kutubxonalar.

Bular bari C va C++ ning operatsiya tizimini yaratish uchun foydalaniladigan asosiy tillarga aylantirdi. Biroq C va C++ jiddiy kamchiliklarga ega:

- ma'lumotlarni to'laqonli ichki tuzilmali (strukturali) turlarining yo'qligi (adresli (manzilli) arifmetikada foydalanadigan psevdotuzilmasi (psevdostrukturali) ma'lumotlarning mavjudligi bu ma'lumotlar ustida ko'p amallarni nazorat qilish uchun yetarli darajada qat'iy aniqlangan. Bu esa o'z navbatida faqat dasturni rostlash jarayonida ko'p sondagi xatoliklarni aniqlashga olib keladi);
- sintaktik jihatdan bir qiymatlilik emasligini (turli xillikning) mavjudligi. Bu ham kompilyatorga dastur to'g'riligini nazorat qilish imkonini bermaydi;
- tag dastur (dastur qismi)ga uzatiladigan parametrlarning cheklangan nazorati. Bu faqat dasturni yopish jarayonida aniqlanadi va hokazo.

Amaliy dasturlash ta'minotini yaratish uchun foydalaniladigan universal dasturlash tillari o'rtasida C va C++ ga muqobil bu Pascal sanaladi. Uning sintaksisi aniq bo'lgani bois kompilyatorlari sintaksisdan tashqari ko'plab semantik xatolarni ham aniqlaydi. Delphi muhitida foydalangan Object Pascal versiyasi professional kutubxona yordamida kuzatib boriladi. Bu katta ishlanmalarni kiritishni osonlashtiradi va Windows ilovasini yaratish uchun Delphini yetarlicha

samarali muhit bilan ta'minlaydi. Bu tillardan tashqari universal guruhga Basic, Modula, Ada va ayrim tillar kiradi. Ular ham C++ va Pascal singari o'zining qo'llanish sohasiga hamda xususiyatlariga ega.

Ishlanma yaratuvchining ixtisoslashgan tillaridan dasturiy ta'minotning konkret turlarini yaratish uchun foydalaniladi. Ularga quyidagilar kiradi:

- ma'lumotlar bazalari tillari;
- tarmoq ilovalarni yaratish tillari;
- sun'iy intellekt tizimini yaratish tillari va h.k.

Bu tillar maxsus kurslarda o'rganiladi va mazkur darslikda ko'rib chiqilmaydi.

**Foydalanuvchining ixtisoslashgan tillari** odatda foydalanuvchi professional muhitining bir qismi sanaladi. U tor yo'nalishli bo'lib, dasturiy ta'minotni ishlab chiquvchi tomonidan foydalanilmaydi.

**Quyi darajadagi tillar** dasturlashni amalda mashina komandalari darajasida amalga oshirish imkonini beradi. Bunda ham bajarish vaqti nuqtayi nazaridan, ham dasturning zarur xotirasi hajmi nuqtayi nazaridan eng maqbul yo'l tanlanadi. Biroq bu tillar katta dasturlarni yaratish uchun yaramaydi. Quyi darajadagi tillar strukturali dasturlash tamoyillariga ega emas. Bu ishlab chiqiladigan dastur texnologiyasini yomonlashtiradi.

Hozirda Assembler tipidagi tillardan quyidagi holatlarda foydalaniladi:

- texnik vositalar bilan (masalan, drayver) bevosita o'zaro ta'sir ko'rsatuvchi oddiy dasturlarni yozishda, chunki bu holatda tegishli jihozni ter to'kib sozlash kerak bo'ladi.

Yuqori darajadagi dasturlash tillar ustunligi muhim bo'lmaydi:

- yuqori darajadagi tillarda dasturga qo'shimcha vosita shaklida, masalan, katta miqdordagi takrorlash bilan sikllarda ma'lumotlar hosil bo'lishini tezlashtirish uchun.

**Dasturlash muhitini tanlash.** *Dasturlash muhiti* deb dasturiy kompleksga aytiladi. U o'z ichiga ixtisoslashgan matnli muharrir, kompilyator, komponovlovchi, ma'lumotnoma tizimi va boshqa dasturlashlarni oladi.

So'nggi vaqtlarda yuqorida qayd etilgan vizual dasturlash keng tarqaldi. Bunga ko'ra dasturchi komponentlarning maxsus kutubxonalaridagi ayrim kodlar dasturiga vizual bog'lanish imkoniga ega bo'ladi.

Borland firmasining (Inprige Corpation) Delphi, C++ Builder, Microsoft firmasining Visual C++, Visual Basic va IBM firmasining Visual Ada vizual muhitlaridan nisbatan ko'p foydalanilmoqda.

Ushbu firmalarning Delphi, C++ Builder va Visual C++ asosiy vizual muhitlari o'rtasida jiddiy farq mavjud. Microsoft firmasining vizual muhitlari «Windows» ostida dasturlashning past darajasini ta'minlaydi. Bu ularning ham yutug'i, ham kamchiligi sanaladi. Yutug'i shundan iboratki, u «nostandart» vaziyatlar yuzaga kelish ehtimolini kamaytiradi. Kamchiligi esa bu Delphi yoki C++ bilan ishlovchi dasturchini eskicha ishga undaydi. Shuningdek, «quyi darajali dasturlash» ga mo'ljallangan Visual C++ interfeysi ham ko'plab norozilikka sabab bo'ladi.

Umuman olganda, agar gap ushbu muhitlardan birini tanlash ustida ketsa, u ko'proq loyiha tavsifi bilan belgilanadi.

**Ishlanma standartlarini tanlash yoki shakllantirish.** Har qanday loyihalash texnologiyasini qo'llash barcha loyiha ishtirokchilari rioya etishi shart bo'lgan qator standartlar shakllantirish yoki tanlashni talab etadi.

- loyihalashtirish standarti;
- loyiha hujjatlashtirishini rasmiylashtirish standarti;
- foydalanuvchi interfeysi standarti.

*Loyihalashtirish standarti* quyidagilarni aniqlashi lozim:

- loyihalashning har bir bosqichida zarur modellarni (sxema, diagramma) tanlash va ularni detallashtirish darajasi;
- diagrammalardagi loyiha qarorlarini qayd etish qoidasini, shuningdek, obyektlar va atamalar bo'yicha kelishuvni nomlash, barcha obyektlar uchun atributlar to'plami va har bir bosqichda ularni to'ldirish qoidasi, diagrammani rasmiylashtirish qoidasi, jumladan obyektlar shakli va o'lchamiga bo'lgan talablar;

• ishlanma muammolarining ish o'rni konfiguratsiyasiga bo'lgan talablar jumladan operatsiya tizimi va foydalanayotgan CASE vositasini sozlash;

• loyiha ustida hamkorlikda ishlashni ta'minlash mexanizmi, jumladan, loyiha tizim ostini integratsiyalash qoidasi hamda loyiha qarorlari tahlili.

*Loyiha hujjatlarini rasmiylashtirish standarti* quyidagilarni rejalashtirishi lozim:

• har bir bosqichda hujjatlashtirish jamlanmasi, tarkibi va strukturasi;

• uni saqlab turish va rasmiylashtirishga nisbatan talablar;

• hujjatlarni tayyorlash, ko'rib chiqish, kelishish va tasdiqlash qoidalari.

*Foydalanuvchi interfeysining standarti* quyidagilarni aniqlashi lozim:

• ekranlarni rasmiylashtirish qoidasi (shriftlar va rang palitrasi), derazalar va boshqaruv elementi tarkibi hamda joylashuvi;

• klaviatura va «sichqon»dan foydalanish qoidasi;

• yordam ko'rsatish matnlarini rasmiylashtirish qoidasi;

• standart xabarlar ro'yxati;

• foydalanuvchi reaksiyasini qayta ishlash qoidasi.

Yuqorida bayon etilgan loyiha qarorlari ishlanmaning ish hajmi va murakkabligiga ta'sir etadi. Faqat ularni qabul qilgandan so'ng loyihalashtirishning dasturiy ta'minotiga o'tish zarur.

**Nazorat savollari:**

1. Dasturiy ta'minot texnologiyasi deganda nima tushuniladi? Nima uchun?

2. Dasturiy mahsulotning qanday tiplarini ajratib ko'rsatish mumkin? Ular qanday farqlanadi?

3. Dasturiy mahsulotga nisbatan asosiy ekspluatatsiya talablarini aytib bering. Ularning har biri qanday vositalar va uslublar bilan ta'minlangan? Dasturiy tizimning qanday turlari uchun ularning har birini ko'rsatish maqsadga muvofiq?

# 4. TUZILMAVIY YONDASHUVDA DASTURIY TA'MINOT TALABLARINI TAHLIL ETISH VA IXTISOSLIGINI ANIQLASH

Har qanday dasturiy ta'minotni ishlab chiqish bo'lajak dasturiy mahsulotga nisbatan talablarni tahlil qilishdan boshlanadi. Tahlil natijasida ishlab chiqilayotgan dasturiy ta'minot spetsifikatsiyasi olinadi. Hal etiladigan vazipredmeting dekompozitsiyasi va mazmun jihatdan qo'yilishi amalga oshiriladi, ularning o'zaro harakati va ekspluatatsiya cheklashi aniqlanadi. Umuman olganda spetsifikatsiyasini aniqlash jarayonida predmet sohasining umumiy modeli quriladi, ishlab chiqilayotgan dasturiy ta'minotning real dunyoning ayrim qismi bilan yoki bu usul bilan o'zaro ta'siri aniqlanadi va uning asosiy funksiyalari konkretlashtiriladi.

## 4.1. Tuzilmaviy yondashuvda dasturiy ta'minot spetsifikatsiyasi

1.5-mavzuda ta'kidlanganidek, spetsifikatsiya ishlab chiqilayotgan dasturiy ta'minot funksiyasining to'liq va aniq bayonini ifodalaydi. Bunda spetsifikatsiyaning bir qismi ishlab chiqilayotgan dasturiy ta'minot funksiyasini bayon qiladi. Ikkinchi qismi esa, texnik vositalarga ishonchlilikka, axborot xavsizligiga va h.k. bo'lgan talablarni belgilaydi.

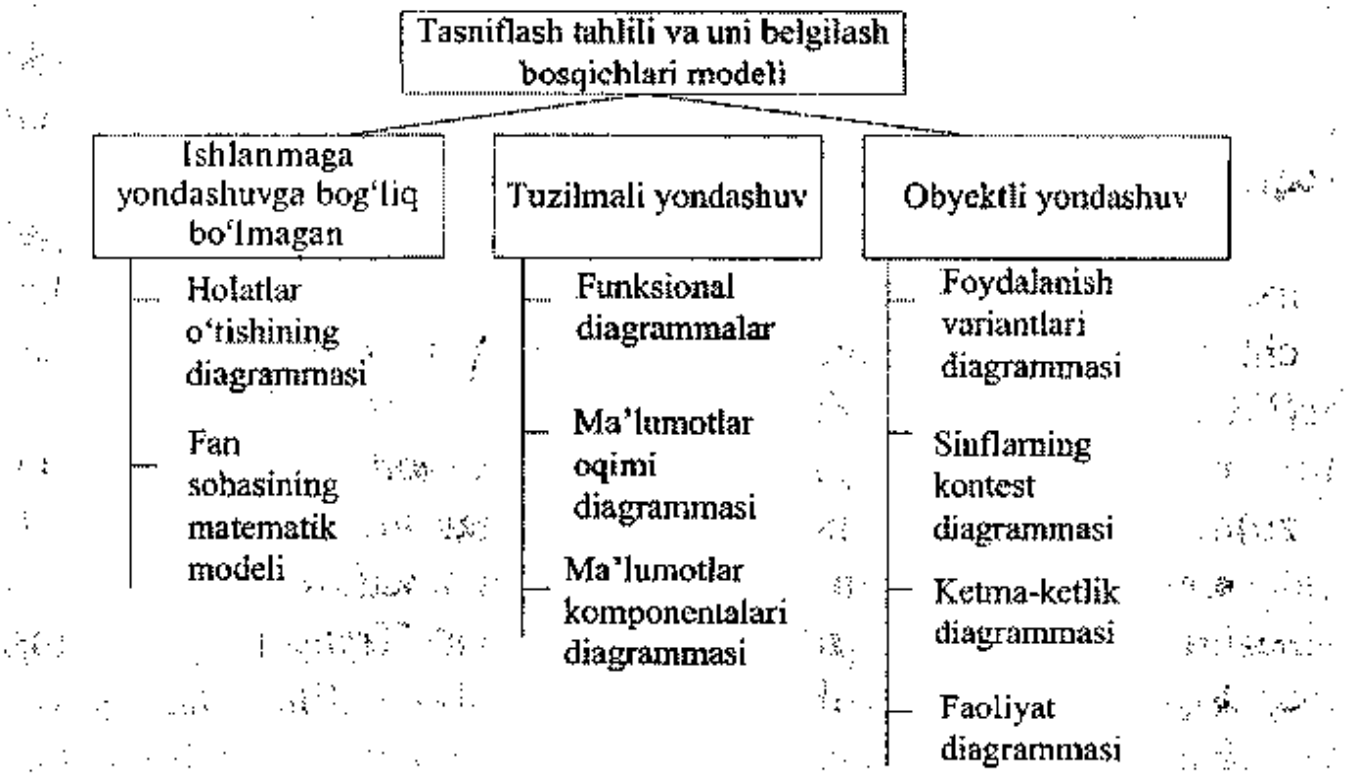
Bu aniqlangan ta'rif spetsifikatsiyaga nisbatan asosiy talablarni aks ettiradi. Funksiyaviy spetsifikatsiyaga nisbatan quyidagilar aniqlanadi:

- to'liqlikka nisbatan talab shuni anglatadiki, spetsifikatsiya barcha muhim axborotni saqlashi, unda hech bir muhim ma'lumot o'tkazib yuborilmasligi, muhim bo'lmagan axborot bo'lmasligi kerak. Masalan, ishlanma muallifiga nisbatan samarali qarorlarni tanlashda xalaqit bermasligi lozim;
- aniqlik talabi shuni anglatadiki, spetsifikatsiya ham buyurtmachi, ham ishlanma muallifi tomonidan bir xilda qabul qilinishi shart.

Oxirgi talabni bajarish ancha qiyin. Chunki, hatto tabiiy tilda batafsil spetsifikatsiyalash kerakli aniqlikni ta'minlamaydi. Aniq spetsifikatsiyani faqat ishlab chiqilayotgan dasturiy ta'minotning ayrim formal modelini ishlab chiqibgina aniq spetsifikatsiyani aniqlash mumkin.

Spetsifikatsiyani aniqlash bosqichida foydalaniladigan formal modellarni ikkita guruhga ajratish mumkin: ishlanmaga yondashuvga bog'liq (strukturaviy yoki obyektga-mo'ljallangan) modellar va unga bog'liq bo'lmagan modellar. Ishlab chiqilayotgan dasturiy ta'minot xususiyatlarini namoyish etuvchi holatlar o'tishining diagrammasi u yoki bu signallarni olishda (4.2-mavzuga qarang) va predmet sohasining matematik modeli (4.6-mavzuga qarang) ishlanmaga har qanday yondashuvda foydalaniladi.

Spetsifikatsiyani tahlil etish va aniqlashga tuzilmaviy yondashuv doirasida modellarning uchta turidan foydalaniladi: funksiyaga mo'ljallangan model, ma'lumotlarga va ma'lumotlar oqimiga mo'ljallangan model. Har bir modeldan dasturiy ishlanmaning o'z maxsus sinfi uchun foydalanish maqsadga muvofiqdir.




**4.1-rasm.** Tasniflashni aniqlash bosqichida foydalaniladigan, ishlab chiqilayotgan dasturiy ta'minot klassifikatsiyasi.

4.1-rasmda spetsifikatsiyani aniqlash bosqichida foydalaniladigan dasturiy ta'minot modellari tasniflanishi keltirilgan.

Shuni hisobga olish kerakki, barcha funksional spetsifikatsiya ishlab chiqilayotgan dasturiy ta'minotning bir xil tavsifini bayon etadi. Bular: funksiyalar ro'yxati va qayta ishlanadigan ma'lumotlar. Ular spetsifikatsiya talablarini tahlil etish va aniqlash jarayonida ishlanma muallifi foydalanadigan tizimli ustuvorlari bilan farqlanadi.

Holatlarning o'tish diagrammasi dasturiy ta'minot vaziyatining asosiy aspektlarini, ma'lumotlar oqimi diagrammasi ma'lumotlar oqimining yo'nalishi va strukturasini, sinflarning konseptual diagrammasi esa predmet sohasidagi asosiy tushunchalar munosabatini belgilab beradi.

Chunki, turli modellar loyihalashtirilayotgan dasturiy ta'minotni turli tarafdan bayon etadi. Bunda bir vaqtning o'zida bir necha modellardan foydalanish va ularda matn, lug'at, izoh va tegishli diagrammani tushuntirib beruvchi boshqa vositalarni qo'llash tavsiya etiladi.

Ma'lumotlar oqimini modellashtirishga asoslangan tuzilmali (strukturali) tahlil va loyihalashtirish metodologiyasi odatda, loyihalashtirilayotgan dasturiy ta'minot kompleks tasavvuridan modellar majmuasi shaklida foydalanadi:

- tizimda amalga oshirilishi kerak bo'lgan jarayonlar orqali axborot manbai va iste'molchilarning o'zaro ta'sirini bayon etuvchi ma'lumotlar oqimi manbai va iste'molchilar diagrammasi (DFD – Data Flow Diagrams);
- ishlab chiqilayotgan tizimning ma'lumotlar bazasi (4.4-mavzu)ni bayon etuvchi «mohiyat-aloqa» diagrammasi;
- tizimning ma'lum vaqtdagi holatini tavsiflovchi (4.5-mavzu) vaziyatlar o'zgarishi diagrammasi (STD – State Transition Diagrams);
- jarayonlar spetsifikatsiyasi;
- atamalar lug'ati.

Bu xilda boyitilgan modellar elementlarining o'zaro aloqasi 4.2-rasmda ko'rsatilgan.

**Jarayonlar spetsifikatsiyasi.** Jarayonlar spetsifikatsiyasi odatda qisqacha matn bayoni sxemasi, psevdokodlar, Flow-forma yoki Nassi-Shneyderman diagrammasi shaklida bo'ladi. Chunki, jarayonlar bayoni ishlanma mutaxassisiga ham, buyurtmachiga ham birday tushunarli va qisqa bo'lishi lozim. Ularni tavsivlash uchun ko'pincha psevdokodlardan foydalaniladi.

**Atamalar lug'ati.** Atamalar lug'ati spetsifikatsiyalash jarayonida foydalaniladigan asosiy tushunchalarning qisqacha bayonidan iborat bo'ladi. U predmet sohasidagi asosiy tushunchalarni aniqlash ma'lumotlar elementlari tuzilmasi, ularning turlari va shakli bayoni, shuningdek, barcha qisqartmalar hamda shartli belgilarni o'z ichiga olishi lozim. Atamalar lug'ati predmet sohasini tushunish darajasini oshirish va buyurtmachi hamda ishlanma ijrochilari o'rtasida modellar muhokamasi borasida qarama-qarshilik yuzaga kelishining oldini olish uchun mo'ljallangan.

Odatda, lug'atda atamalar bayoni quyidagicha sxema bo'yicha bajariladi:

- atama;
- kategoriya (predmet sohasi tushunchasi, ma'lumotlar bazalari, shartli belgilar va h.k.)
- qisqacha bayon.

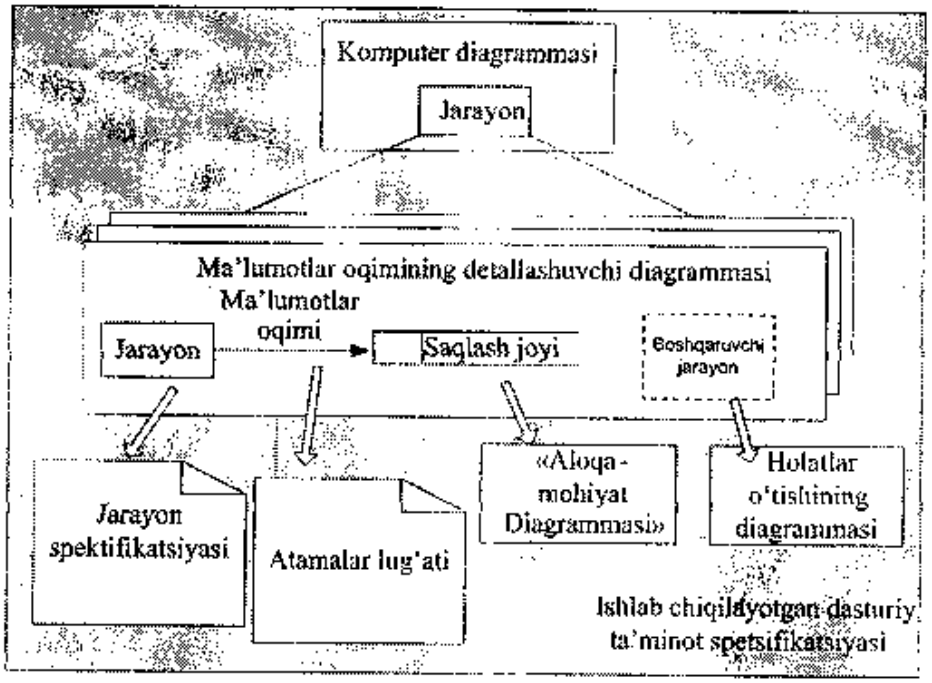
Misol tariqasida kombinator-optimizatsiyali masalalarni yechish tizimining atamalardan birining bayonini keltiramiz:

Atama.....Algoritm

Kategoriya.....Predmet sohasi tushunchasi

Bayon.....Mazkur loyihada «tanlangan usul bilan aniq bir masalani hal etish amallarini bajarish»ning to'liq tushunchasini belgilash uchun foydalaniladi.

Ko'rsatilgan modellardan tashqari to'liq spetsifikatsiya tarkibiga har qanday yondashuvda predmet sohasi obyektlari bayonining matematik modellari ham kiradi. Ular tahlil etilayotgan o'lchamlar va shu o'lchamlarga qo'yilgan cheklovlarning asosiy nisbatini aniqlash imkonini beradi. Sanab o'tilgan modellarni batafsil ko'rib chiqishga o'tamiz.




**4.2-rasm.** Dasturiy ta'minotni strukturaviy tahlil etish va ma'lumotlar oqimiga asoslangan loyihalashtirish metodologiyasining to'liq spetsifikatsiyasi elementlari.

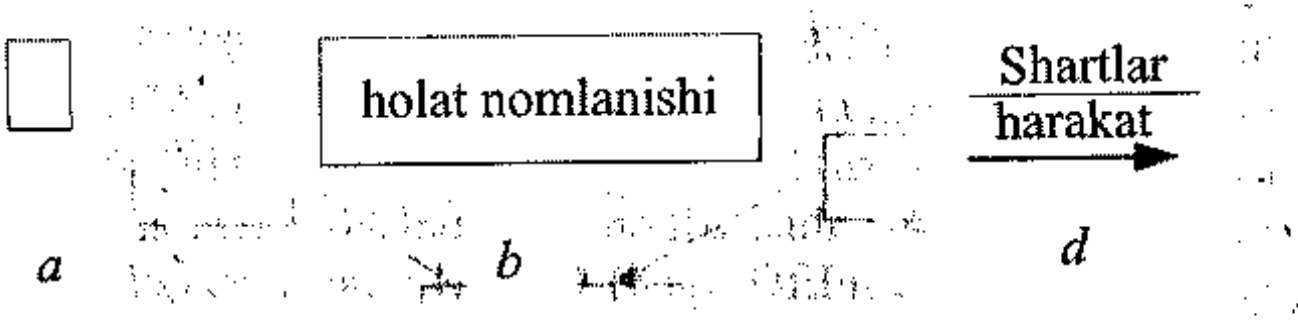
## 4.2. Holatlarning o'tish diagrammasi

*Holatlarning o'tish diagrammasi* – chekli avtomat, ya'ni matematik abstraksiyani taqdim etishning grafik shakli sanaladi. Undan texnik obyekt yoki real dunyo obyektlarining determinallashgan xatti-harakatini modellashtirish uchun foydalaniladi. Spetsifikatsiya talablarini tahlil etish va belgilash bosqichida holatlar o'tishining diagrammasi boshqaruvchi ta'sirni olishda ishlab chiqiladigan dasturiy tizim xatti-harakatini namoyon qiladi. Boshqaruvchi ta'sir yoki signallar deganda, tizim orqali olinadigan boshqaruvchi ma'lumot tushuniladi. Masalan, foydalanuvchining komandalari va kompyuter tizimiga ulangan datchik signallari boshqaruvchi harakat sanaladi. Ana shunday boshqaruvchi ta'sir olgach, ishlab chiqilayotgan tizim ma'lum bir harakatni amalga oshirishi yoki tashqi ta'sir bilan o'zaro ta'sirlashib, boshqa holatga o'tishi yoki o'sha holatda qolishi mumkin.

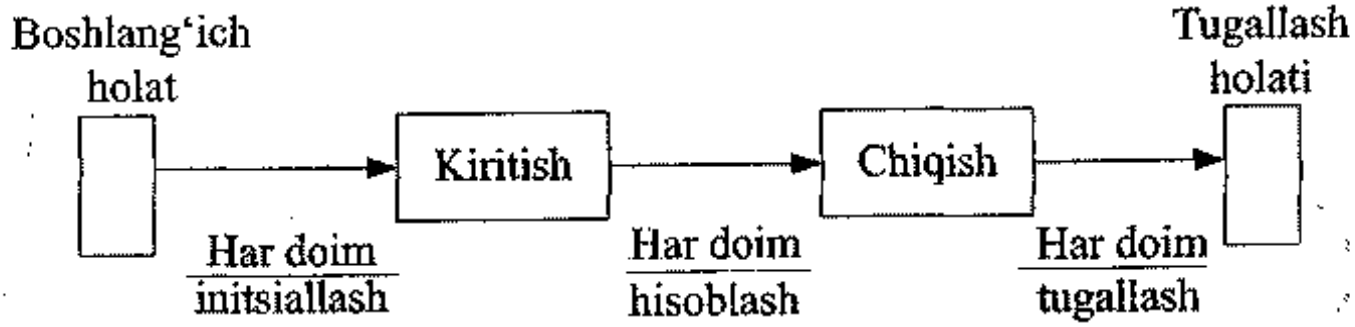
Holatlarning o'tish diagrammasini tuzish uchun yakuniy avtomatlar nazariyasiga ko'ra quyidagilarni aniqlab olish lozim:

asosiy holat, boshqaruvchi ta'sir (yoki o'tish shartlari), bajariladigan xatti-harakat va bir holatdan boshqasiga o'tish variantlari, holatlarning o'tish diagrammasini tuzishda foydalaniladigan shartli belgilar 4.3-rasmda ko'rsatilgan.

Agar dasturiy tizim ishlash jarayonida atrof-muhit (foydalanuvchi yoki datchiklar) bilan o'zaro faol harakat qilmasa, masalan, sodda interfeysdan foydalansa va ayrim hisob-kitoblarni berilgan boshlang'ich ma'lumotlar bo'yicha amalga oshirsa, holatlarning o'tish diagrammasi odatda hech qanday qiziqish uyg'otmaydi. Bu holda u faqat ketma-ket bajariladigan o'tish amallarini namoyon qiladi. Ya'ni, boshlang'ich holatdan ma'lumotlarni kiritish holatiga o'tadi, so'ng hisob-kitob qilingach chiqish holatiga, va nihoyat ishni yakunlash holatiga o'tiladi (4.4-rasm).




**4.3-rasm.** Holatlar o'tish diagrammasining shartli belgilari:
a – terminal holat, b – oraliq holat, d – o'tish.

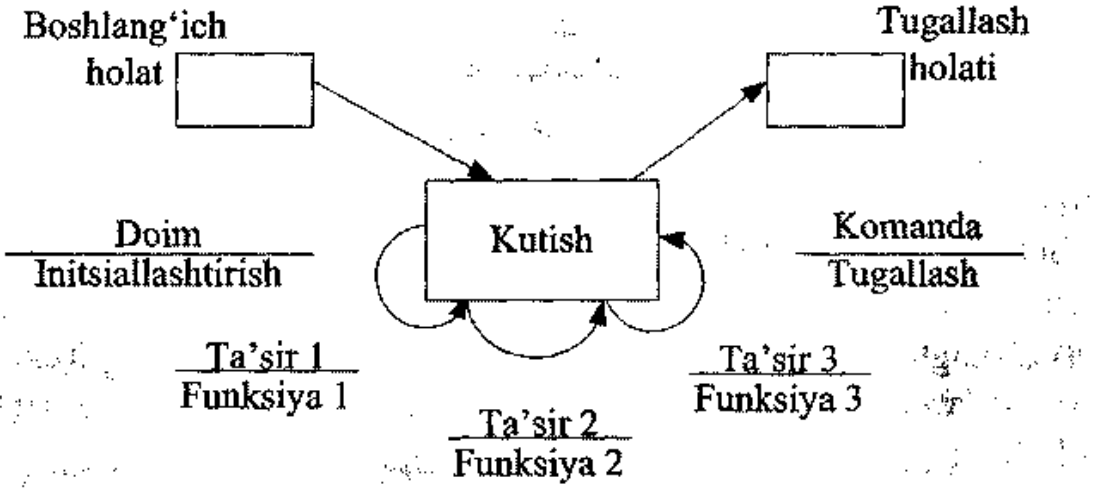



**4.4-rasm.** Atrof-muhit bilan o'zaro faol harakatlanmaydigan dasturiy ta'minot holatining o'tish diagrammasi.

Rivojlangan interfeysga ega interaktiv dasturiy faoliyat uchun asosiy boshqaruvchi harakat bu foydalanuvchining buyruqlari, aniq vaqtdagi dasturiy ta'minot uchun – datchiklar va (yoki) ishlab chiqarish jarayoni operatoridan keladigan signallardir. Inter-

aktiv dasturiy ta'minot uchun turli xil (4.5-rasm), agar bu aniq vaqtdagi dasturiy ta'minot bo'lsa, bir turdagi signallar qabul qilish xos xususiyatidir.

Interaktiv tizimlardan farqli ravishda, joriy vaqt tizimi uchun dasturiy ta'minotning olingan signalni qayta ishlashi uchun qat'iy cheklangan vaqt belgilangan. Bunday cheklashlar masalan, Petri tarmog'i yoki Markov jarayonidan foydalanilgan holda tizim harakati bo'yicha qo'shimcha tekshirishlarni o'tkazishga majbur qiladi. Holatlarning o'tish diagrammasini tuzish vositasi orqali xatti-harakatlar xususiyatini aniqlashni talab etuvchi dasturiy ta'minotga tarmoqda ishlashga mo'ljallangan turi ham kiradi (1.2-mazu). Bu holatda server va mijoz o'rtasida uzatiladigan ma'lumotlarni tasavvur etgan holda boshqaruvchi ta'sir ko'rinishida ular xatti-harakatining modeli alohida tuziladi.




**4.5-rasm.** Tashqi muhit bilan o'zaro faol ta'sir ko'rsatuvchi dasturiy ta'minot holatining o'tish diagrammalari namunasi.

Dastur interfaol sinfga mansub bo'lib, spetsifikatsiyalashni tahlil etish va belgilash bosqichiga muvofiq foydalanuvchi bilan interfeys darajasida dastur mohiyatini aniqlash maqsadga muvofiqdir.

Holatlar o'tishining olingan diagrammasini dasturiy ta'minot buyurtmachisi bilan kelishib olish zarur.

## 4.3. Funksional diagrammalar

Birinchi galda ishlab chiqiladigan dasturiy ta'minot funksiyalarining o'zaro aloqasini aks ettiruvchi diagrammalar funksiyaviy deb yuritiladi.

Funksiyaviy modelga misol tariqasida faol modelni ko'rib chiqamiz. U SADT (Structured Analysis and Design Technique – strukturaviy tahlil va loyihalashtirish) funksiyaviy modellashtirish metodologiyasi tarkibida D. Ross tomonidan taklif etilgan.

**Eslatma.** SADT metodologiyasi shuni nazarda tutadiki, u tizim funksiyalariga yoki uning predmetlariga (ma'lumotlari, jihozlar, axborotlar va h.k.) asoslanadi. Har ikki holatda o'xshash grafik notatsiyalar qo'llaniladi. Ammo birinchi holatda blok funksiyalarga, ikkinchisida ma'lumotlar elementiga mos keladi. Tegishli modellarni faol modellar va ma'lumotlar modellari deb atash qabul qilingan. To'liq model o'z ichiga dasturiy ta'minotning nisbatan to'liq bayonini ta'minlovchi har ikki model tuzilishini oladi, biroq faol (funksional) modellar keng qo'llaniladi. SADT metodologiyasi asosida keyinchalik IDEFO (Icam DEFinition – ICAM notatsiyasi) murakkab tizim bayonining mashhur metodologiyasi yaratilgan. U ICAM (Integrated Computer-Aided Manufacturing – integrallashgan kompyuter ishlab chiqarish) dasturining asosiy qismi sanaladi. Dastur AQSH harbiy havo kuchlari tashabbusi bilan o'tkaziladi.

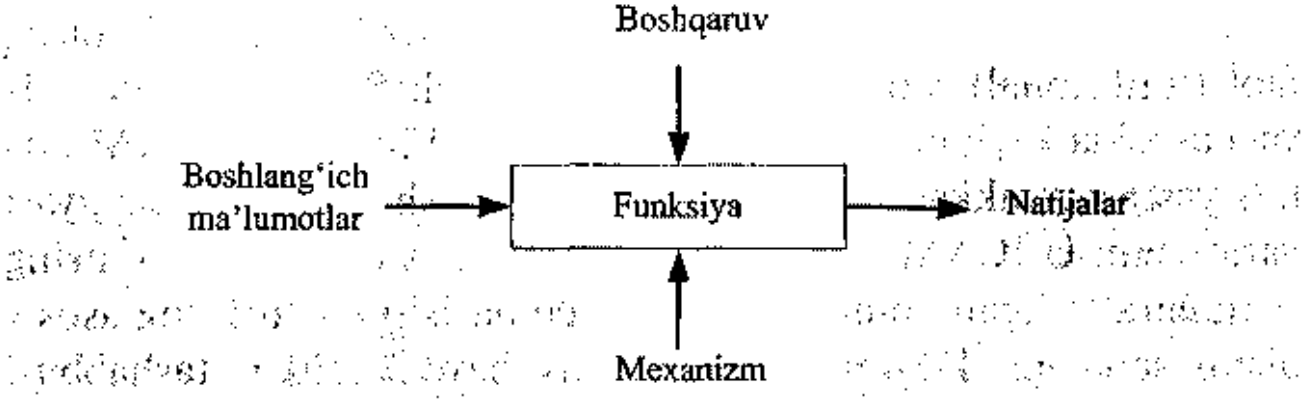
Faollik modeli funksiyalarining o'zaro aloqasini aks ettirish bir necha funksiyaviy diagramma o'zaro aloqasini sxematik taqdim etuvchi funksiyaviy diagramma iyerarxiyasini tuzish vositasida amalga oshiriladi. Bunday diagrammaning har bir bloki ayrim funksiyalarga mos keladi. Unda boshlang'ich ma'lumotlar, natijalar, boshqaruvchi axborotlar va uni amalga oshiruvchi mexanizmlar, ya'ni odam yoki texnik vositalar aniqlab berilishi lozim.

Yuqorida qayd etilgan funksiyalarning barcha aloqalari yoy bilan ko'rsatib beriladi. Ayni paytda aloqa turi va uning yo'nalishi qat'iy reglamentlangan.

Har bir aloqa turini tasvirlovchi yoy blokka ma'lum bir tomondan yondashishi (4.6-rasm), aloqa yo'nalishi esa yoy oxirida strelka bilan ko'rsatilishi kerak.

Boshlang'ich ma'lumotlarning yoylari, natijalari va boshqaruvi funksiyalar o'rtasida o'zaro uzatiladigan ma'lumotlar to'plamini taqdim etadi. Funksiyalarni amalga oshirish mexanizmini belgilovchi yoylar asosan, murakkab axborot tizimining spetsifikatsiyasini bayon etishda foydalaniladi. Mazkur tizim ham avtomatlashgan, ham qo'lda bajariladigan operatsiyalarni o'z ichiga oladi. Bloklar va yoylar tabiiy tilda matnlar bilan belgi qo'yiladi (markirovkalanadi).

Diagrammada bloklar ularning ish ketma-ketligiga yoki bir blokning ikkinchisiga ko'rsatadigan ta'siriga muvofiq «zinama-zina» sxemasi bo'yicha joylashgan bo'ladi. SADT funksiyaviy diagrammalarida bloklar o'zaro ta'sirining beshta turi farqlanadi. Ular:




**4.6-rasm.** Funksional blok va interfeys yoylari.

- blokning kirish-chiqishi nisbatan kam ustunlik bilan blok kirishiga beriladi (4.7-rasm a);
- boshqaruv-blok chiqishi kam ustunlik bilan keyingi blok uchun boshqaruv sifatida foydalaniladi (4.7-rasm b);
- blokning kirish-chiqish bo'yicha qayta aloqasi blok kirishiga katta ustunlik bilan uzatiladi (4.7-rasm d);
- boshqaruv bo'yicha qaytuvchi aloqa-blok chiqishi blok uchun katta ustunlik bilan boshqaruvchi aloqa sifatida foydalaniladi (4.7-rasm);

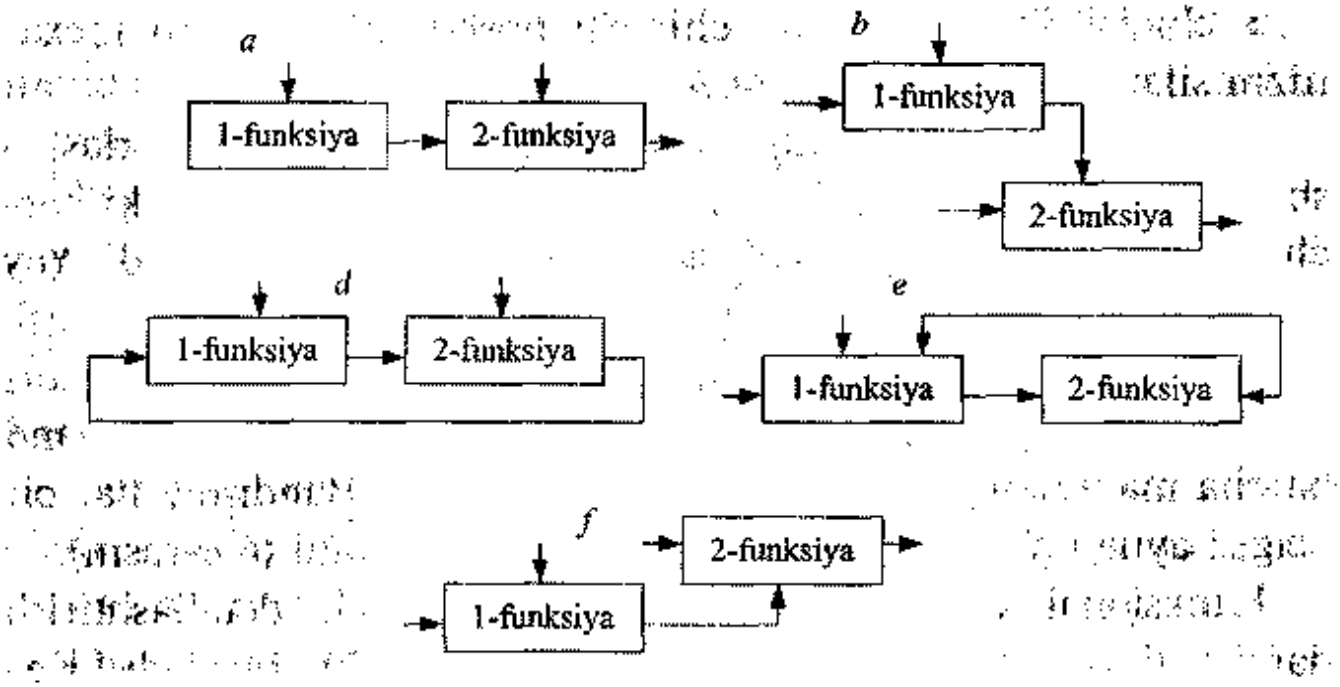
• chiqish-ijrochi – blok chiqishi boshqa blok uchun mexanizm sifatida foydalaniladi (4.7-rasm f).

Chiziqlar turlicha tarmoqlanishi va turli usullar bilan birlashishi mumkin. Tarmoqlanish ma'lumotlarning bir qismi yoki barchasi har ajralgan yoyda foydalanish mumkinligini anglatadi. Yoy har doim uzatiladigan ma'lumotlar to'plamini identifikatsiyalash uchun tarmoqlanguncha belgilanadi. Agar yoy bandi tarmoqlangandan so'ng belgilanmagan bo'lsa, u holda belgilanmagan band barcha ma'lumotlar to'plamini o'zida saqlaydi. Bandning har bir belgisi aynan shu band mavjudligini aniqlab beradi (4.8-rasm).

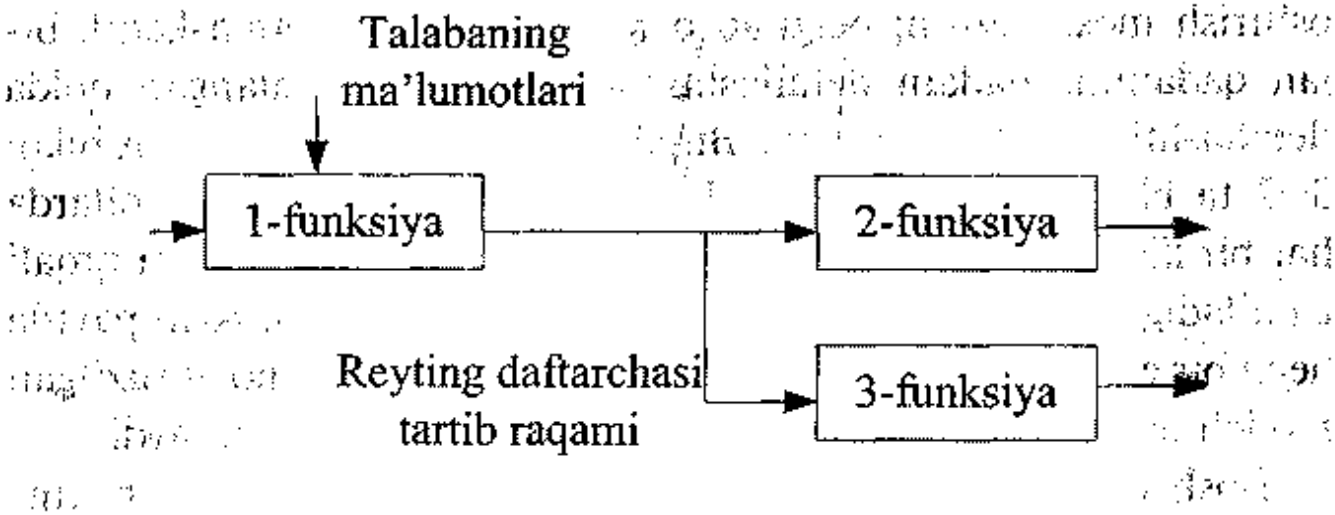
Funksional diagramma iyerarxiyasini tuzish detallashtirish darajasini oshirish orqali boqichma-bosqich olib boriladi. Keyingi har bir daraja diagrammasi asl blok strukturasini aniqlab beradi. Modelni tuzish yagona blokdan boshlanadi va uning uchun boshlang'ich ma'lumotlarni, natijalar, boshqaruv va amalga oshirish mexanizmini belgilab beradi. So'ngra u ketma-ketlik bilan qadamma-qadam detallashtirish usulidan foydalangan holda detallashtiriladi (1.4-mavzu). Bunda har bir funksiyani ko'pi bilan 3–7 ta blok bilan taqdim etish tavsiya qilinadi. Barcha hollarda har bir kichik funksiya faqat foydalanilgan yoki asl funksiya orqali uzatiladigan ma'lumotlar elementlardan foydalaniladi. Ayni paytda hech bir element tushirib qoldirilmasligi lozim va bu holat tuzilgan modelning to'g'riligini, ya'ni uni zid kelmasligini ta'minlaydi.

Bosh diagrammadan kiruvchi yoki chiquvchi strelkalar simvollar (belgilar) va sonlardan foydalangan holda raqamlanadi. Ramz aloqa turini bildiradi: i – kiruvchi, S – boshqaruvchi, M – mexanizm, R – natija, Son – asosiy blokning tegishli tomoni bo'yicha aloqa nomeri yuqoridan pastga va chapdan o'ngga qarab hisoblanadi.

Barcha diagrammalar bir-biri bilan bloklarning iyerarxik nomerlanishi orqali bog'lanadi: birinchi daraja – A0, ikkinchisi – A1, A2, uchinchi daraja A11, A12, A13 va h.k. Bunda birinchi raqam – bosh blok nomeri, oxirgisi esa – bosh blokka tegishli ma'lum bir kichik blok nomeri sanaladi.




**4.7-rasm.** Bloklarning ta'sir turlari: a – kirish; b – boshqaruv; d – kirish bo'yicha qayta aloqa; e – boshqaruv bo'yicha qayta aloqa; f – chiqish ijrochi.




**4.8-rasm.** Tarmoqlanganda yoylarni belgilash namunasi.

## 4.4. Ma'lumotlar oqimlari diagrammasi

*Ma'lumotlar oqimlari diagrammasi* ishlab chiqilayotgan dasturiy ta'minot funksiyasini ham, ular orqali qayta ishlanadigan ma'lumotlarni spetsifikatsiyalashga imkon beradi. Ushbu modeldan foydalanishda tizim ma'lumotlar oqimi diagrammasining iyerarxiyasi ko'rinishida taqdim etiladi. Iyerarxiyaning keyingi har bir darajasida navbatdagi jarayon elementar deb qayd etilmaguncha jarayonlar aniqlanib boriladi.

**Eslatma.** Ma'lumotlar oqimi modeli bir-biridan mustaqil ravishda avvaliga Y. Iordan (1975), so'ngra Ch. Geyn va T. Sarsonlar tomonidan (1979) taklif etilgan, Iordan–De Mark va Geyn Sarson fikricha muvofiq ushbu modellarda dasturiy ta'minotni strukturaviy tahlil etish va loyihalashtirish klassik metodologiyasiga asoslangan. Shuningdek, model Buyuk Britaniyada axborot tizimini ishlab chiqishning milliy standarti sifatida qabul qilingan SSADM (Structural Systems Analysis and Design Method)ni strukturaviy tahlil etish va loyihalashtirish metodologiyasida foydalaniladi.

Model asosida tashqi mohiyat, jarayon, ma'lumotlar va ma'lumotlar bazalari ombori (saqlovchisi) tushunchalari yotadi.

*Tashqi mohiyat* axborot manbai yoki qabul qiluvchi, masalan, buyurtmachi, personal, yetkazib beruvchi, mijozlar, bank va shu kabi moddiy obyekt yoki jismoniy shaxsdir.

*Jarayon* – ma'lum bir algoritmlarga muvofiq kiruvchi ma'lumotlar oqimining chiquvchi oqimga aylanishi. Tizimdagi har bir jarayon o'z nomeriga ega va mazkur qayta hosil bo'lishni amalga oshiruvchi ijrochi bilan bog'liq. Funksional diagramma holidagidek bu yerda almashtirishlar kompyuterda, qo'lda yoki maxsus qurilmalar bilan amalga oshirilishi mumkin. Iyerarxiyaning yuqori darajasida hali jarayonlar aniqlanmaganda «jarayon» tushunchasi o'rniga «tizim» va «tizim osti» tushunchasidan foydalaniladi. Ular tizimga muvofiq to'liq yoki funksiyaviy jihatdan tugallangan qismni bildiradi.

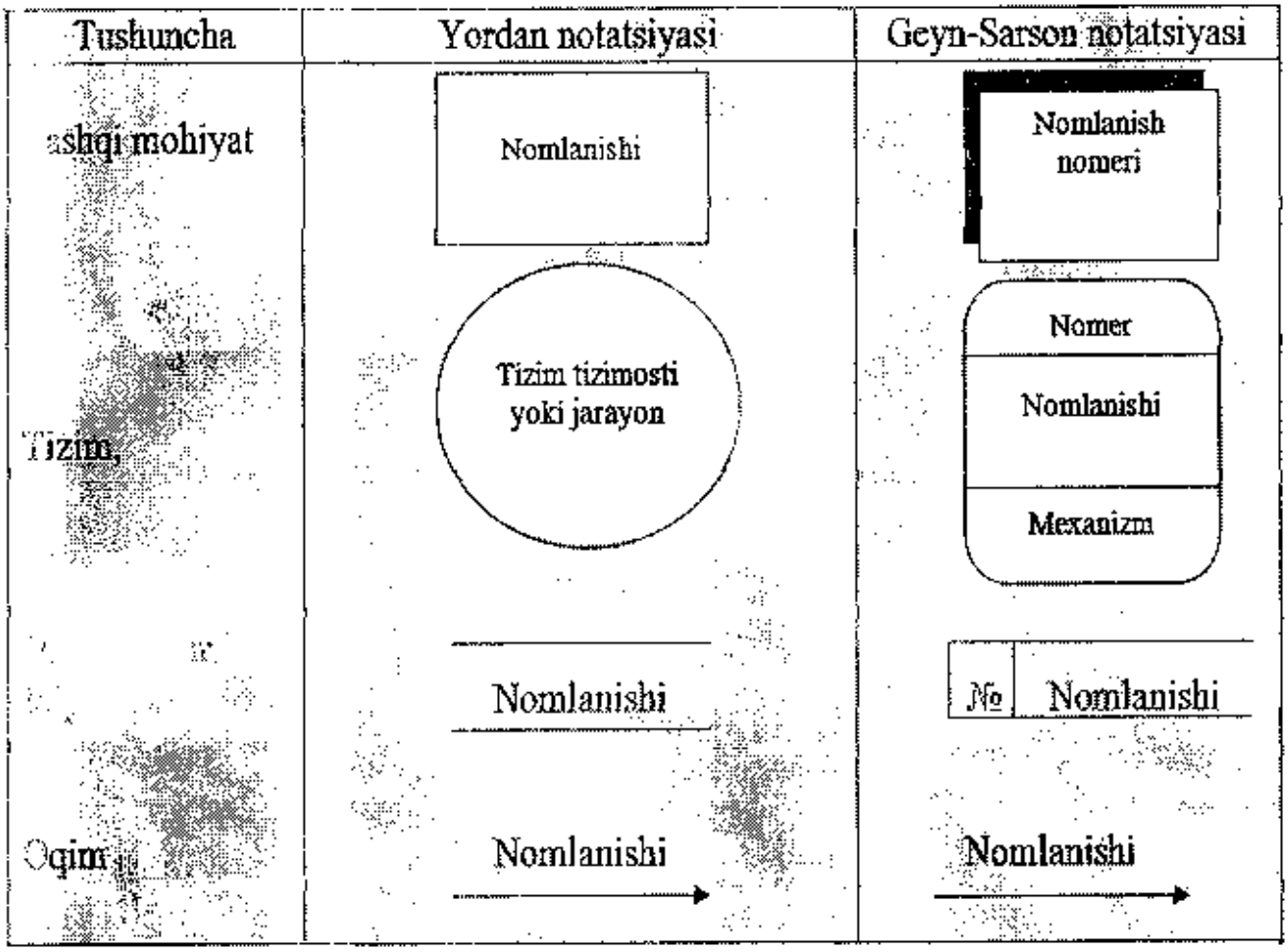
*Ma'lumotlar ombori* – axborotlarni saqlash uchun abstrakt qurilma. Qurilma turi va axborotni joylashtirish, olish va saqlash usullari bunday qurilma uchun detallashtirilmaydi. Jismoniy jihatdan bu ma'lumotlar bazasi, fayl, tezkor xotiradagi jadval, qog'oz ko'rinishidagi kartoteka va hokazolar bo'lishi mumkin.

*Ma'lumotlar oqimlari* – ayrim ma'lumotlarning manbadan uni qabul qiluvchiga uzatish jarayoni. Axborotni uzatish jarayoni kabel orqali dastur yoki tizim dasturi boshqaruvida yoki qurilma orqali qo'lda yoki loyihalashtirilayotgan tizimdan tashqari insonlar ishtirokida amalga oshirilishi mumkin.

Shunday qilib, diagramma ayrim tashqi mohiyatlardan yuzaga kelgan ma'lumotlar oqimi qanday qilib tegishli jarayonlar (yoki kichik tizim) bilan transformatsiyalashuvini, ma'lumotlar yig'uvchilari bilan qanday saqlanishi va boshqa tashqi mohiyatlarga – axborot tashuvchilariga qanday uzatilishini namoyon etadi. Natijada biz axborotni saqlash qayta ishlashning tarmoq modeliga ega bo'lamiz.

Ma'lumotlar oqimi diagrammasini tasvirlash uchun odatda notatsiyaning ikki xil turidan foydalaniladi. Bular: Iordan va Geyn-Sarson notatsiyasi (4.1-jadval).

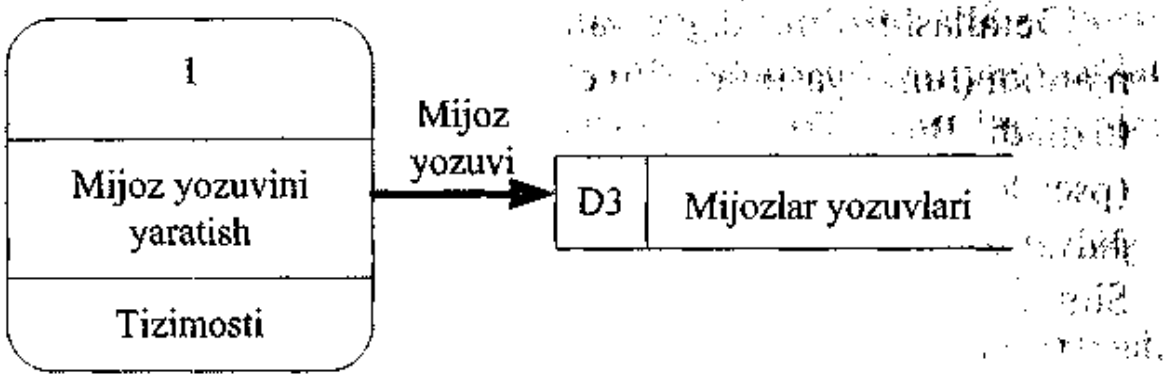
*4.1-jadval*




*Strelka* bilan belgilanadigan oqim chizig'i ustida ushbu holatda aynan qanday axborot uzatilishi ko'rsatiladi (4.9-rasm).

Ma'lumotlar oqimi diagrammasining iyerarxiya tuzilishi alohida turdagi diagramma – kontekstli diagramma bilan boshlanib, u tizimning nisbatan umumiy ko'rinishini belgilab beradi. Bunday

diagrammada ishlab chiqilayotgan tizim ijrochining ko'rsatmasisiz axborot qabul qiluvchi va uning manbai o'rtasida o'zaro qanday harakat qilishi ko'rsatiladi. Aniqrog'i, tizim va tashqi olam o'rtasidagi interfeysni yozib oladi. Odatda boshlang'ich kontekstli diagramma yulduzlar shaklida bo'ladi.




**4.9-rasm.** Ma'lumotlar oqimi namunasi (Geyn-Sarson notatsiyasi).

Agar loyihalashtirilayotgan tizim ko'p sonli tashqi muhitga (10 dan ortiq), alohida xususiyatga ega bo'lsa yoki mavjud tizimostini o'z ichiga olsa, u holda kontekstli diagramma iyerarxiyani tuzadi.

Kontekstli diagrammani ishlab chiqishda kelgusi tizimning funksiyaviy tuzilmasini detallashtirish amalga oshiriladi. Agar ishlanma bir necha jamoa tomonidan olib borilayotgan bo'lsa juda muhim.

Ana shunday tarzda olingan tizim modeli tizim obyektlari to'g'risidagi boshlang'ich ma'lumotlarning to'liqligi va ularning boshqa obyektlar bilan axborot aloqasi yo'qligini tekshiradi.

Keyingi bosqichda kontekstli diagrammaning har bir tizimosti ma'lumotlar oqimi diagrammasi yordamida detallashtiriladi. Detallashtirish jarayonida balanslash qoidasiga rioya etiladi. Tizimostini detallashtirishda faqat ishlab chiqilayotgan tizimosti bilan axborot aloqasi bo'lgan tizimosti komponentalaridangina foydalanish mumkin.

Jarayonni detallashtirishni yakunlash to'g'risidagi qaror quyidagi hollarda qabul qilinadi:

• jarayon 2 ta ma'lumotlar oqimi bilan o'zaro harakat qilganda;

• jarayonni ketma-ket (uzviy) algoritmlar bilan bayon etish imkoniyati bo'lganda;

• jarayon kirituvchi ma'lumotlarni chiquvchi ma'lumotga qayta aylantirishning yagona mantiqiy funksiyasini bajarganda.

Detallashtirilmaydigan jarayonlar uchun mazkur jarayonning mantiqi (funksiyasi) ta'rifini o'z ichiga olgan spetsifikatsiya (tasnif) tuziladi. Bunday ta'rif tabiiy tilda, tuzilmalashtirilgan tabiiy tilni (psevdokodlar) qo'llagan holda, algoritmlar sxemasi ko'rinishida jadval va masalalar «daraxti»ni, shuningdek flow-forma va Nassi-Shneyderman diagrammasidan foydalanib tuzilishi mumkin.

Detallashtirilayotgan tizimosti jarayonlarini qabul qilishni osonlashtirish uchun ular iyerarxiya nomerlariga rioya etilgan holda raqamlanadi. Bunda jarayonni yoki 1-tizimostini detallashtirishda olingan jarayonlar «1.1», «1.2» va hokazo shaklda nomerlanishi lozim. Bundan tashqari har bir diagrammada 3 tadan 6–7 tagacha jarayonlarni joylashtirish va ushbu darajada muhim bo'lmagan detallar bilan diagrammani tirband qilib qo'ymaslik tavsiya etiladi.

Ma'lumotlar oqimini dekompozitsiyalashni jarayonlarni dekompozitsiyalash bilan parallel holda amalga oshirish zarur.

Modellarni *yakuniy ishlab chiqish* ikki bosqichda amalga oshiriladi.

**1-bosqich** – kontekstli diagrammani tuzish quyidagi amallarni bajarishni o'z ichiga oladi:

• ko'plab talablarni tasniflash va ularni asosiy guruhlar, jarayonlarga tashkil etish;

• tizim bilan bog'langan tashqi obyektlar, tashqi mohiyatni identifikatsiyalash;

• asosiy axborot turlarini, tizim va tashqi obyektlar o'rtasida aylanuvchi malumotlar oqimini identifikatsiyalash;

• kontekstli diagrammani dastlabki qayta ishlash;

• dastlabki kontekstli diagrammani o'rganish va unga barcha bo'limlar bo'yicha masalalarni o'rganishda yuzaga keladigan javoblar natijalariga ko'ra o'zgartirishlar kiritish;

• dastlabki diagrammalar barcha jarayonlarini bitta jarayonga birlashtirish orqali kontekst diagrammani tuzish.

**2-bosqich** – ma'lumotlar oqimi diagrammasining iyerarxiyasini shakllantirish (har bir daraja uchun):

• detallashtiruvchi diagramma yordamida ma'lumotlar oqimining joriy diagrammasiga oid har bir jarayonni dekompozitsiyalash. Agar ayrim funksiyalar murakkab bo'lsa, jarayonlar kombinatsiyalari orqali ifodalash mumkin bo'lmasa, jarayonlar tasnifini tuzish;

• yangi oqimlar diagrammada paydo bo'lganda ularni ma'lumotlar lug'atiga qo'shib qo'yish;

• ikki-uchta daraja tuzilgandan so'ng ko'rinishni yaxshilash va to'g'riligini tekshirish maqsadida taftish o'tkazish.

Jarayonlarni to'liq tasniflash ham oqimlardagi axborotni uzatishda, ham ularni saqlashda foydalaniladigan ma'lumotlar tuzilmasini bayon etishni o'z ichiga oladi. Bayon etiladigan ma'lumotlar tuzilmasi muqobillarni, shartli kirish va iteratsiyani o'z ichiga olishi mumkin. Shartli ravishda kirish deganda ma'lumotlarning tegishli elementlari tuzilmada bo'lmasligi mumkinligini tushuniladi.

Muqobillik shuni anglatadiki, tuzilmaga qayd etilgan elementlardan biri kirishi mumkin. Iteratsiya esa elementning bir necha marta qaytarilishi mumkinligini anglatadi.

Bundan tashqari ma'lumotlar uchun uzluksiz yoki diskret ahamiyatga ega turlari ko'rsatilishi kerak. Uzluksiz ma'lumotlar uchun o'lchov birligi, ahamiyat ko'lami, tasavvur aniqligi va fizik kodlashtirish shakli belgilanishi mumkin. Diskretli tur uchun esa mumkin bo'lgan qiymatlar jadvali ko'rsatilishi mumkin.

Olingan tugallangan modelning to'liqligi va mosligini tekshirib ko'rish lozim. Model mosligi deganda mazkur holatda barcha ma'lumotlar oqimi uchun axborotni saqlash qoidasi bajarilganligi tushuniladi. Keluvchi barcha ma'lumotlar hisoblanishi va yozib olinishi lozim.

**Ma'lumotlar oqimi diagrammasi yordamida boshqaruvchi jarayonlarni modellashtirish.** Boshqaruvchi jarayonlarni taqdim etish

uchun loyihalashtirilayotgan tizimda 4.2-mavzuda ko'rib chiqilgan holatlar o'tishi diagrammasini yoki ma'lumotlar oqimini boshqarish diagrammasini qo'llash mumkin. Unda boshqaruvchi jarayon, ma'lumotlarning boshqaruvchi oqimi va ma'lumotlarni boshqarish ombori kabi tushunchalardan foydalaniladi.




**4.10-rasm.** Ikkinchi darajadagi ma'lumotlar oqimining detallashtiruvchi diagrammasi (Geyn-Sarson notatsiyasi).

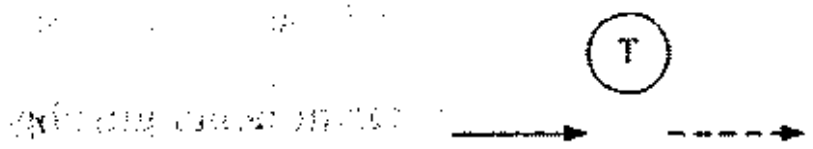
Boshqaruvchi jarayon boshqaruvchi oqim yordamida tizimdagi vaziyat to'g'risida ayrim axborotlarga ega bo'ladi va boshqaruvchi oqim vositasida tegishli jarayonlarni initsiallaydi.

Ma'lumotlar oqimini boshqaruvchi diagrammada ham, odatdagi oqimlar uchun belgilardan foydalaniladi. Lekin ularni punktli chiziq bilan tasvirlaydi. Qo'shimcha ravishda boshqaruvchi oqimning quyidagi turi ko'rsatilishi mumkin:

- T-oqim (Trigger-Flow – tirger oqim) – jarayonni faqat «yoqish» imkoniga ega boshqaruv oqimi, navbatdagi boshqaruvchi signal yana jarayonni «yoqish» bo'ladi (garchi jarayon faol bo'lsa ham).
- A-oqim (Activator Flow – detallashtiruvchi oqim) – boshqaruvchi jarayonni ham «yoqish», ham «o'chirish» imkoniga ega boshqaruv oqimi.
- Ye/D-oqim (Enable Flow – o'zgaruvchan, ko'chuvchi oqim) – jarayonni bitta (Ye) liniya bo'yicha yoqish va boshqa liniya (D) signali bo'yicha o'chirishi mumkin bo'lgan boshqaruv oqimi.

Zarur holatda ma'lumotlar oqimi turini (boshqaruvchi yoki odatdagi) o'zgartirish mumkin. Buning uchun maxsus belgi – ma'lumotlar oqimi turlarini o'zgartirish tugunidan foydalaniladi

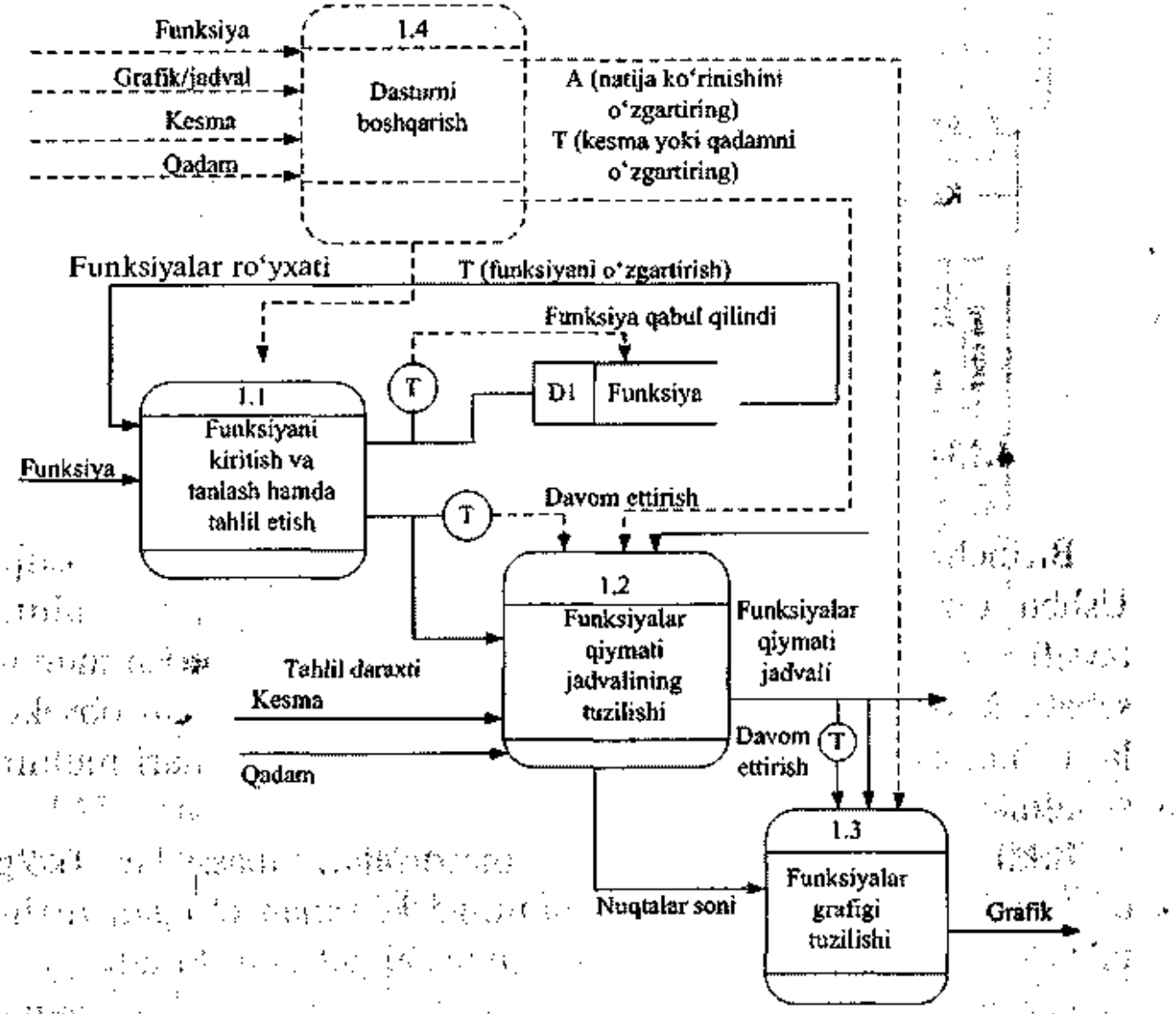
(4.11-rasm). Bu tugunga oqim ma'lumotlar oqimi sifatida keladi, undan esa boshqaruvchi oqim sifatida chiqadi.




**4.11-rasm.** Ma'lumotlar oqimiga o'xshash o'zlashtirish tuguni.

## 4.5. Ma'lumotlar tuzilmasi va ma'lumotlar komponentlari munosabatining diagrammasi

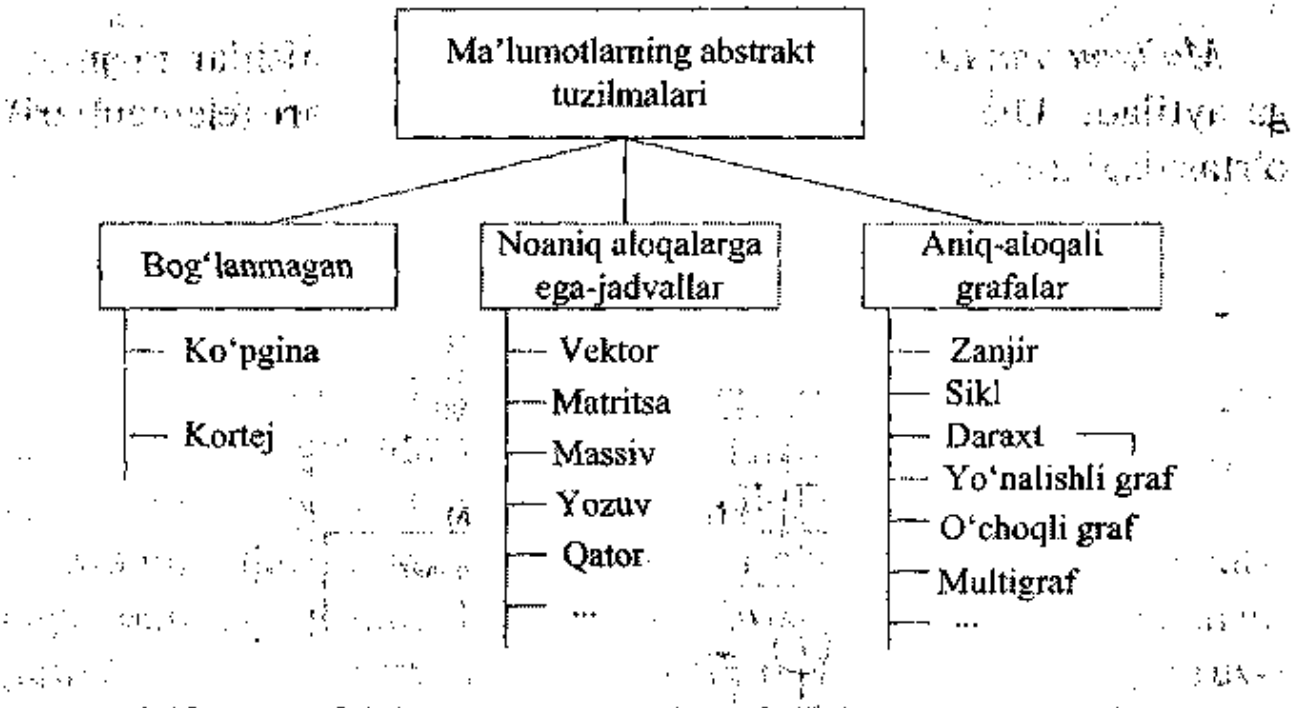
*Ma'lumotlar tuzilmasi* deb, qoidalar va cheklanishlar majmuiga aytiladi. Ular ma'lumotlarning alohida qismlari (elementlari) o'rtasidagi aloqalarni aks ettiradi.




**4.12-rasm.** Boshqaruvchi ma'lumotlar oqimi diagrammasi bilan to'ldirilgan ma'lumotlar oqimi diagrammasi.

Elementlar o‘rtasidagi aloqani aniqlash uchun foydalanadigan ma’lumotlarning abstrakt tuzilmasi va dasturdagi ma’lumotlarni taqdim etish uchun foydalanadigan konkret tuzilmalar farqlanadi.

Ma’lumotlarning barcha abstrakt tuzilmalarini uchta guruhga ajratish mumkin: bir-biri bilan o‘zaro bog‘lanmagan tuzilmalar, elementlar. Elementlar noaniq bog‘langan strukturalar. Elementlar grafalar aloqasi aniq ko‘rsatilgan tuzilmalar (4.13-rasm).



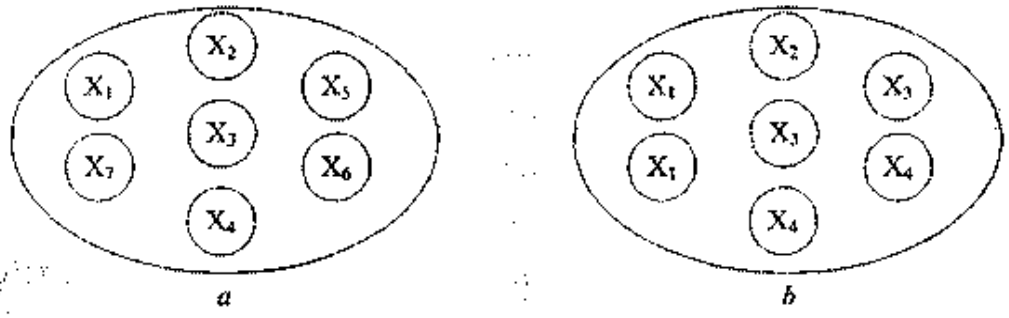

**4.13-rasm. Ma’lumotlarning abstrakt tuzilmalari tasnifi.**

Birinchi guruhga ko‘plik va kortej aloqalar kiradi (4.14-rasm). Ushbu tuzilmada ma’lumotlar elementining nisbatan muhim tavsifi – uning qaysidir to‘plamga tegishliligi, ya’ni kirish munosabati. Mazkur abstrakt tuzilmalar agar, ta’riflanadigan obyektlar uchun elementlarning hech bir boshqa munosabatlari muhim sanalmasa foydalaniladi.

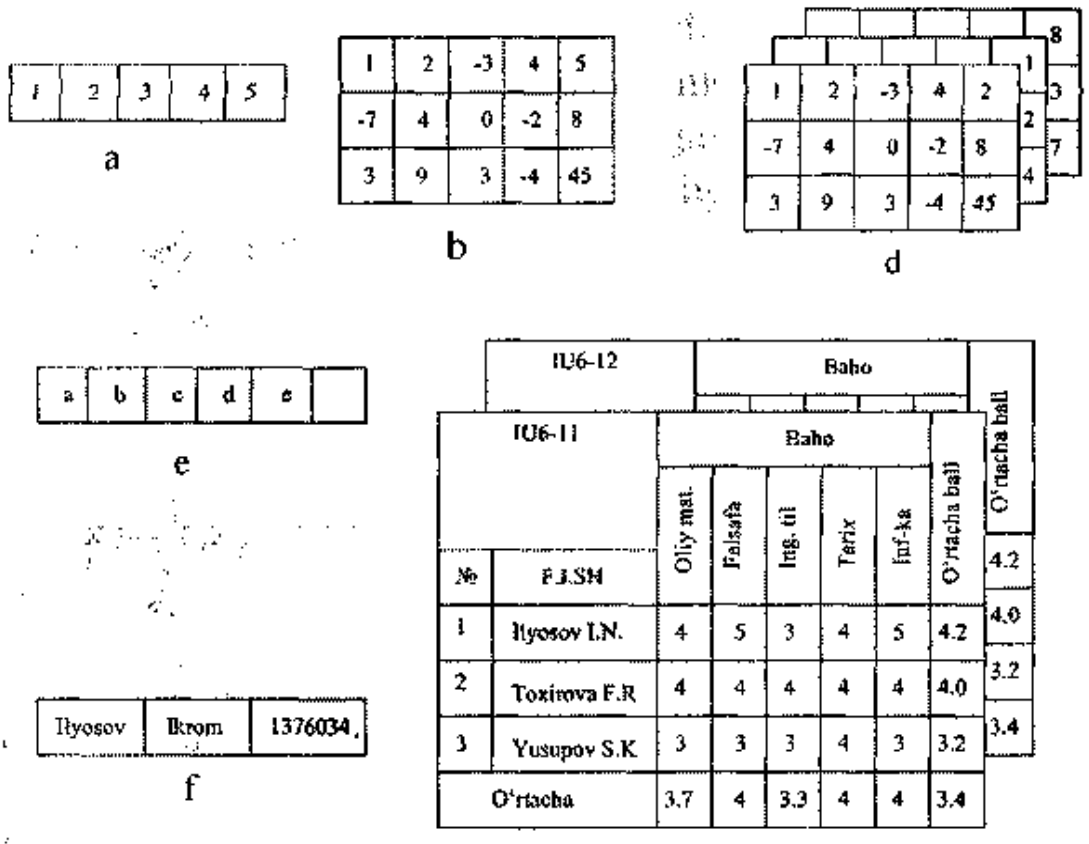
Ikkinchi guruhga vektorlar, matritsalar, massivlar (ko‘p o‘lchamli), yozuvlar, qatorlar, shuningdek, sanab o‘tilgan tuzilmalarni bir qism sifatida o‘z ichiga oluvchi jadvallar kiradi.

Ushbu abstrakt tiplardan foydalanish shuni anglatishi mumkinki, bunda ma’lumotlar elementining qaysidir tuzilmaga kirishi-

gina emas, ularning tartibi, shuningdek tuzilmalar iyerarxiyasining munosabatlari, ya'ni tuzilmaning yanada yuqori darajadagi tuzilmaga kirishi ham muhim sanaladi (4.15-rasm).




**4.14-rasm.** Ko'plik (a) va kortej (b).




**4.15-rasm.** Jadvallar: a – raqamli vektor; b – matritsa; d – uch o'lchovli massiv; e – qator; f – yozuv; g – tuzilmalar kiritilgan bir tipdagi jadvallar massivi.

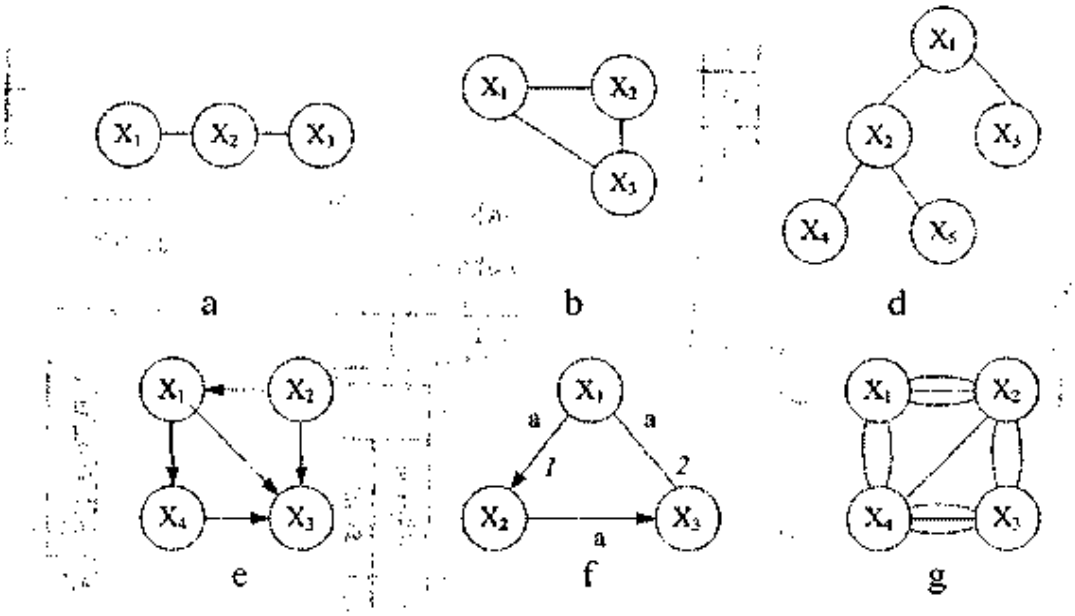
Ma'lumotlar elementlari o'rtasidagi aloqalar muhim bo'lganda ma'lumotlar tuzilmasi modeli sifatida grafalardan foydalaniladi. 4.16-rasmda grafali modellarning turli variantlari ko'rsatilgan.

Ta'rif turlariga ko'ra, ma'lumotlar tuzilmasi modeli munosabatlari iyerarxik va tarmoqli modellarga ajratiladi.

*Iyerarxik modellar* ma'lumotlar elementlarining yuqori darajadagi komponentalarga kirishga oid tartibga solingan va tartibsiz munosabatlarini ta'riflash imkonini beradi.

Iyerarxik modelga Jekson-Orr modeli mansub bo'lib, grafik tasvir uchun quyidagilardan foydalanish mumkin:

- *Jekson diagrammasi.* U 1975-yil mazkur muallifning dasturiy ta'minotni loyihalash metodikasi tarkibida taklif etilgan;
- *Orrning qavsli diagrammasi.* U Varner Orrning dasturiy ta'minotni loyihalashtirish metodikasi tarkibida taklif etilgan.




**4.16-rasm.** Grafalar. a – zanjir; b – sikl; d – daraxt; e – mo'ljallangan grafa; f – muallaq aralash grafa; g – multigrafa.

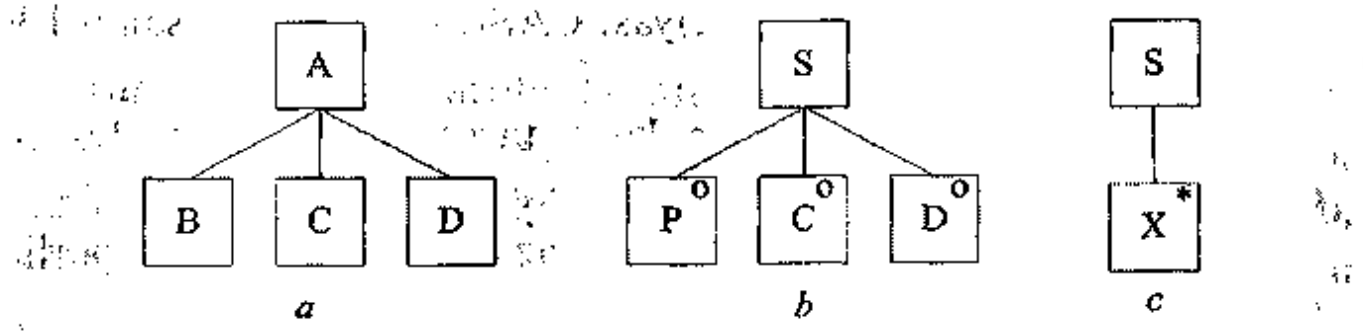
Tarmoq modellar grafalarga asoslangan, shu bois munosabatlar turidan qat'i nazar ma'lumotlar elementlari, shuningdek, ko'pgina kombinatsiyalar, jadvallar va grafalarning aloqalarini bayon etishga imkon beradi. Tarmoq modellarga masalan, odatda ma'lumotlar bazalarini ishlab chiqishda foydalaniladigan «mohiyat-aloqa» (ER-Entity – Relationship) modeli kiradi.

**Jekson diagrammasi.** Jekson diagrammasi asosida shunday faraz yotadiki, unga ko'ra dasturlar kabi ma'lumotlar tuzilmasini

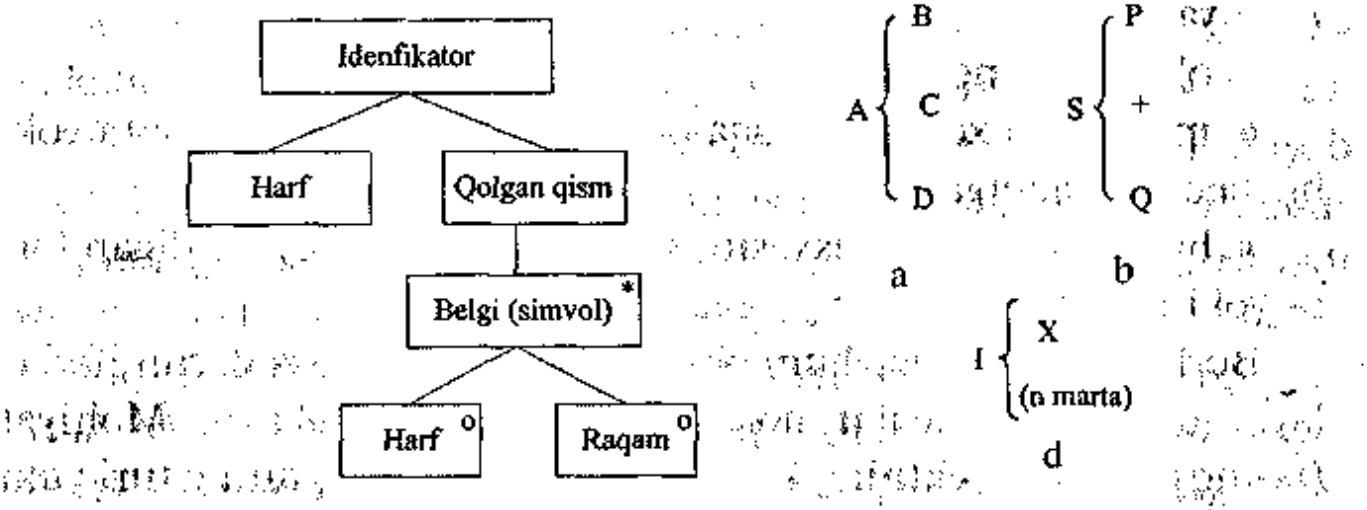
ham uchta asosiy konstruksiya – uzviylik, tanlash va takrorlashdan foydalangan holda elementlardan tuzish mumkin.

Har bir konstruksiya ikki darajali iyerarxiya ko'rinishida ifodalanadi. Yuqori darajada konstruksiya bloki, quyi darajada esa elementlar bloki joylashgan. Uzviylik tasvirida qo'shimcha ramz mavjud emas. Tanlov tasviriga «O» (lotincha) ramzi – inglizcha «yoki» so'zining qisqartmasi (or). Uzviylik va tanlov konstruksiyasi ikkinchi darajadagi ikki va undan ortiq elementga ega bo'lishi lozim. Qayta tasvirda yagona (takrorlanuvchi) elementga «*» ramzi qo'yiladi.

4.17, a-rasmda ko'rsatilgan sxema shuni anglatadiki, A konstruksiya B, C va D elementlardan iborat. 4.17, b-rasmdagi sxema shuni bildiradiki, S konstruksiya Q elementidan yoki R elementidan iborat. 4.17, d-rasmda tasvirlangan sxema ko'rsatishicha, konstruksiya elementlarga ega bo'lmasligi, yoki bitta va undan ko'p X elementiga ega bo'lishi mumkin.




**4.17-rasm.** Konstruksiyalarni tasvirlash uchun Djekson notatsiyasi:
a – ketma-ketlik; b – tanlash; d – takrorlanish.




**4.18-rasm.**

Agar takrorlash konstruksiyasi bitta yoki undan ko'p elementga ega ekanligini ko'rsatish kerak bo'lsa, takrorlash va uzviylikning ikki tuzilmasidan foydalaniladi.

**Orrning qavsli diagrammasi.** Orr diagrammasi ham Jekson diagrammasi kabi ma'lumotlar va dasturlar tuzilmasi mosligi to'g'risidagi taxminga asoslanadi. Farq faqat shartli yozma belgilarda. Muallif ma'lumotlar konstruksiyasini tasavvur etish uchun figurali qavslardan foydalanadi.

**Ma'lumotlarning tarmoq modeli.** Agar ma'lumotlar komponentalari orasidagi munosabatlar qo'shilishlarda tugamasa, u holda ma'lumotlarning tarmoq modelidan foydalaniladi. Ushbu modellar turini grafik taqdim etish uchun bir necha notatsiyalardan foydalaniladi. Ulardan quyidagilar eng mashhur sanaladi:

- P. Chen notatsiyasi;
- R. Barker notatsiyasi;
- IDEF1 notatsiyasi (ushbu notatsiyaning nisbatan zamonaviy varianti – IDEF1 X metodologiyasi CASE tizimida, masalan ER Win tizimida foydalaniladi).

Barner notatsiyasi nisbatan keng tarqalgan. Bundan buyon ushbu bo'limda aynan shu notatsiyaga asoslanamiz.

Ma'lumotlar tarmoq modelining bazaviy tushunchasi – borliq (mohiyat), abstrakt va aloqa sanaladi.

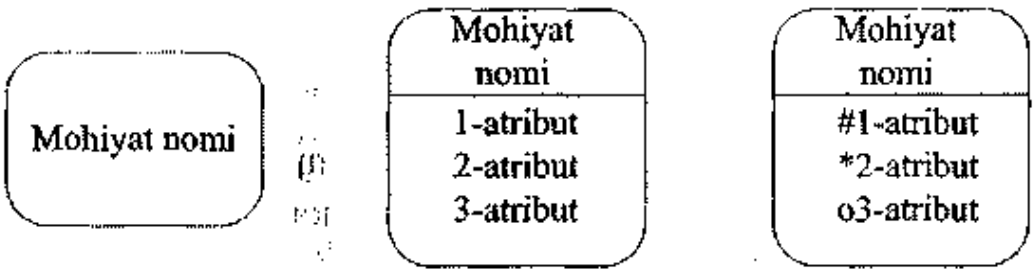
Borliq (mohiyat) – ko'rib chiqilayotgan predmet sohasi uchun muhim ahamiyatga ega aniq yoki tasavvurdagi obyekt. Har bir mohiyat:

- o'ziga xos nomga;
- mohiyatga xos yoki aloqa orqali meros qilinadigan bitta yoki bir necha atributga;
- har bir mohiyat nusxasini so'zsiz identifikatsiyalaydigan bitta yoki bir nechta atributga ega bo'lishi lozim.

Borliq real yoki mavhum obyektlarning ko'p nusxalarini ifodalaydi (odamlar, voqealar, holatlar, buyumlar va hokazo). Mohiyat (borliq) nomi obyektning konkret nusxasini emas, uning turi yoki sinfini aks ettirishi lozim (masalan, Vnukovo emas, Aeroport).

Barker notatsiyasi diagrammasida mohiyat to'g'ri burchak bilan ba'zan aylana bilan tasvirlanadi (4.19, a-rasm).

Har bir mohiyat bir yoki bir necha atributga ega. Atribut ko'rib chiqilayotgan predmet sohasi uchun ahamiyatli va mohiyat holatini tasniflash, identifikatsiyalash, kvalifikatsiyalash, miqdoriy tavsiflashga mo'ljallangan har qanday ta'rifdir (4.19, b-rasm).




**4.19-rasm.** Barner notatsiyasida mohiyat mazmuni: a – atributlarsiz; b – atributlar ko'rsatilgan holda; d – atributlar va ularning turlari aniqlangan holda (# – asosiy, * – majburiy, o – majburiy bo'lmagan).

Tarmoq modelda atributlar aniq mohiyat bilan assotsiatsiyalanadi. Mohiyat nusxasi assotsiatsiyalangan atribut uchun yagona, ma'lum bir ahamiyatga ega bo'lishi lozim. Atribut, shunday qilib, ko'plab rol yoki mavhum obyektlar bilan assotsiyalashgan ayrim tavsiflar yoki xususiyatlar turidan iborat. Atribut nusxasi – mohiyatning aniq nusxasiga xos ma'lum bir tavsif. U tavsif turi va atribut ahamiyati bilan belgilanadi.

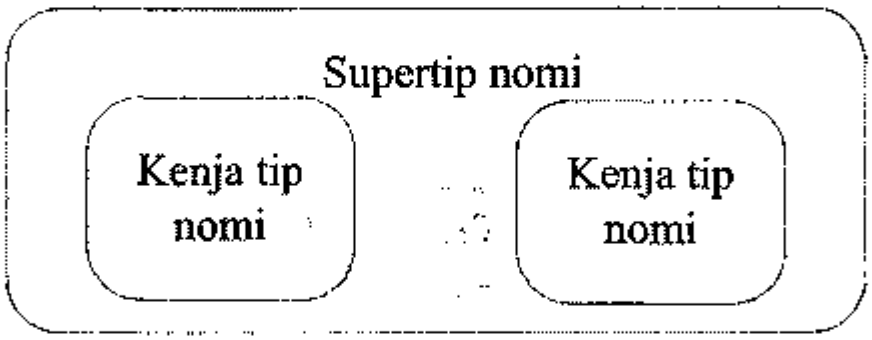
Atributlar asosiy (noyob identifikator tarkibiga kiruvchi dastlabki kalit atributlar) va bayon etiladigan boshqa atributlarga bo'linadi.

*Boshlang'ich kalit* – mohiyatning har bir nusxasini noyob identifikatsiyalash uchun mo'ljallangan aloqalar yoki atributlar majmui (yoki atribut). Kalitli atributlar ro'yxat boshiga kiritiladi va «#» ramzi bilan belgilanadi (4.19, b-rasm).

Bayon etiladigan atributlar majburiy va nomajburiy bo'ladi. Majburiy atributlar mo'ljallangan ahamiyat belgilanmagan bo'lishi mumkin. Majburiy atributlar «*», majburiy bo'lmaganlari «0» ramzlari bilan belgilanadi.

Mohiyat uchun supertip va tip ostisi tushunchalari belgilangan.

*Supertip* – mohiyatning ayrim guruhlarini (tip ostidagilarni) boyituvchi tur. Supertip atributlar va munosabatlarga ega tip ostidagilar uchun umumiyligi bilan izohlanadi. Masalan, ayrim masalalarni yechishda «o'quvchi» supertipi «maktab o'quvchisi» va «talaba» tip ostidagi (pod tip)larini boyitadi (4.20-rasm).



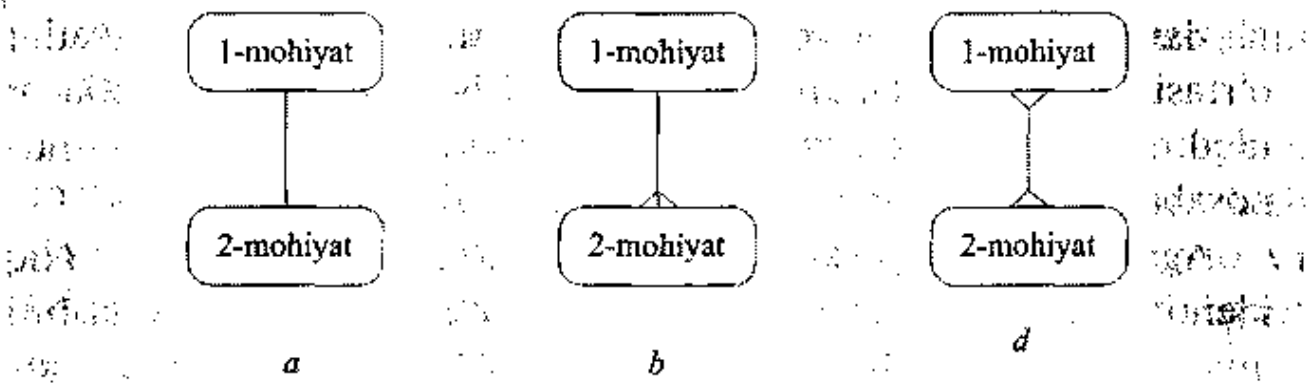

**4.20-rasm.** Barker notatsiyasida supertiplar va tip ostidagilar ahamiyati.

Aloqa qaralayotgan predmet coha uchun ahamiyatli bo'lgan ikki yoki undan ortiq mohiyatlar o'rtasidagi nomlangan assotsiatsiya deyiladi.




**4.21-rasm.** Barker notatsiyasidagi aloqalarning belgilanishi: a – majburiy; b – shart bo'lmagan (punktir aloqa chizig'ining yarmigacha ko'rsatiladi).

*Aloqa* – ko'rib chiqilayotgan predmet sohasi uchun mo'ljallangan ikki yoki undan ortiq mohiyat o'rtasidagi bitta mohiyatning har bir nusxasi ikkinchi mohiyat nusxalarining ixtiyoriy miqdori bilan assotsiatsiyalangan. Agar bitta mohiyatning istalgan nusxasi boshqa mohiyatning hech bo'lmasa bitta nusxasi bilan bog'langan bo'lsa, unda aloqa majburiy bo'ladi (4.21, a-rasm). Shart bo'lmagan aloqa mohiyatlar o'rtasidagi shartli munosabatlarni ifodalaydi (4.21, b-rasm).




**4.22-rasm.** Barker notatsiyasida munosabatlarning belgilanishi: a – «bittaga-bitta»; b – «bitta-ko'pga»; d – «ko'p-ko'pga».

Munosabatlarning uchta tipi farqlanadi (4.22-rasm).

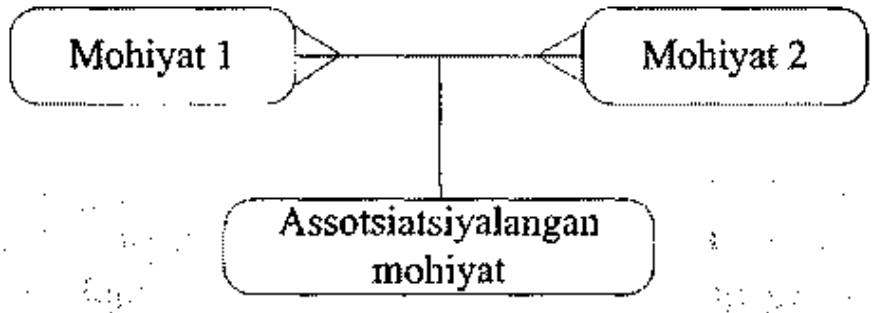
| * | – «bittaga-bitta», birinchi mohiyatning bitta nusxasi ikkinchi mohiyatning bitta nusxasiga mos keladi.

1* n – «bitta-ko'pga», birinchi mohiyatning bitta nusxasi ikkinchi mohiyatning ko'pgina nusxasiga mos keladi.

n * m – «ko'p-ko'pga», birinchi mohiyatning har bir nusxasi ikkinchisining bir necha nusxasiga mos kelishi mumkin va aksincha, ikkinchi mohiyatning har bir nusxasiga birinchi mohiyatning bir necha nusxalari to'g'ri keladi.

Bundan tashqari, mohiyat mustaqil, tobe va assotsiatsiyalangan bo'ladi. Mustaqil mohiyat doimiy ravishda tizimda bo'ladigan mustaqil ma'lumotlarni taqdim etadi. Ular o'sha tizimdagi boshqa mohiyatlar bilan bog'liq yoki bog'lanmagan bo'lishi mumkin.

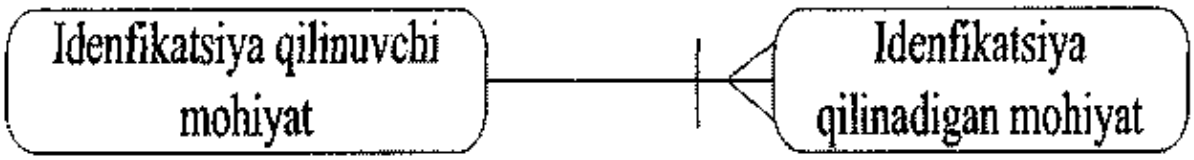
*Tobe (bog'liq) mohiyat* tizimdagi boshqa mohiyatlarga bog'liq ma'lumotlarni taqdim etadi. Shu bois u doim boshqa mohiyatlar bilan bog'liq bo'lishi lozim.




**4.23-rasm.** Barker notatsiyasida assotsiatsiyalangan mohiyatni belgilash.

*Assotsiatsiyalangan mohiyat* ikki va undan ortiq mohiyatlar o'rtasidagi munosabatlar bilan assotsiatsiyalangan ma'lumotlarni taqdim etadi. Odatda mohiyatning ushbu turi «ko'p-ko'pga» munosabatlarini hal etish modelida foydalaniladi (4.23-rasm).

Agar mohiyat nusxasi o'zining kalit atributlari bilan to'liq identifikatsiyalansa, bu mohiyatning to'liq identifikatsiyalashni anglatadi. Aks holda, mohiyatni identifikatsiyalash bog'liq mohiyatlar atributlaridan foydalangan holda amalga oshiriladi. Bu aloqa chizig'ida defis bilan ko'rsatiladi (4.24-rasm).




**4.32-rasm.** Barker notatsiyasida boshqa mohiyat vositasidagi identifikatsiyani belgilash.

Bundan tashqari, model o'zaro inkor etuvchi, rekursiv va aralashmaydigan aloqalarni o'z ichiga oladi. Bir-birini rad etish holatida mohiyat nusxasi bir necha aloqa guruhidan faqat bittasida ishtirok etadi. Rekursiv aloqa shuni nazarda tutadiki, mohiyat o'z-o'zi bilan aloqada bo'lishi mumkin. Aralashmaydigan aloqa deganda mohiyat nusxasi bitta aloqa nusxasini boshqasiga o'tkazib bo'lmasligini bildiradi.

## 4.6. Masalalar ishlanmalarining matematik modeli yoki hal etish usullarini tanlash

Topshiriqlar, algoritimlar uchun turli xil matematik modellardan foydalaniladi. Bunday modellarni tuzish quyidagilarni o'z ichiga oladi:

- masalalar shartlari tahlili;
- matematik abstraksiyalar tanlovi, adekvat, ya'ni dastlabki ma'lumotlar va natijalarni aniq va to'liqligini talab etuvchi;
- masalani formal qo'yilishi;

• dastlabki ma'lumotlarni natijaga aylantirish, ya'ni masalani hal etish usulini aniqlash.

Amaliyotda tez-tez uchraydigan ko'plab masalalar uchun matematikada jarayonni (muammoni) modeli va uni yechish algoritmlari aniqlangan. Bunday masalalarga, masalan fazo va tekislikdagi analitik geometriyaning ko'plab masalalari, diskret tizimlarini modellashtirish masalalari kiradi.

Bunday hollarda asosiy muammo konkret masalani yechish uchun u yoki bu matematik model qo'llanilishini asoslashdan iborat.

Qator hollarda masalani formal qo'yilishi uni yechishni bir qiymatini aniqlaydi. Lekin qoidaga ko'ra yechish usullari mavjud bo'lsa, u holda usulni tanlash uchun maxsus tadqiqot o'tkazilishi talab etilishi mumkin:

• Predmet sohasi bilan bog'liq konkret masalalar ma'lumotlarning o'ziga xos xususiyatlari (xatoliklar mumkin bo'lgan alohida holatlar va h.k.).

• Natijalarga talablar (yo'l qo'yish mumkin bo'lgan xatolar).

• Usul tavsifi (aniq yoki shunga yaqin ta'rif, natijalar xatolari, hisoblash va hajm jihatidan murakkablik, amalga oshirish murakkabligi va h.k.).

## 4.7. Minimal yo'l siklini qidirish masalani formal qo'yilishini bajarish (kommivoyajer masalasi)

Ma'lumki, kommivoyajer masalasi yoki eng qisqa uzunlikdagi siklni qidirish oddiy variantda quyidagicha ta'riflanadi. Deylik, shaharlar va ularni birlashtirib turuvchi yo'llar ro'yxati berilgan. Shaharlar oralig'idagi masofa ham ma'lum. Bunda barcha shaharlardan shunday aylanib chiqish kerakki, unda bir shaharga ikki marta kirish mumkin emas. Biror shahardan boshlab so'ng, dastlab yo'lga chiqilgan shaharga qaytib kelish kerak. Bosib o'tilgan yo'lning umumiy uzunligi esa imkon qadar eng qisqa masopredmetini tashkil etishi lozim.

Masala shartlari imkoni shuni ko'rsatadiki, tizim obyektlarining matematik modeli va mavjud yoki ular o'rtasidagi mumkin bo'lgan

aloqalar o'lchangan, mo'ljalli yoki yo'nalishi yo'naltirilmagan graf – G (X, < U, L >) bo'lishi mumkin. Bunda, X, U, L – uchlar, qirralar va qirralar og'irliklar to'plami hisoblanadi.

Masalalar obyektidan ularning matematik modeliga o'tish uchun quyidagilar talab etiladi:

- obyektlar komponentlarining modellar komponentiga muvofiqligi qoidasini ta'riflash;
- ushbu muvofiqlik turlarini aniqlash (o'zaro aynan, ko'p jihatli);
- tanlangan matematik abstraksiya tavsifida obyekt komponentlari xususyatlari va tavsifini aks ettirish usulini aniqlash.

Bular barchasi obyekt komponentlari o'rtasida mavjud munosabatlardan, shuningdek, obyekt xususiyatlari va uning komponentlari tavsifidan kelib chiqib aniqlanadi.

G grafda tizimning E obyektlari (shaharlar) to'plami bilan X to'plam o'rtasida o'zaro bir qiymatli moslik o'rnatilgan. Obyektlar juftliklari o'rtasidagi C aloqalar to'plamiga esa xuddi shunday U qirralar to'plamiga mos qo'yilgan. Shaharlar o'rtasidagi masofa $(e_i, e_j)$ mos qirra $w(x_i, x_j)$ning vazni sifatida interpretatsiya qilinadi.

Shunday qilib, graflar nazariyasi atamalarida ko'rib chiqilayotgan masala bu gamilton grafida minimal uzunlikni qidirishdan iborat.

Masalaning formal qo'yilishi boshlang'ich G grafni Gc natija grafiga qayta o'zgartirishdan iborat:

$$G(X, <U, W>) \xrightarrow{D} G_C(X, U_C),$$

Shu sababdan :

$$U_C \subset U, |U_C| = |X_C| : W(G_C) = \sum_{u_r \in U_C} w(u_r) \to \min,$$

bu yerda

$$(\forall x_i \in X)\, p(x_i) = 2,\ (\forall x_i, x_i \in X)\, \exists\, S(x_i, x_i),$$

Bunda: $p_{(x_i)\text{-}x_j}$ graf uchiga keluvchi qirralar soni: a $S_{(x_i, x_j)}$- qirralar o'rtasidagi marshrut $(x_i, x_j)$, ya'ni ushbu qirralari bog'lovchi aralash qirralar ketma-ketligi.

Shuni nazarda tutish kerakki, agar har qanday juft qirralarning lokal darajasi yig'indisi (summasi) uning qirralari sonidan ko'p yoki teng bo'lsa graf gamiltonli siklga ega bo'ladi:

$$(\forall x_i,\ x_i \in X)\left[\rho(x_i) + \rho(x_j)\right] \geq n.$$

Aks holda grafa gamilton sikliga ega bo'lmasligi mumkin. Bu ayrim hollarda masalaning yechimiga ega bo'lmasligini anglatadi. Masala *NP* – murakkab masalalar guruhiga mansub bo'lib, eksponensial tavsifga ega. Ya'ni, masalalar hajmi oshishi bilan juda tez o'sib ketadi.

Masala *NP* – murakkab masalalar sifatiga tegishli bo'lib, hisoblash murakkabligi uning kirish (shaharlar soni) o'lchami polinom ko'rinishda ifodalanmagan eksponsial tasnifidir, ya'ni masala o'lchami ortishi bilan hisoblash murakkabligi tez o'sadi. Ma'lumki, kommivoyajer masalasining aniq yechimini to'la saralash, tanlash yoki soha va chegara usullari yordamida olish mumkin. Taqribiy yechimni esa chuqurligi bo'yicha qidirish va ikkilik o'rash usullari orqali olish mumkin.

Texnik topshiriqqa asosan aniq yechimni ta'minlash talab etilgani tufayli yuqorida ko'rsatilgan usullaridan hech bo'lmaganda birortasini realizatsiya qilish lozim. Usulni tanlash uchun qo'shimcha tadqiqot o'tkazish lozim.

Masalalarni hal etish usullarini aniqlab olgach, boshlang'ich ma'lumotlarning ayrim variantlari uchun qo'lda, kalkulyatorda yoki boshqa vositalardan foydalangan holda kutilayotgan natijani hisoblab chiqish maqsadga muvofiq. Ushbu ma'lumotlardan keyinchalik dasturiy ta'minotni testdan o'tkazishda foydalaniladi. Bundan tashqari, operatsiyani qo'lda bajarish harakat uzviyligini aniqlashga imkon beradi. Bu esa algoritmlar ishlanmasini soddalashtiradi.

Bundan tashqari, qanday boshlang'ich ma'lumotlar uyg'unlashuvi uchun natijalar mavjud emas yoki ushbu usul bilan olish mumkin emasligini o'ylab ko'rish lozim. Buni dasturiy ta'minotni ishlab chiqishda hisobga olish zarur.

**Nazorat savollari:**

1. Dasturlashga nisbatan tuzilmaviy yondashuv mohiyati nimadan iborat? Bunday yondashuv qanday bosqichlarni qamrab oladi?

2. «Spetsifikatsiya» atamasi nimani anglatadi? Ularni aniqlash murakkabligi nimada? Tuzilmaviy yondashuvda funksiyaviy tasniflash sifatida foydalaniladigan modellarni aytib bering?

3. Qanday vaziyat va holatlar o'tishi diagrammasidan foydalanish maqsadga muvofiq? Oldingi bob vazifalariga muvofiq siz tuzgan texnik vazifalarning kalkulyator uchun o'tish diagrammasini ishlab chiqing.

4. Funksional diagramma va ma'lumotlar oqimi diagrammasi o'rtasidagi asosiy farqlar nimada? Kalkulyatorning ichki xotirasidan foydalangan holda hisoblashni bajarish uchun diagrammaning har ikki ko'rinishini tuzing. O'xshash tomonlar va farqlarni tahlil qiling. Qanday hollarda ma'lumotlar oqimi diagrammasidan foydalanish maqsadga muvofiq?

5. «Ma'lumotlar tuzilmasi» deb nimaga aytiladi? Qanday ma'lumotlar nazarda tutiladi? Qanday hollarda ma'lumotlar tuzulmasini bayon etish lozim? Ma'lumotlar tuzulmasini ta'riflash uchun qanday modellardan foydalaniladi?

6. Taklif etilgan ma'lumotlar bayoni modellaridan foydalangan holda ularning ketma-ketligini yozing. Ushbu tuzilmaning qanday aspektlari yozilmadi va nima uchun?

7. Qanday hollarda matematik modellardan foydalaniladi? Modellar adekvatligi deganda nima tushuniladi? Adekvatligi dalilini keltirish nima uchun kerak va bunday dalillar qanday tuziladi?

# 5. TUZILMAVIY YONDASHUVDA DASTURIY TA'MINOTNI LOYIHALASHTIRISH

Yuqorida qayd etilganidek, tuzilmaviy yondashuv mohiyati dasturlarini dekompozitsiyalash yoki funksional tamoyil bo'yicha dasturiy tizimlardan iborat bo'ladi. Taklif etilayotgan barcha dekompozitsiya usullari oddiy interfeys turlaridan foydalanadi. Jumladan, sodda interfeyslar va an'anaviy menyu. Ular ma'lumotlar tuzilmasi va ularning qayta ishlovchi dasturini tahlil etish hamda loyihalashtirishga mo'ljallangan.

Ko'pgina hollarda qayta ishlanadigan komponentlarni loyihalashtirish boshlang'ich sanaladi. Ma'lumotlar tuzilmasini loyihalashtirish parallel ravishda bajariladi. Ayni paytda muqobil yondashuv mavjud. Unda ma'lumotlarni loyihalashtirish boshlang'ich sanaladi, qayta ishlovchi dastur esa ma'lumotlarning olingan tuzilmasi tahlil etilgan holda olinadi.

Har qanday holatda ham dasturiy ta'minotni loyihalashtirish uning tuzilmasini aniqlashdan boshlanadi.

## 5.1. Tuzilmaviy va funksional sxemani ishlab chiqish

Murakkab dasturiy ta'minotni loyihalashtirish uning tuzilmasini aniqlashdan, ya'ni tuzilmaviy komponentlar va ular o'rtasidagi aloqalarni aniqlashdan boshlanadi. Tuzilmani aniqlash natijalari tuzilmaviy va (yoki) funksional sxema hamda komponentlar bayoni ko'rinishida taqdim etilishi mumkin.

**Ishlab chiqilayotgan dasturiy ta'minotning tuzilmaviy sxemasi.** Tuzilmaviy sxema debganda, ishlab chiqilayotgan dasturiy ta'minotning tarkibi va uning qisimlarini boshqarish bo'yicha o'zaro xarakterlar tushuniladi.

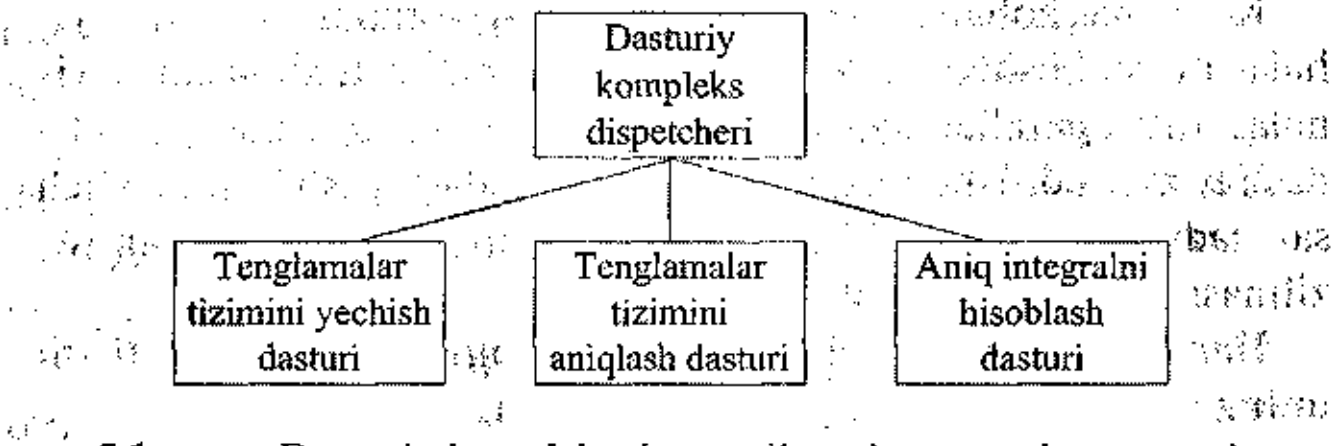
Dasturlar paketining tuzilmaviy sxemasi informativ emas, paketda dasturlarni tashkil etish ularni boshqarishni nazarda tutmaydi. Shu bois tuzilmaviy sxema har bir paket dasturi uchun ishlab chiqiladi.

Dasturiy ta'minotning eng oddiy turi — bu tuzilmaviy komponent sifatida faqat qism dastur va resurslar kutubxonasini kiritishi

mumkin. Dasturning tuzilmaviy dasturini ishlab chiqish qadam-baqadam detallashtirish usuli orqali bajariladi (5.2-mavzuga qarang).

Dasturiy tizimning yoki dasturiy kompleksning tuzilmaviy komponentlari sifatida dasturlar, tizimosti, ma'lumotlar bazasi, resurslar kutubxonasi va hokazolar xizmat qilishi mumkin.

Dasturiy kompleksning tuzilmaviy sxemasi boshqaruvini dasturning distpetcheridan tegishli dasturga uzatishni namoyon qiladi (5.1-rasm).




**5.1-rasm.** Dasturiy kompleksning tuzilmaviy sxemasi namunasi.

Dasturiy tizimning tuzilmaviy sxemasi odatda, tizimosti yoki boshqa tuzilmaviy komponentlar mavjudligini ko'rsatadi. Dasturiy kompleksdan farqli ravishda dasturiy tizimning alohida qismlari (tizimosti) muntazam ravishda o'zaro ma'lumot almashadi. Dasturiy tizimning tuzilmaviy sxemasi odatda buni aks ettirmaydi (5.2-rasm).

Loyihalashtirilayotgan dasturiy ta'minot to'g'risidagi yanada to'liq tasavvurni funksional sxema beradi.

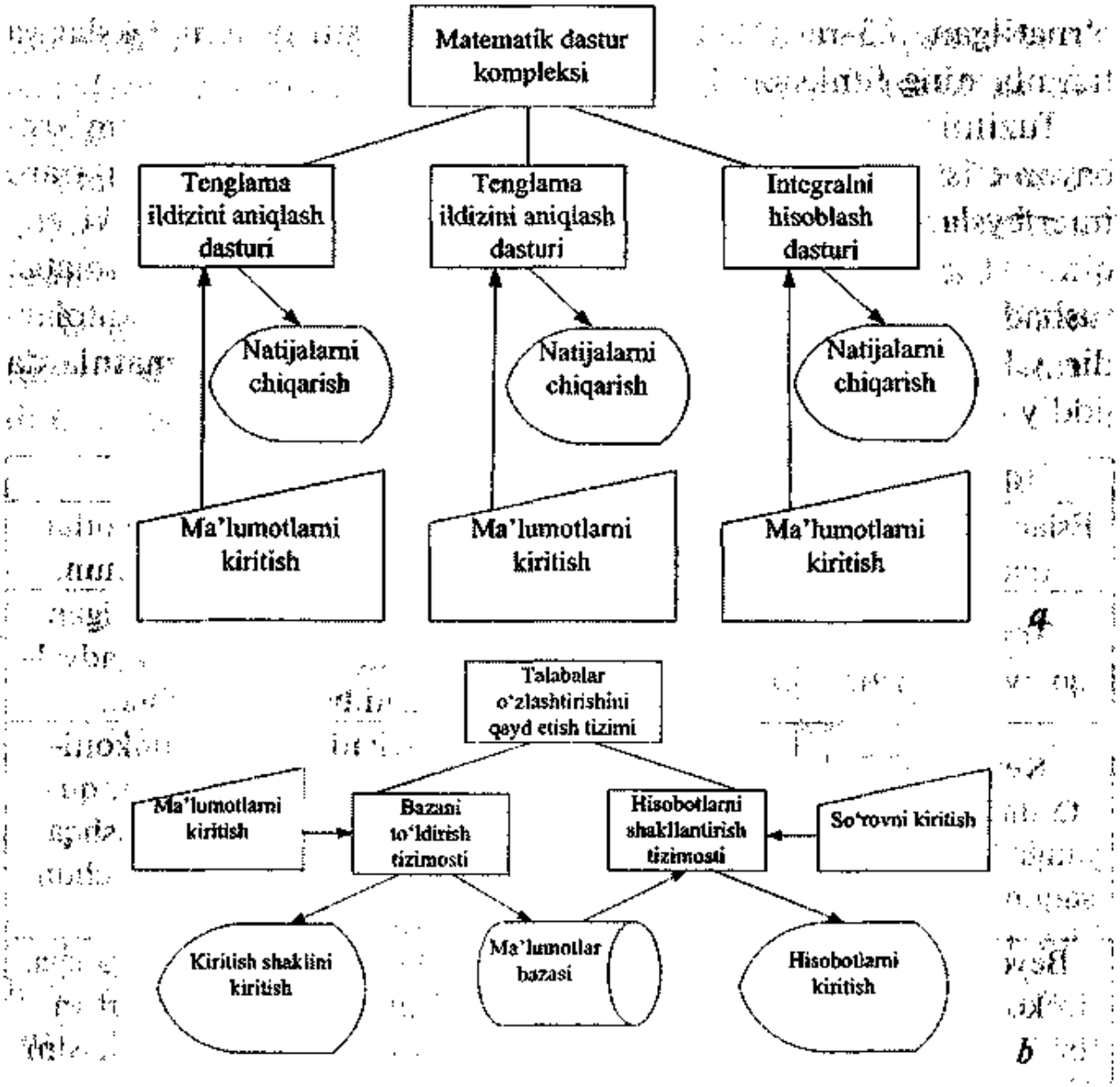
*Funksional tizim.* Funksional sxema yoki ma'lumotlar sxemasi (GOST 19.701–90) bu dasturiy ta'minot komponentlarining o'zaro harakati bo'lib, axborot oqimlari bayoni, oqimdagi ma'lumotlar tarkibi va foydalaniladigan fayllar hamda qurilmalar ko'rsatiladi. Funksiyaviy sxemalarni tasvirlash uchun maxsus belgilardan foydalaniladi. Bu belgilar GOST 19.701–90 bo'yicha ma'lumotlar sxemasining asosiy belgilari standarti tomonidan

o'rnatilgan. 5.3-rasmda taqqoslash uchun dasturiy komplekslar va tizimlarning funksional sxemasi keltirilgan.

Tuzilmaviy va funksional sxemalarning barchasi tizimlarni bayon etishi lozim. Tuzilmaviy yondashuvda ayniqsa, dasturlararo interfeyslar spetsifikatsiyasini ishlab chiqishi lozim. Chunki eng qimmat xatolar soni bayon etish sifatiga bog'liq. Eng qimmatga tushadigan xatolar kompleks test paytidan aniqlanadigan xatolardir. Chunki ularni bartaraf etish uchun bekor qilingan matnlarda jiddiy o'zgarish qilishni talab etishi mumkin.

| Blok nomi | Belgilar | Blok vazifasi |
|---|---|---|
| Eslab qolinadigan ma'lumotlar | | Jadval va boshqa ma'lumotlar tuzilmasini belgilash uchun, |
| Tezkor eslab qoluvchi qurilma | | Tezkor xotirada saqlanadigan ma'lumotlar tuzilmasi va jadvallarini belgilash uchun. |
| Keyinchalik tanlash imkoniyatiga ega xotirada saqlovchi qurilma | | Keyinchalik tanlash imkoniyatiga ega bo'lgan, jadval qurilmada saqlanadigan boshqa ma'lumotlarni belgilash uchun (magnit tasmada). |
| Bevosita kirish imkoniyatiga ega bo'lgan, xotirada saqlovchi qurilma | | Bevosita kirish imkonyatiga ega, qurilmada (diskda) jadval va boshqa ma'lumotlar tuzilmasini belgilash uchun. |
| Hujjat | | Jadval va bosib chiqaruvchi qurilmaga chiqariladigan boshqa tuzilmalarni belgilash uchun. |
| Qo'lda kiritish | | Ma'lumotlarni qo'lda klaviatura orqali belgilash uchun. |
| Xarita | | Magnit yoki perforir xaritalarda ma'lumotlarni belgilash uchun. |
| Displey | | Kompyuter displeyiga chiqariladigan ma'lumotlarni belgilash uchun. |

**5.2-rasm.** Dasturiy tizimning tuzilmaviy sxemasi.




**5.3-rasm.** Funksional sxemalar namunalari: a – dasturlar kompleksi; b – dasturiy tizim,

## 5.2. Dasturiy ta'minot tuzilmasini loyihalashtirish uchun qadam-baqadam detallashtirish usulidan foydalanish

XX asrining 70-yillarda shakllangan dasturlashga tuzilmaviy yondashuv dasturini dekompozitsiyalashni qadam-baqadam detallashtirish usuli orqali amalga oshirish taklif etilgan. Dekompozitsiya natijasi dasturning tuzilmaviy sxemasi sanaladi. U boshqarish bo'yicha qism dastur (dastur qismi)lar o'zaro ta'sirining ko'p darajali iyerarxik sxemasini ifodalaydi.

Qadam-baqadam detallashtirish usuli (1.4-mavzuga qarang) pastlashuvchi yondashuvini amalga oshiradi. Mazkur usul algoritmning qadamlar bo'yicha ishlab chiqishni nazarda tutadi va tartibli (tuzilmaviy) dasturlashga asoslanadi. Bunda har bir qadam funksiyalarning tagfunksiyalarga bo'laklanishini o'z ichiga oladi. Binobarin, birinchi bosqichda umumiy tagmasalalarni ajratgan holda qo'yilgan masalaning yechimi bayon etiladi, keyingisida o'xshash tarzda tagmasalalar yechimi keyingi daraja tagmasalalarni shakllantirgan holda bayon etiladi. Shu tariqa har bir qadamda loyihalanayotgan dasturiy ta'minot funksiyalarining aniqlashtirish yuz beradi. Jarayon yechimi algoritmlari ayon tagmasalaga yetgunga qadar davom ettiriladi.

Dasturni qadam-baqadam detallashtirish usuli bilan dekompozitsiyaning tuzilmaviy dekompozitsiyaning vertikal boshqaruv tamoyilidan kelib chiquvchi asosiy qoidasiga rioya qilish lozim: birinchi navbatda dekompozitsiyalanayotgan komponentning boshqaruv jarayonlarini detallashtirish, bunda ma'lumotlar bilan operatsiyalarni aniqlashtirishni keyinga qoldirish kerak. Bu boshqaruv jarayonlarining ustuvor detallashtirish iyerarxiyaning barcha darajasidagi komponentlar tuzilmasini mohiyatan soddalashtirishi va qaror qabul qilish jarayonini uning bajarilishidan ajratmaslikka imkon berishi bilan bog'liq: xususan, qandaydir muqobil tanlov shartini belgilagach, darhol uni amalga oshiruvchi modul chaqiriladi.

Tuzilmalar bilan operatsiyalarni detallashtirish oxirgi navbatda ularning spetsifikatsiyalari aniqlashtirilishini keyinga qoldirishga imkon beradi va tegishli ma'lumotlarga qaram modullar miqdorini qisqartirish hisobidan mazkur tuzilmalarni nisbatan og'riqsiz modifikatsiyalash imkoniyatini ta'minlaydi.

Bundan tashqari, quyidagi tavsiyalarga rioya qilish maqsadga muvofiq:

- initsializatsiya va yakunlash operatsiyalarini tegishli ishlovdan ajratmaslik kerak, chunki initsializatsiya va yakunlash modullari yomon aloqaviylikka (muvaqqat) hamda kuchli ulashuvga (boshqaruv bo'yicha) ega;

• ortiqcha ixtisoslashgan yoki ortiqcha universal modullarni loyihalashtirmaslik darkor, chunki haddan ziyod maxsus modullarni loyihalash ularning miqdorini ko'paytiradi, haddan ziyod universal modullarni loyihalash esa ularning murakkabligini orttiradi;

• turli modullarda amallarning dubllanishidan qochish lozim, zero ularning o'zgarishida dasturning ular bajaradigan barcha fragmentlariga tuzatishlar kiritishga to'g'ri keladi, bunday holatda ushbu amallarni alohida modulda amalga oshirish maqsadga muvofiq;

• xatolar haqidagi xabarlarni resurslar kutubxonasi tipi bo'yicha bitta modulga guruhlashtirish kerak, shunda ta'birlarni muvofiqlashtirish yengil bo'ladi, xabarlarning dubllanishidan saqlanadi, shuningdek xabarlarni boshqa tilga tarjima qilish yengillashadi.

Har bir masalaning yechimini bayon etishda sikl, hozircha yoki tarmoqlanish kabi ko'pi bilan bir-ikkita tuzilmaviy boshqaruv konstruksiyalardan foydalangan ma'qul, bu esa tashkillashtiruvchi hisoblash jarayoni tuzilmasini aniq tasavvur qilishga imkon beradi.

**5.1-misol.** Argumentlarning o'zgarish intervali $[x_1,x_2]$da bir o'zgaruvchili funksiyani uning butun intervaldan uzluksizlik shartini hisobga olgan holda mazkur funksiya grafigini qurish dasturining algoritmi ishlab chiqilsin. Bir o'zgaruvchili berilgan argument o'zgarishlari [x1, x2] intervalida funksiyalari grafiklarining butun o'zgarish intervalida funksiyaning uzluksiz bo'lishi sharti bilan tuzilishi dasturi algoritmi ishlab chiqilsin.

**Eslatma.** Funksiyalar grafiklari tuzilishini birinchi va ikkinchi qator uzilish nuqtalari bilan birgalikda dasturlash uchun topshirilgan funksiyalarni tahliliy tatqiq qilish zarur, bu esa alohida va yetarlicha murakkab masalani ifoda etadi. Ushbu masala sonli usullar bilan yechilmaydi.

Umumiy ko'rinishda funksiya grafigini tuzish masalasi qandaydir miqyosda bajarilgan real grafikni (5.4, a-rasm) ekran oy-

nasidagi tegishli tasvirda (5.4, b-rasm) aks ettirish masalasi sifatida qo'yiladi.

Grafikni tuzish uchun koordinatalar o'qlari bo'yicha masshtablarni belgilash zarur:

$$m_x = \frac{wx_2 - wx_1}{x_2 - x_1}, \qquad m_y = \frac{wy_2 - wy_1}{y_{max} - y_{min}}$$

Shuningdek, grafik nuqtalari koordinatalari ham belgilanadi:

$$px_i = ](x_i - x_1) \cdot m_x[ + wx_1,$$
$$py_i = ](y_{max} - y_i) * m_y[ + wy_1.$$

Bunda koordinata to'rning vertikal va gorizontal bo'yicha qadamini quyidagi formulalar bo'yicha aniqlash mumkin:

$$lpm_x = \frac{wx_2 - wx_1}{nl_x}, \qquad lpm_y = \frac{wy_2 - wy_1}{nl_y},$$

$n_lx$, $n_ly$ – tegishli ravishda vertikal va gorizontal chiziqlar miqdori.

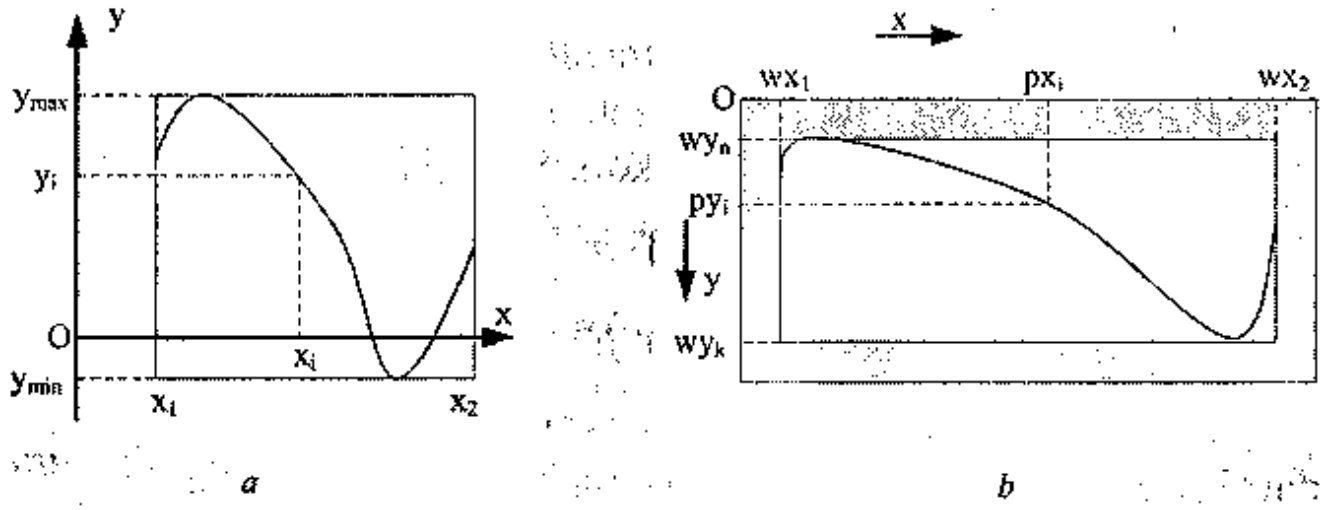
To'rni o'lchash uchun o'lchashning vertikal va gorizontal bo'yicha qadamlarini aniqlash zarur:

$$lm_x = \frac{x_2 - x_1}{nl_x}, \qquad lm_y = \frac{y_{max} - y_{min}}{nl_y}.$$

Shunday qilib, grafik tuzish uchun funksiya, shuningdek funksiya uzluksiz bo'luvchi argument o'zgarishlari $[x_1, x_2]$ intervali, grafik nuqtalari miqdori n, grafik tuzilishi zarur ekran oynasi o'lchami va holati: $wx_1$, $wy_1$, $wx_2$, $wy_2$ hamda gorizontal va vertikal bo'yicha to'r chiziqlari miqdori nlx, nly topshirilishi darkor. $wx_1$, $wy_1$, $wx_2$, $wy_2$, nlx, nly birliklarini ekran o'lchamidan kelib chiqqan holda topshirish mumkin, interval va grafik nuqtalari sonini esa kiritish kerak.

Algoritm ishlab chiqilishini yozish uchun psevdokoddan foydalangan holda qadam-baqadam detallashtirish usuli bilan bajaramiz.

Dastur foydalanuvchi bilan funksiya, kesma, qadam, natija ko'rinishi, bajarilsin, chiqish punktlariga ega an'anaviy iyerarxik menyu (5.4-rasmga qarang) orqali o'zaro harakatlanadigan bo'lishini qabul qilamiz.




**5.4-rasm.** Funksiya grafigini tuzish masalasi:
a – funksiya grafigi; b – uning ekran oynasidagi tasviri.

**1-qadam.** Alohida holatda menyu bilan ishni klaviatura orqali amalga oshiradigan boshqaruv dasturi tuzilmasini belgilaymiz:

**Dastur**
*Global birliklar initsiallashtirilsin*
*Sarlavha va menyu kiritilsin*
**Bajarilsin**
*Agar Buyruq tanlanilsa*
*u holda Buyruq bajarilsin*
*boshqacha Boshqaruv klavishini bosish ishlanilsin*
*hammasi-agar*
*to Buyruq=Chiqish*
**Tamom.**

Ekranni tozalash, sarlavha va menyuni chiqarish, shuningdek buyruqning tanlovi nisbatan sodda operatsiyalar, binobarin ularni detallashtirmaslik ham mumkin.

**2-qadam.** Buyruq bajarilsin operatsiyasini detallashtiramiz:

**Buyruq bajarilsin:**

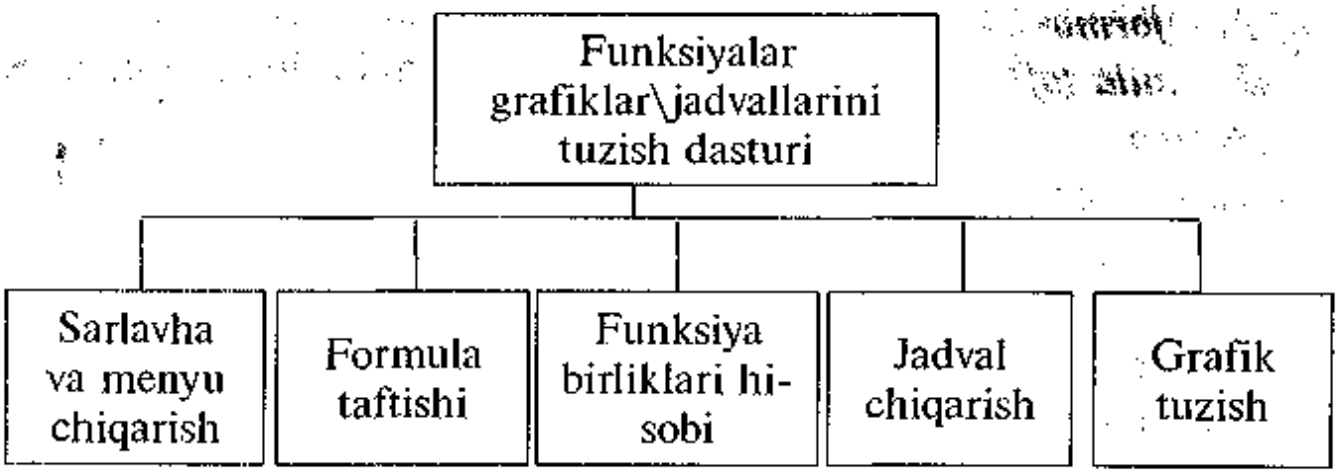
**Tanlov** *Buyruq*
*Funksiya:*
*Fun formulasi kiritilsin yoki tanlanilsin*
*Formula taftishi bajarilsin*
*Kesma:*
*$x_1$, $x_2$ birliklari kiritilsin*
*Qadam:*
*H birligi kiritilsin*
*Natija ko'rinishi:*
*Natija ko'rinishi kiritilsin*
*Bajarilsin:*
*Funksiya birliklari hisoblansin*
**Agar** *Natija ko'rinishi = Grafik*
**u holda** *Grafik tuzilsin*
**boshqacha** *Jadval chiqarilsin*
**Hammasi-agar**
**Hammasi-tanlov**

Qaysi fragmentlarni tagdasturlar tarzida amalga oshirish ma'noga egaligini aniqlab olamiz. Birinchidan, sarlavha va menyu chiqarish fragmenti, chunki bu operatorlarning yetarlicha uzun yo'nalishi bosqichliligi va uning alohida protseduraga ajratilishi boshqaruvchi dasturni qisqartirishga imkon beradi. Ikkinchidan, Formula taftishi, funksiya birliklari hisobi, grafik tuzilishi va jadval chiqarish fragmentlari, chunki bular yetarlicha murakkab operatsiyalar. Bu dastur tuzilmasini belgilaydigan / birinchi daraja tagdasturlaridar (5.5-rasm). Ushbu tagdasturlar uchun asosiy dasturdan interfeyslarni, ya'ni parametrlar ro'yxatlarini belgilaymiz.

Sarlavha va menyuni chiqarish tagdasturi parametrlarni nazarda tutmaydi. Formula taftishi tagdasturi ikki parametrga ega bo'lishi shart: Fun — funksiyaning tahliliy topshirig'i, Tree — taftish daraxtining qaytariluvchi parametr — manzili.

Funksiya birliklari hisobi tagdasturi Tree taftish daraxti manzili, $x_1$ va $x_2$ birliklar kesmasini, shuningdek h qadamni olishi

shart. U funksiya nuqtalari miqdori n bo'lgan X(n) va Y(n) funksiya birliklari jadvalini dasturga qaytarishi shart.



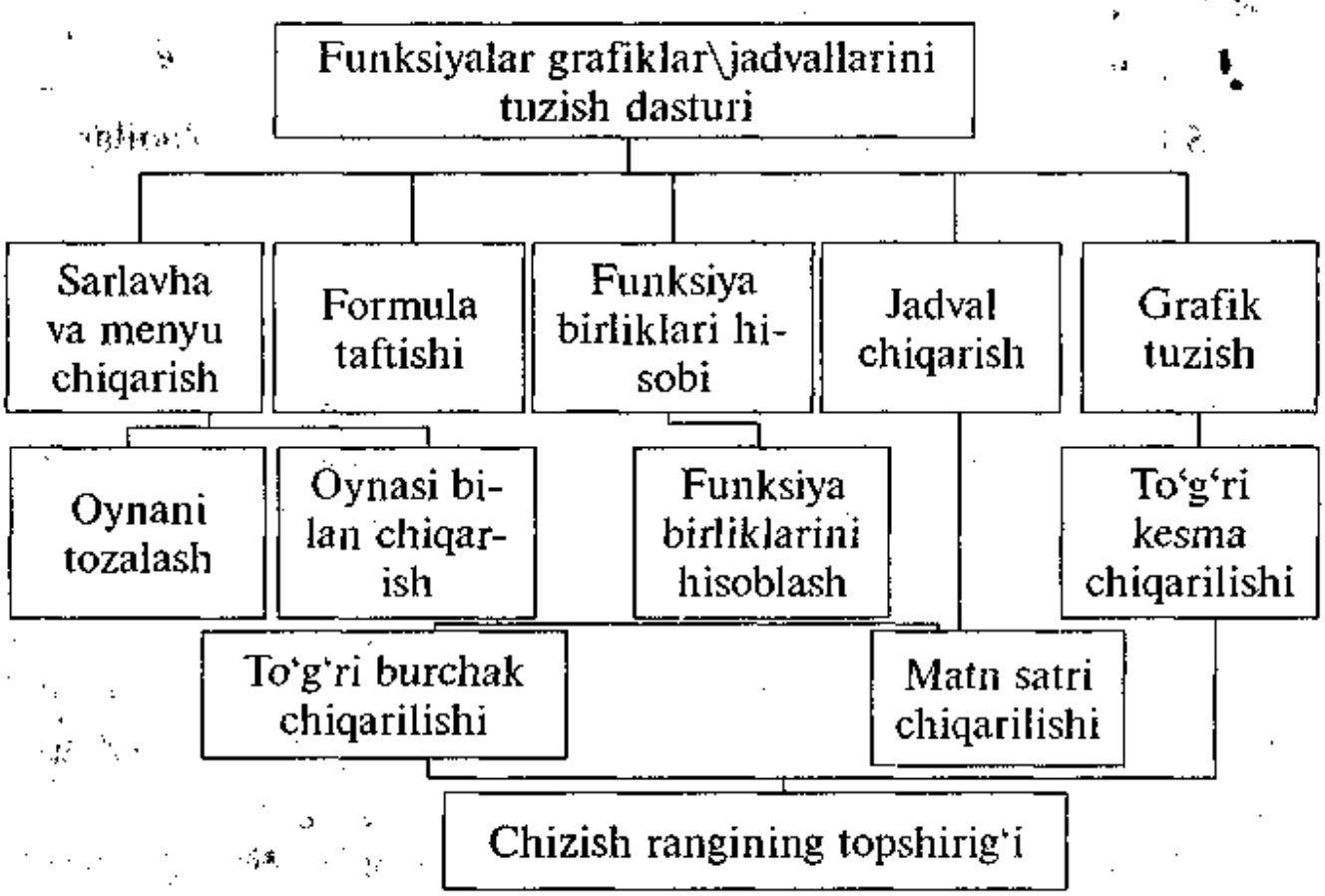

**5.5-rasm.** Funksiyalar grafiklari va jadvallarini tuzish dasturining tuzilmaviy sxemasi.

Jadval chiqarish va grafik tuzish tagdasturlari funksiya birliklari hamda nuqtalar miqdori jadvalini olishlari shart.

O'zgaruvchilarning nomlari aniqlashtirilgandan so'ng asosiy dastur algoritmi quyidagicha ko'rinishda bo'ladi:

**Dastur.**
*Sarlavha va menyu chiqarish*
*Bajarilsin*
*Agar Buyruq tanlanilsa*
*u holda*
*Tanlov Buyruq*
*Funksiya:*
*Fun formulasi chiqarilsin yoki tanlanilsin*
**Formula taftishi (Fun; Var Tree)**
*Kesma:*
$x_1$, $x_2$ *birliklari kiritilsin*
*Qadam:*
*h birligi kiritilsin*
*Natija ko'rinishi:*
*Natija ko'rinishi kiritilsin*
*Bajarilsin:*

*($x_1$, $x_2$, h, Tree; Var X, Y, n) funksiya birliklari hisobi*
**Agar** *Natija ko'rinishi=Grafik*
**u holda (X, Y, p) grafigi tuzilishi**
**boshqacha (X, Y, p) jadvali chiqarilishi**
**Hammasi-agar**
**Hammasi-tanlov**
boshqacha *Boshqaruv klavishini bosish ishlanilsin*
**Hammasi-agar**
*To Buyruq=Chiqish*
**Tamom.**



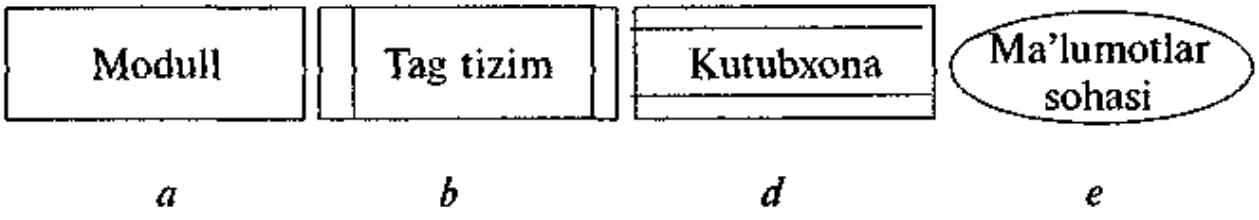

**5.6-rasm.** Grafiklarni/jadvallarni tuzish dasturining to'liq ko'p darajali tuzilmaviy sxemasi.

## 5.3. Konstantayn tuzilmaviy xaritalari

Keyingi qadamlarda tagdasturlar algoritmlarini detallashtirish bajarilishi zarur. Dastur algoritmi to'liq tushunarli bo'lgunga qadar detallashtirish bajaraveriladi. Mazkur dastur to'liq tuzilma-

viy sxemasining mumkin bo'lgan variantlaridan biri 5.6-rasmda ko'rsatilgan.

Yuqorida eslatib o'tilganidek, qadam-baqadam detallashtirish usulidan foydalanish ishlab chiqilayotgan dasturiy ta'minotning yuqori darajadagi texnologikligini ta'minlaydi, zero u faqat boshqaruvni berishning tuzilmaviy usullaridan foydalanishga imkon beradi.




**5.7-rasm.** IBM, ISO va ANSI standartlari bo'yicha cho'qqilar belgilanishi: a – modul; b – tagtizim; d – kutubxona; e – ma'lumotlar sohasi.

Loyihalashning ushbu turida modullarga bo'laklash tavsiya etilgan modullar o'lchamlaridan (20–60 satrdan) va tuzilma murakkabligidan (ikki-uchta sarflangan boshqaruvchi konsruksiyalardan) kelib chiqqan holda evristik ravishda bajariladi. Tamoyil jihatidan modul (tagdastur) sifatida detallashtirish jarayonning har bir qadamida shakllantirilgan tagmasalalar yechimini amalga oshirish mumkin, biroq dasturni modullarga bo'laklashda ko'rib chiqilgan modullar texnologikligini ta'minlash prinsiplari hal qiluvchi ahamiyat kasb etadi.

Olingan modullar iyerarxiyasi texnologikligining tahlili uchun konstantayn yoki Jekson tuzilmaviy xaritalaridan foydalanish maqsadga muvofiq.

Tuzilmaviy xaritada modullar o'rtasidagi munosabatlar graf tarzida ifodalanadi hamda uning uchlariga – ma'lumotlar modullari va umumiy sohalari, yoylariga esa modullararo chaqiruvlar va ma'lumotlar umumiy sohalariga murojaatlar muvofiq bo'ladi.

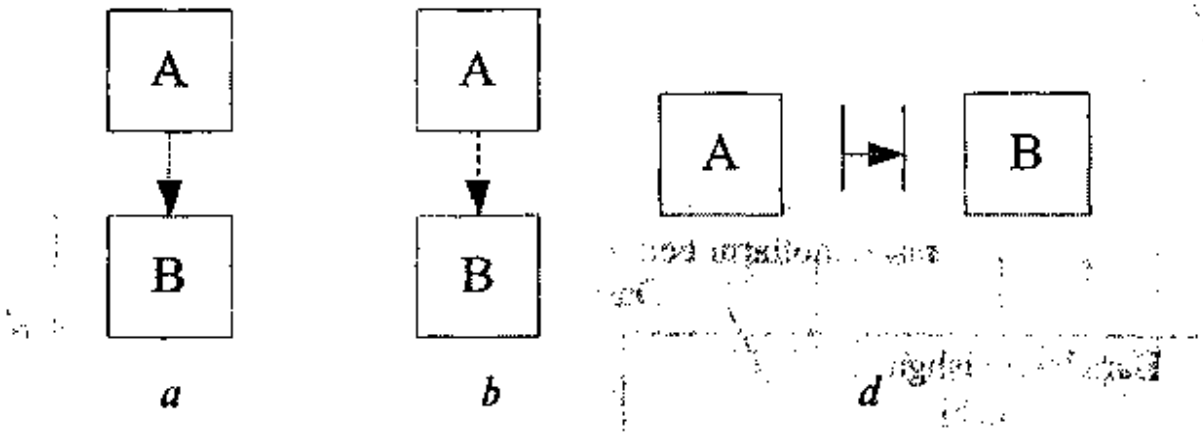
Uchlarning to'rtta tipi farqlaniladi (5.7-rasm):

- modul – tagdastur;

• tagtizim – dastur;

• kutubxona – alohida modulda joylashtirilgan tagdasturlar jamlanmasi;

• ma'lumotlar sohasi – tashqaridan chaqiruv mumkin bo'lgan maxsus yo'sinda dasturlashtirilgan ma'lumotlar jamlanmasi.

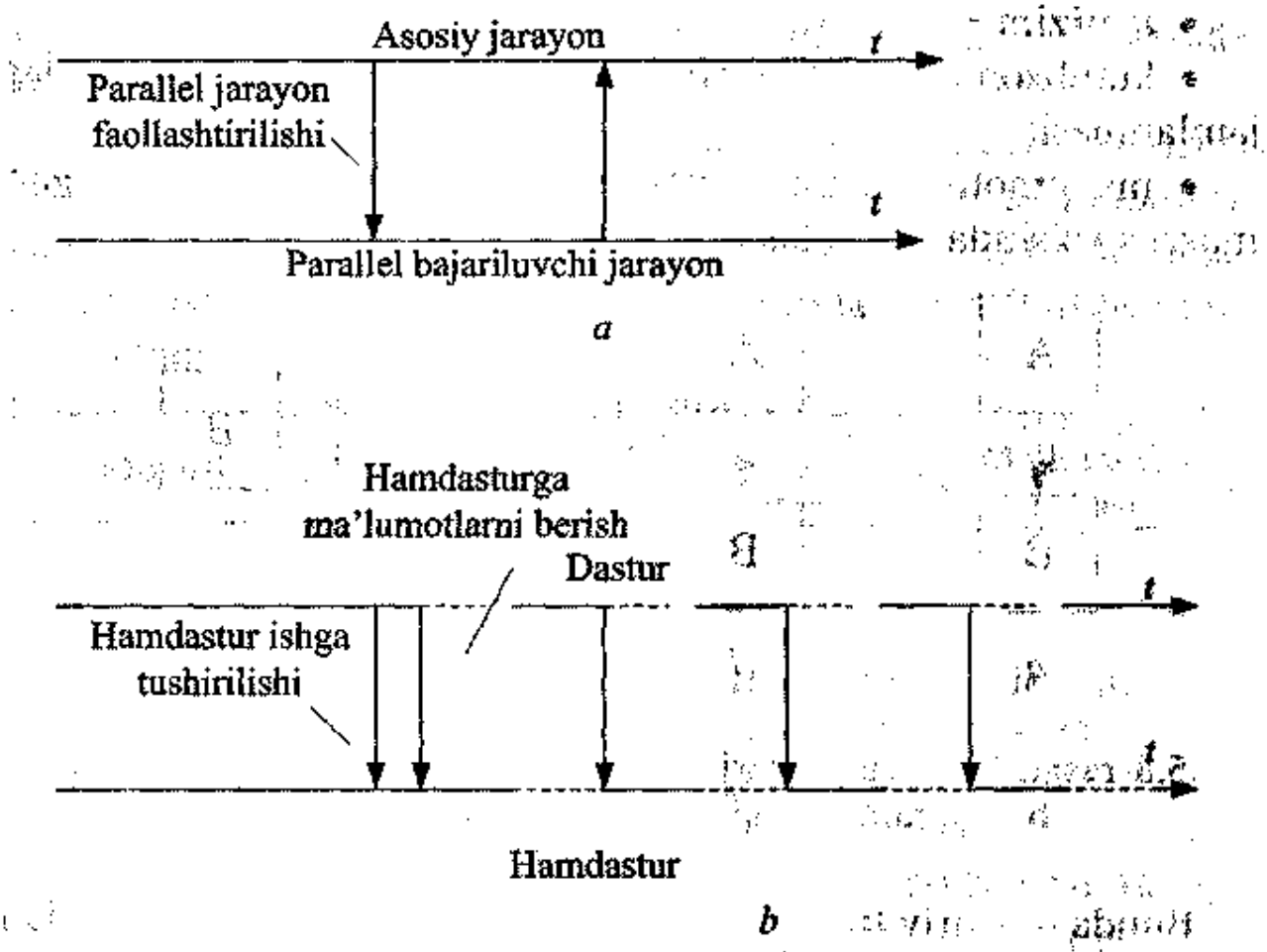



**5.8-rasm.** Chaqiruv tiplari belgilanishi: a – bosqichli chaqiruv; b – parallel chaqiruv; d – hamdastur chaqiruv.

Bunda dasturiy tizimning (dasturning, tagdasturning) alohida qismlari bosqichli, parallel yoki dasturdoshlar sifatida chaqirilishi mumkin. Bosqichli chaqiruv eng ko'p qo'llaniladiki, bunda modullar boshqaruvni berib, uzilgan ishlovni davom ettirish uchun chaqirilgan dastur yoki tagdastur bajarilishining tugallanishini kutib turadi.

Parallel chaqiruv deyilganda hisoblashlarni bir necha hisoblovchilarda parallellashtirish tushuniladi va bunda boshqa jarayon faollashtirilganda tegishli jarayon ishni davom ettiradi (5.9, a-rasm). Bir protsessorli kompyuterlarda multdasturlar muhitida ushbu holatda tegishli dasturlarning birin-ketin bajarilishi boshlanadi. Parallel jarayonlar sinxron va asinxron bo'ladi. Sinxron jarayonlar uchun sinxronlashtirishning nuqtalari, jarayonlar o'rtasidagi axborot almashinuvi hosil qildiriladigan vaqt momentlari belgilanadi.

Asinxron jarayonlar faqat parallel jarayon faollashtirilishi momentida axborot bilan almashinadi.



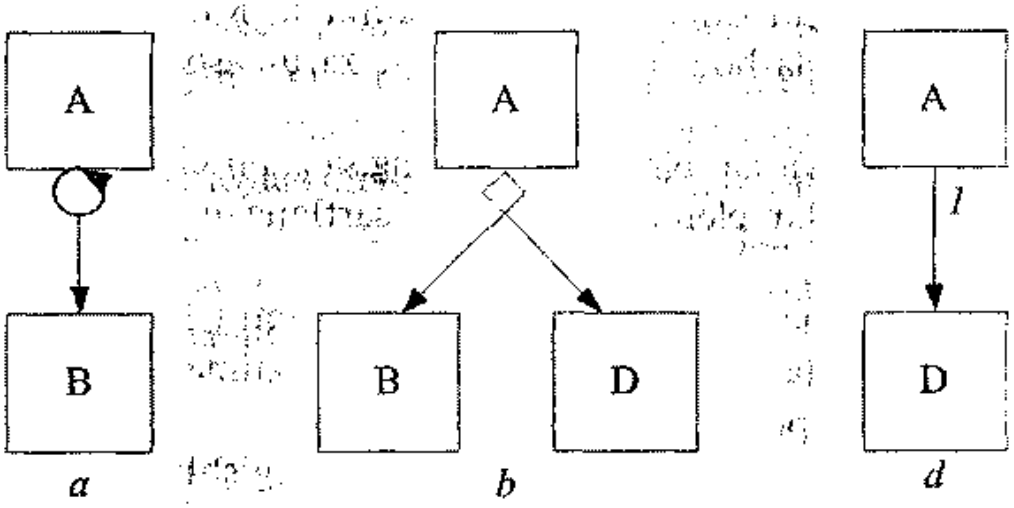

**5.9-rasm.** Parallel chaqiruv (a) va hamdastur chaqiruvi (b) amalga oshirilishi diagrammalari: bajarilish; kutish.

Hamdastur chaqiruvi deyilganda bir vaqtning o'zida ishga tushirilgan ikki dasturning galma-gal bajarilishi tushuniladi, masalan bir dastur chiqarilishi uchun ma'lumotlar paketini tayyorlagan bo'lsa, ikkinchisi uni chiqarishi, so'ngra keyingi paketni kutish holatiga o'tishi mumkin. Bunda multidasturiy tizimlarda asosiy dastur ma'lumotlarni bergandan so'ng ishlashni davom ettiraveradi, 5.9, b-rasmda tasvirlangani kabi kutish holatiga o'tmaydi.

Agar chaqiruvni tasvirlovchi strelka blokka tegsa, u holda murojaat modulga yaxlit o'tadi, blokka kirganda esa modul ichidagi elementga o'tadi.

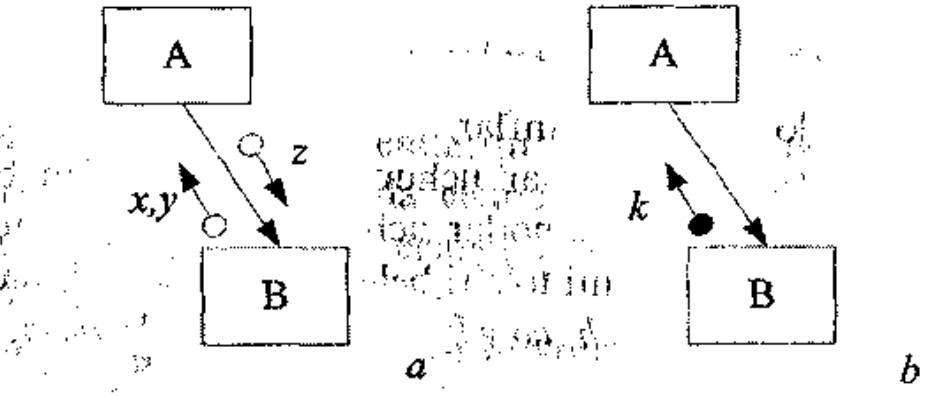
Zaruriyatga ko'ra tuzilmaviy xaritada chaqiruvning alohida shartlarini aniqlashtirish mumkin (5.10-rasm): siklik chaqiruv,

shartli chaqiruv va bir martalik chaqiruv – asosiy modul takroran chaqirilganda bir martalik chaqiruvli modul faollashtirilmaydi.




**5.10-rasm.** Chaqiruv alohida shartlari belgilanishi: a – siklik; b – shartli; d – bir martalik.

Ma'lumotlar va boshqaruv bo'yicha aloqalar bilan belgilanadi, strelka yo'nalishi aloqa yo'nalishini ko'rsatadi (5.11-rasm).




**5.11-rasm.** Aloqa tipi belgilanishi: a – ma'lumotlar bo'yicha; b – boshqaruv bo'yicha.

Konstantayn tuzilmaviy xaritalari dasturning modullarga dekompozitsiyasi natijasini ko'rgazmali taqdim etishga va uning sifatini, ya'ni tuzilmaviy dasturlash tavsiyalariga muvofiqligini baholashga imkon beradi (ulashuv va aloqaviylik).

## 5.4. Ma'lumotlar tuzilmalarini loyihalash

Ma'lumotlar tuzilmalarini loyihalash deyilganda ularning xotirada taqdimlarini ishlab chiqish tushuniladi. Ma'lumotlar tuzilmalarini loyihalashda hisobga olinishi zarur asosiy parametrlar quyidagilardir:

- ma'lumotlar har bir elementlarning saqlanuvchi axboroti turi;
- ma'lumotlar elementlarning va sarflangan tuzilmalarning aloqalari;
- tuzilma ma'lumotlarining saqlanish vaqti («hayot vaqti»);
- ma'lumotlar elementlari, sarflangan tuzilmalar va umuman tuzilma bilan operatsiyalar jamlanmasi.

*Saqlanuvchi axborot turi* xotiraning tegishli tipini belgilaydi. Foydalanilayotgan dasturlarning tipiga bog'liq ravishda ma'lumotlar elementlari sifatida quyidagilar qaralishi mumkin:

- turli formatdagi butun va haqiqiy sonlar;
- timsollar (simvollar);
- Bull qiymatlari: true va false.

Shuningdek, ma'lumotlarning ayrim tuzilmaviy tiplari, masalan:

- satrlar;
- yozuvlar;
- maxsus e'lon qilingan sinflar.

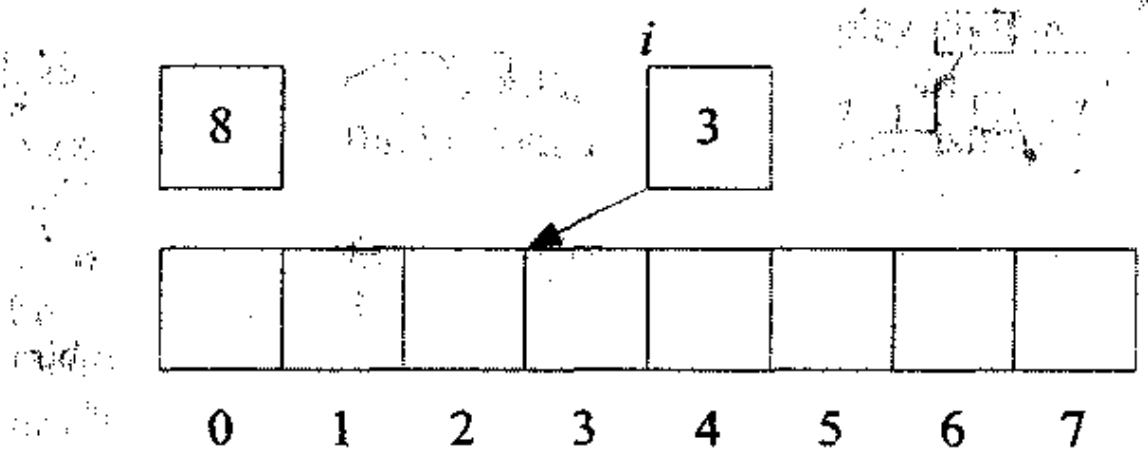
Bu o'rinda sonli maydonlar uchun mumkin bo'lgan birliklar diapazonini, satrli ma'lumotlar uchun esa satrning mumkin bo'lgan maksimal uzunligini to'g'ri belgilash g'oyat muhimdir.

*Elementlar va ichma-ich qo'yilgan tuzilmalar aloqalari*, shuningdek ularning barqarorligi hamda elementlar va sarflangan tuzilmalar bilan operatsiyalar jamlanmasi ma'lumotlar taqdimi uchun foydalaniluvchi xotira tuzilmalarini ham belgilaydi. Ma'lumotlarni statik yoki dinamik, shuningdek tashqi xotiraga joylashtirishda hayot vaqti hisobga olinadi.

Ma'lumotlar, ularning elementlari va ular o'rtasidagi aloqalar ichki taqdimining mavjud variantlarini yanada batafsil ko'rib chiqamiz.

*Ma'lumotlarning operativ xotiradagi taqdimi.* Ma'lumotlarni operativ xotirada tashkillashtirishning ikki tayanch tuzilmasi – vektorli va ro'yxatli tuzilmalar farqlaniladi.

*Vektorli tuzilma* ma'lumotlar maydonlarini joylashtirish uchun foydalaniladigan xotira baytlari bosqichliligini o'zida ifoda etadi (5.12-rasm). Ma'lumotlarning tashkillashtirilgan tuzilmalarini bosqichlilikda joylashtirish elementlarga to'g'ridan-to'g'ri daxlni: massivlarda yoki satrlarda indeks (indekslar jamlanmasi) bo'yicha yoki yozuvlarda yoxud obyektlarda maydon nomi bo'yicha amalga oshirish imkonini beradi.



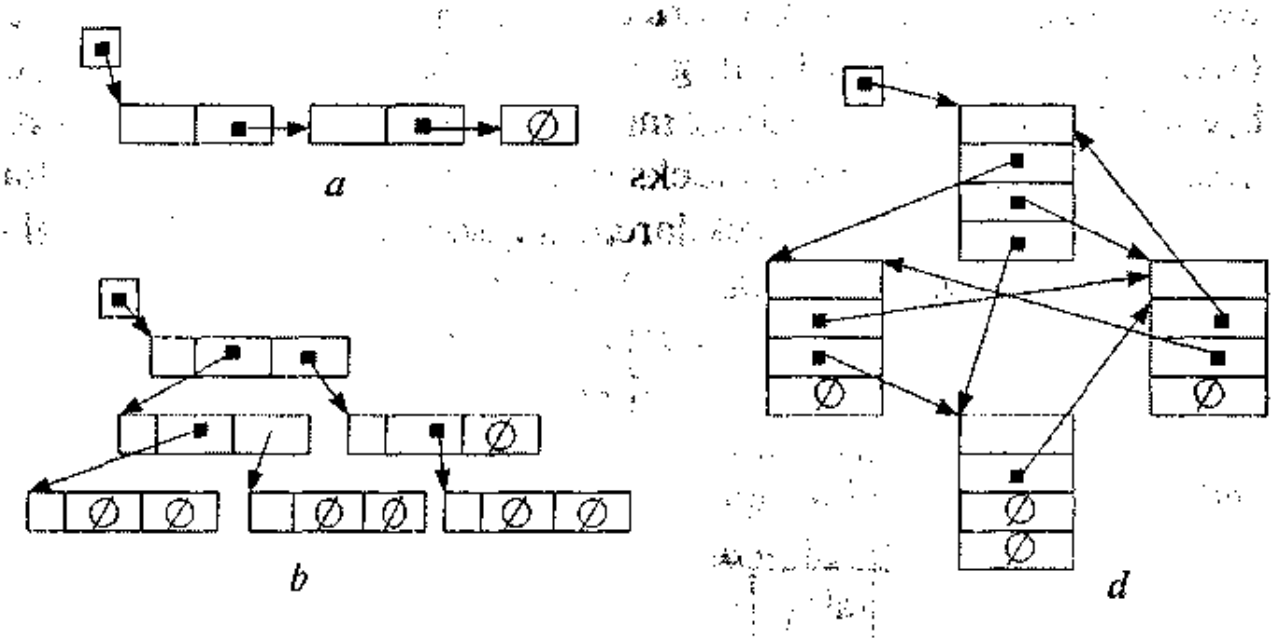

**5.12-rasm.** Xotiraning vektorli tuzilmasi.

Biroq massivlar elementlarini joylashtirish uchun vektorli tuzilmalardan foydalanishda elementlarni qo'shish yoki yo'qotish operatsiyalarini bajarish elementlarning ko'p oshirilishini talab qilish mumkin.

Ma'lumotlar tuzilmalarini vektorli taqdimda ham statik, ham dinamik xotiraga joylashtirish mumkin. Vektorli taqdimlarning dinamik xotiradagi joylashuvi ba'zan operativ xotiradan foydalanish samaradorligini jiddiy ravishda orttirishga imkon beradi. Dinamik xotirada oraliq natijalarni saqlovchi muvaqqat tuzilmalarni hamda o'lchami kiritiluvchi boshlang'ich ma'lumotlarga kuchli bog'liq tuzilmalarni joylashtirgan ma'qul.

*Ro'yxatli tuzilmalar* axborot qism bilan birga yana bir yoki bir necha ko'rsatkichlarni – elementlarning yoki shu elementlarga

aloqador sarflangan tuzilmalarning manzillarini ham qamrab oluvchi maxsus elementlardan barpo etiladi. Bunday elementlarni dinamik xotirada joylashtirgan holda turli ichki tuzilmalarni tashkillashtirish mumkin (5.13-rasm).




**5.13-rasm.** Xotira ro'yxatli tuzilmalari misollari: a – yo'nalishli bir aloqali ro'yxat; b – daraxtsimon ro'yxat; d – n-aloqali ro'yxat.

Biroq ro'yxatli tuzilmalardan foydalanishda quyidagilarni yodda tutish lozim:

- ko'rsatkichlarni saqlash uchun qo'shimcha xotira zarur;
- yo'nalishli ro'yxatlarda axborotni izlash bosqichma-bosqich amalga oshiriladi va shu boisdan ko'proq vaqt talab qiladi;
- ro'yxatlarni tuzish va ro'yxatlarda saqlanayotgan ma'lumotlar elementlari ustidan operatsiyalarni bajarish dasturchilarning yanada yuqori malakasini hamda ko'p ishlashni talab qiladi, tegishli dasturlar esa ko'proq xatolarga egaligi bois yanada batafsil testlashni talab etadi.

Odatda vektorli taqdim statik ko'pliklarni, jadvallarni (bir o'lchamli va ko'p o'lchamli), masalan satrlar, yozuvlar matritsalarini, shuningdek o'rindoshlik matritsasi, insidentlik matritsasi yoki tahliliy taqdim etilgan graflarni saqlash uchun qo'llaniladi. Ro'yxatli taqdim dinamik (o'zgaruvchan) tuzilmalarni va murakkab aloqali tuzilmalarni saqlash uchun qulay.

Eng mas'uliyatli holatlarda ichki taqdimni tanlashda ma'lumotlar tuzilmasi yoki turli variantlar uchun uning elementlari bilan eng ko'p uchraydigan operatsiyalarni bajarishning *hisoblash murakkabligini belgilash maqsadga muvofiq*. Shuningdek, ularning sig'imiy murakkabligini ham baholash kerak.

**Tashqi xotirada ma'lumotlar taqdimi.** Zamonaviy operatsion tizimlar tashqi xotirada ma'lumotlarni tashkillashtirishning ikki usulini: bosqichli va bevosita daxlli usullarini quvvatlaydi.

Ma'lumotlarga bosqichli daxlda faqat ma'lumotlar elementlarini bosqichli o'qish yoki ularning bosqichli yozuvi mumkin. Bunday variant klaviatura yoki displey tipidagi mantiqiy qurilmalar bilan ishlashda, yozuvlar formati ish jarayonida o'zgaradigan fayllarga yoki matnli fayllarga ishlov berish nazarda tutiladi.

Bevosita daxl faqat qaydlangan uzunlikdagi yozuvlar bilan axborot almashinuvi amalga oshiriluvchi diskli fayllar (C ikkilanma fayllari yoki Pascal tiplashtirilgan fayllari) uchun mumkin. Bunday faylning yozuvi manzilini uning raqami bo'yicha aniqlash mumkin, bu esa kerakli yozuvga bevosita murojaat qilishga imkon beradi.

Ma'lumotlar tuzilmalarini joylashtirish uchun xotira tipini tanlashda quyidagilarni nazarda tutish lozim:

- operativ xotiraga ham o'qish, ham o'zgartirish uchun tezkor ruxsatli kirish zarur ma'lumotlar joylashtiriladi;
- tashqi xotiraga dastur tugallanganidan so'ng saqlanishi shart ma'lumotlar joylashtiriladi.

Mumkinki, ish vaqtida ma'lumotlarni ularga ruxsatli kirishni tezlashtirish uchun operativ xotirada saqlash, ish tugallangandan keyin uzoq muddatli saqlanishi uchun tashqi xotiraga ko'chirib yozish maqsadga muvofiq. Matnli muharrirlarning aksariyati aynan shu usuldan foydalanishadi: matn bilan ishlash paytida matn to'liq yoki qisman operativ xotiraga joylashtiriladi va u joydan zaruratga ko'ra tashqi xotiraga ko'chiriladi. Bunday holat-

larda ma'lumotlarning ikki taqdimi: operativ hamda tashqi xotiradagi taqdimlari ishlab chiqiladi.

Tuzilmalarning to'g'ri tanlanishi ko'p jihatdan ishlab chiqilayotgan dasturiy ta'minot samaradorligini va uning texnologik sifatlarini belgilaydi va shu boisdan mazkur masalaga foydalanilayotgan yondashuvdan qat'i nazar yetarlicha e'tibor berilishi shart.

## 5.5. Ma'lumotlar dekompozitsiyasiga asoslangan dasturiy ta'minlashni loyihalash

Dasturiy ta'minotni loyihalashning ma'lumotlar dekompozitsiyasiga asoslangan Jekson hamda Varnye–Orr metodikalari amalda bir vaqtning o'zida taklif etilganligi eslatib o'tilgan edi. Har ikki metodika murakkab, lekin iyerarxik tashkillashtirilgan ma'lumotlar tuzilmalari bilan ishlovchi «sodda» dasturlarni ishlab chiqish uchun mo'ljallangan. Dasturiy tizimlarni ishlab chiqish zarur bo'lganda har ikki holatda ham avval tizimni dasturlarga bo'laklash, so'ngra metodika ma'lumotlaridan foydalanish nazarda tutiladi.

**Jekson metodikasi.** O'z metodikasini yaratishda M.Jekson boshlang'ich ma'lumotlar va natijalar tuzilmalari dastur tuzilmasini belgilashidan kelib chiqqan.

Metodika boshlang'ich ma'lumotlar va natijalar tuzilmalarining muvofiqliklarini izlashga asoslangan. Biroq uning qo'llanishida qandaydir darajada muvofiqliklar mavjud bo'lmagan vaziyatlar yuzaga kelishi mumkin. Masalan, boshlang'ich fayl yozuvlari tegishli satrlar hisobotda namoyon bo'lishi shart tartibda tartiblanmagan. Bunday vaziyatlar «to'qnashuvlar» deb nomlangan edi. Turlicha hal qilinadigan to'qnashuvlarning bir necha tiplari ajratiladi. Yozuvlarning turlicha bosqichliligida ularni ishlov berishga qadar shunchaki tartiblanadi. To'qnashuvlarni hal etish usullarida yanada batafsil bayon etilgan.

Dastur tuzilmasini metodikaga muvofiq ishlab chiqish quyidagicha yo'sinda bajariladi:

- kirish va chiqish ma'lumotlari tuzilmalarining tasviri barpo qilinadi;
- ushbu ma'lumotlar o'rtasidagi ishlov aloqalari (muvofiqliklar) identifikatsiyasi bajariladi;
- ma'lumotlar tuzilmalari va aniqlangan muvofiqliklar asosida dastur tuzilmasi shakllantiriladi;
- muvofiqliklar aniqlanilmagan elementlar ishlovi bloklari qo'shimcha qilinadi;
- nomuvofiqliklar tahlil qilinadi va ishlovdan o'tkaziladi, ya'ni «to'qnashuvlar» hal etiladi;
- zaruriy operatsiyalar (fayllarni kiritish, chiqarish, ochish/yopish va hokazo) qo'shimcha qilinadi;
- dasturni tuzilmaviy notatsiyada (psevdokodda) yozib qo'yiladi.

**Varnye—Orr metodikasi.** Varnye—Orr metodikasi ham Jekson metodikasidagi kabi qoidaga tayanadi, ammo dastur barpo etilishida chiqish ma'lumotlarining tuzilmalari asosiy hisoblanadi va kirish ma'lumotlarining tuzilmalari chiqish ma'lumotlari tuzilmalariga muvofiq kelmagan taqdirda ularni o'zgartirishga yo'l qo'yiladi. 5.4-misolda ishlovni soddalashtirish uchun predmetlar nomining va baholarining o'rnini almashtirish maqsadga muvofiq.

Biroq amaliyotda va doim ham chiqish ma'lumotlari tuzilmalarini qayta ko'rib chiqish imkoniyati mavjud bo'lavermaydi: masalan, boshqa dasturlarni bajarishda olingan ma'lumotlardan foydalanilgan taqdirda ushbu tuzilmalar oldindan qat'iy yuklatilgan bo'lishi mumkin, shu boisdan ushbu metodika kamroq qo'llaniladi.

Yuqorida bayon etilganlardan kelib chiqadiki, ishlab chiqilayotgan dasturlar ma'lumotlari iyerarxiya yoki iyerarxik jamlanma tarzida taqdim etilishi imkoni bo'lgan holatdagina Jekson va Varnye—Orr metodikalaridan foydalanish mumkin.

## 5.6. Tahlil va loyihalashning tuzilmaviy metodologiyalariga asoslangan CASE texnologiyalar

Hozirgi vaqtida tegishli CASE vositalarda tuzilmaviy tahlil va loyihalashning aksariyat mashhur metodologiyalaridan muvafaqqiyatli foydalanish tajribasi to'plangan. SADT (3,3%), Geyn–Sarson tuzilmaviy tizimiy tahlil (20,7%), Yordan–De Marko tuzilmaviy tahlil va loyihalash (36,5%), Jekson tizim rivoji (7,7%), Varnye–Orr DSSD (Data Struktured System Development) tuzilmaviy sxemalar rivoji (5,8%), Uord–Mellor va Xartli real vaqt tizimlari tahlili hamda loyihalanishi, Martin axborot modellashtirish (22,1%) metodikalari eng ko'p joriy etilgan.

Keltirilgan statistik ma'lumotlardan ayonki, ma'lumotlar oqimlari diagrammalaridan foydalanuvchi tuzilmaviy metodikalar eng ko'p qo'llanilmoqda. Buning ikkita sababi bor:

- ma'lumotlar oqimlari diagrammalari funksional diagrammalarga qiyosan hozirgi paytda ko'p sonli axborot tizimlar spetsifikasini yanada batafsil aks ettiradi: ishlanayotgan axborotning qat'iy tiplashtirilishini talab qilmaydi, ma'lumotlar saqlanishi imkoniyatini nazarda tutadi, tashqi dunyo bilan o'zaro harakatlanishni muayyanlashtiradi, dasturiy ta'minlash kompleks modeli olinishini nazarda tutadi va hokazo;
- ma'lumotlar oqimlari diagrammalari bo'yicha loyiha spetsifikatsiyalari (Jekson yoki Konstantayn tuzilmaviy xaritalari) barpo etilishining usuli ishlab chiqilgan, bu esa avtomatik ravishda shunday spetsifikatsiyalarni yaratish imkonini beradi.

5.2-jadvalda tegishli paketni quvvatlovchi modellar haqidagi ma'lumotlar, 5.3-jadvalda esa tegishli axborot taqdimining notatsiyalari ko'rsatiladi.

Keyingi paytda dasturiy ta'minlash ishlab chiqishning obyektli-mo'ljalli vositalari tobora kengroq tarqalayotganligiga qaramay, tuzilmaviy metodikalar takomillashuvi davom etayotir. Ko'plab dasturiy mahsulotlarni, masalan, asosiy qismi ma'lumotlar bazalari bo'lgan tizimlarga talablarni aniqlashtirish uchun das-

turiy mahsulotlar ishlab chiqishda ulardan muvaffaqiyatli foydalanilmoqda, ma'lumotlar oqimlari diagrammalari juda ko'p qo'llanilmoqda.

*5.2-jadval*

| Model nomi | Firma | Funksiya-lar | Ma'lumot-lar | Hodisalar |
|---|---|---|---|---|
| BPWin | Logic Works | + | – | – |
| CASE Analitik | Eyteks | + | + | + |
| CASE/4/0 | MicroTOOL | + | + | + |
| DatabaseDesigner | Oracle | – | + | – |
| Design/IDEF | Meta Software | + | + | – |
| Designer/2000 | Oracle | + | + | – |
| EasyCASE | Evergreen CASE Tools | + | + | + |
| ERWin | Logic Works | – | + | – |
| I-CASEYourdan | CAYENNE | + | + | + |
| Prokit*WORK | BENCHMDIS | + | + | – |
| S-Designer SIL-VERRun | Sybase/Power-soft CSA | + + | + + | + |
| VIsible Analyst Workbench | Visible Sys-tems | + | + | – |

*5.3-jadval*

| Nomi | DFD notatsiya | Spetsifikatsiyalar | Harakati | Tuzilmaviy xaritalar |
|---|---|---|---|---|
| CASE Analitik | Geyn-Sarson | Tuzilmaviy til | Boshqaruvchi oqimlar | — |
| CASE/4/0 | Iordan (kengaytirilgan) | — | Uord-Mellor (STD bilan) | Jekson |
| Designer/2000 | Geyn-Sarson | — | — | Jekson |
| I-CASE Yourdan | Geyn-Sarson, Iordan | Tuzilmaviy til | Uord-Mellor (STD bilan) | Konstantayn |
| EasyCASE | Iordan | 3GL | STD | Konstantayn |
| Prokit* | Geyn-Sarson | — | — | Konstantayn |
| WORKBENCH | | | | |
| S-Designer | Geyn-Sarson, Iordan | — | — | Konstantayn |
| SILVERRun | Ixtiyoriy | | Boshqaruvchi oqimlar | - |
| | | | | |
| Visible Analyst WORKBENCH | Geyn-Sarson, Iordan | — | — | Konstantayn |

**Nazorat savollari:**

1. Dasturiy ta'minlashning tuzilmaviy va funksional sxemalari deganda nima tushuniladi? Ular qanday qo'llaniladi? Dasturiy ta'minlashning turlicha arxitekturali tuzilmaviy va funksional sxemalari nima bilan farqlanadi?

2. Qadam-baqadam usuli dasturiy tuzilmalarning qanday xossalariga asoslangan? Nima uchun uning qo'llanishi bilan faqat tuzilmaviy algoritmlar olinadi? Sizningcha, ushbu usulning asosiy murakkabligi nimada?

3. Dasturiy ta'minlashning algoritmlari va tuzilmalarini ishlab chiqishda qadam-baqadam detallashtirish usulidan qanday foydalaniladi?

4. Qadam-baqadam detallashtirish usulidan foydalangan holda rim raqamlarida yozilgan sonlar qo'shuvi algoritmini ishlab chiqing: I-1; II-2; III-3; IV-4; V-5; VI-6; IX-9; X-10; L-50; C-100; D-500; M-1000.

5. Konstantayn tuzilmaviy xaritalari nima uchun tuziladi? 4-topshiriq uchun Konstantayn xaritalarini tuzing. Jekson tuzilmaviy xaritalari Konstantayn tuzilmaviy xaritalaridan nima bilan farqlanadi?

6. Jekson va Varnye—Orr metodikalari asosiga nima qo'yilgan? Mazkur metodikalar nima bilan farqlanadi?

7. Ma'lumotlar tuzilmalarini loyihalashda qanday masalalar hal etiladi? Bunda loyihalanayotgan tuzilmalarning qanday tavsiflari hisobga olinadi? 3-topshiriq dasturi uchun ma'lumotlar tuzilmalarining bir necha variantini taklif eting. Ulardan qay biri eng yaxshi va nima uchun?

8. Qanday ishlanmalar uchun tuzilmaviy metodologiyalardan foydalanish maqsadga muvofiq?

# 6. OBYEKTLI YONDASHUVDA TALABLAR TAHLILI VA DASTURIY TA'MINOT SPETSIFIKATSIYALARINI BAHOLASH

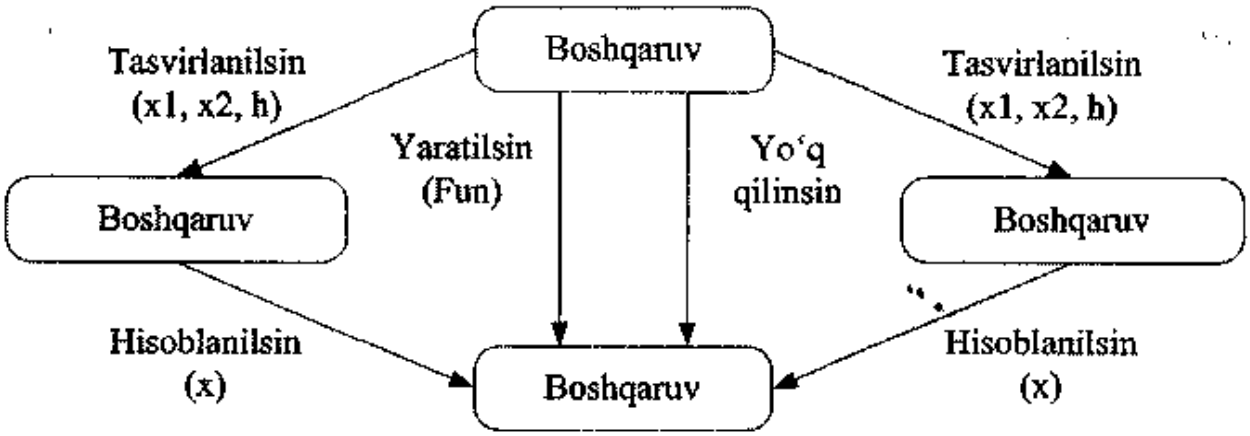
Ishlab chiqiluvchi dasturiy ta'minotning obyektli yondashuvidagi modellari real dunyo predmetlari va hodisalariga asoslangan. Shuningdek, ushbu modellar asosida ishlab chiqiluvchi dasturiy ta'minlashning talab qilinuvchi xatti-harakati, ya'ni uning funksionalligi bayoni turadi, biroq bu xatti-harakat muayyan predmetli soha elementlari (obyektlari)ning holatlari bilan bog'liq bo'ladi.

Shunday qilib, tahlil bosqichida ikki vazifa qo'yiladi:

- ishlab chiqiluvchi dasturiy ta'minotning talab qilinuvchi xatti-harakatini aniqlashtirish;
- qo'yilgan vazifalar nuqtayi nazaridan uning predmetli sohasi konseptual modelini ishlab chiqish.

## 6.1. UML – dasturiy mahsulotlar ishlab chiqishning obyektli yondashuv bilan birgalikdagi standart tili

Dasturiy ta'minotni ishlab chiqishga obyektli yondashuv asosida obyektli dekompozitsiya, ya'ni ishlab chiquvchi dasturiy ta'minotni obyektlar jamlanmasi tarzida taqdim etish turadiki, ushbu obyektlarning o'zaro harakati jarayonida axborotlarni berish orqali talab qilinuvchi funksiyalarni bajarish yuz beradi (6.1-rasm).




**6.1-rasm.** Jadvallar va grafiklar tuzish dasturining obyektli dekompozitsiyasi.

Biroq xuddi tuzilmaviy yondashuvdagi kabi obyektli yondashuvda ham faqat juda oddiy dasturiy ta'minot dekompozitsiyasini darhol bajarish mumkin. Shu boisdan obyektli, mo'ljalli dasturlash davri ibtidosida dasturiy ta'minotni obyektli yondashuv doirasida tahlil qilish va loyihalashning har xil modellar hamda notatsiyalardan foydalanilgan turlicha usullari taklif qilingandi. Mazkur usullar va modellarning afzalliklari hamda kamchiliklari xususida uzoq bahslashish mumkin edi. Ushbu vaziyat «usullar urushi» nomini olgandi.

«Usullar urushi»ga 1995-yilda UML (Unified Modeling Language – modellashtirishning unifikatsiyalangan tili) tili birinchi versiyasining paydo bo'lishi yakun yasadi. Ushbu til hozirgi paytda obyektga – mo'ljallangan yondashuv bilan yaratiluvchi loyihalar bayonining standart vositasi deb amalda tan olingan. Ushbu sohalardagi yetakchi mutaxassislar Gradi Buch, Ivar Yakobson va Jeyms Rombo mazkur tilning yaratuvchilari bo'lib, ular «usullar urushi» vaqtida ushbu mualliflarning yondashuvlarida paydo bo'lgan barcha yaxshi narsalardan o'z tilida foydalanishgan.

UML dan foydalanishda ishlab chiqiluvchi dasturiy ta'minot spetsifikatsiyasi bir necha modellarni: foydalanish, mantiqiy, amalga oshirish, jarayonlar, yoyish modellarini birlashtiradi (8.2-rasm).

**Foydalanish modeli** foydalanuvchi nuqtayi nazaridan dasturiy ta'minlash funksionalligi bayonini o'zida ifoda etadi.

**Mantiqiy model** dasturiy ta'minotning kalit abstraksiyalarini (sinflar, interfeyslar va h.k.), ya'ni talab qilinuvchi funksiyalarni ta'minlovchi vositalarni bayon etadi.

**Amalga oshirish modeli** ishlab chiqish muhitida dasturiy modellarning real tashkillantirilishini belgilaydi.

**Jarayonlar modeli** hisoblashlar tashkillashtirilishini aks ettiradi va «jarayonlar» va «iplar» tushunchalari bilan ish yuritadi. Ushbu model dasturiy ta'minlash mahsuldorligini, miqyoslanuvchanligini va ishonchliligini baholashga imkon beradi.

Nihoyat, **yoyish modeli** muayyan uskunada dasturiy komponentalarni joylashtirishning xususiyatlarini ko'rsatadi.

Shunday qilib, ko'rsatilgan modellarning har biri loyihalanuvchi tizimning muayyan aspektini tavsiflaydi, ularning hammasi esa birgalikda ishlab chiqiluvchi dasturiy ta'minlashning nisbatan to'liq modelini tarkib toptiradi.

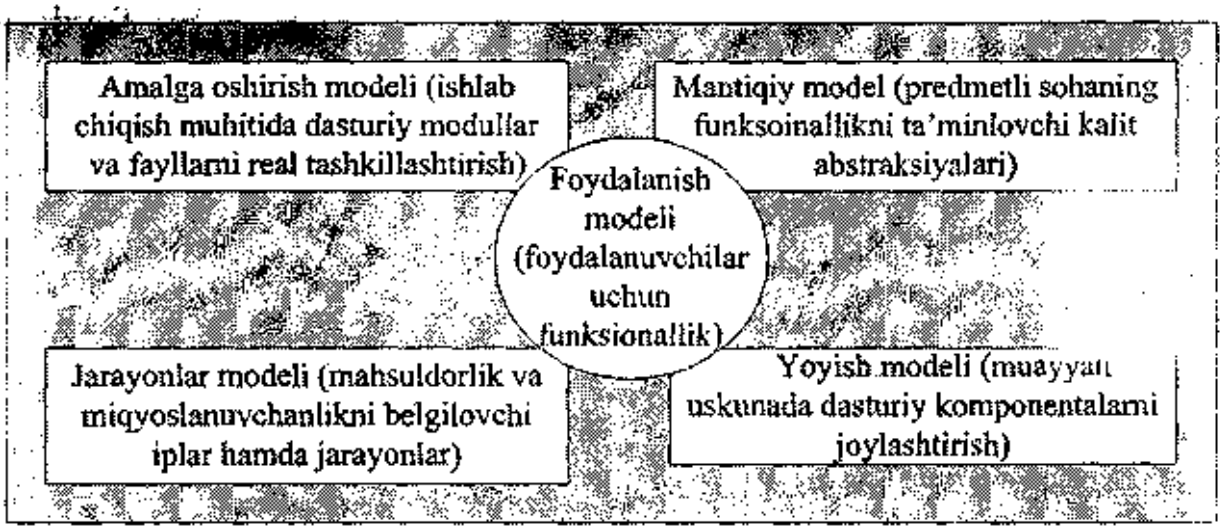
Umuman UML turli modellarga mansub va bir-birini to'ldiruvchi to'qqizta diagrammani taklif etadi:

- foydalanish variantlari diagrammalari;
- sinflar diagrammalari;
- paketlar diagrammalari;
- amallar bosqichliliklari diagrammalari;
- kooperatsiya diagrammalari;
- faoliyatlar diagrammalari;
- obyektlar holatlari diagrammalari;
- komponentalar diagrammalari;
- joylashtirish diagrammalari.

Ko'rsatilgan barcha diagrammalar imkon boricha yagona grafikli notatsiyadan foydalanadi, bu ularni tushunishni yengillashtiradi.

Ko'rsatilgan diagrammalar bilan birga, tuzilmaviy yondashuvdagi kabi, spetsifikatsiya amallar lug'atini, shuningdek turli xil bayonlarni va matnli spetsifikatsiyalarni muqarrar qamrab oladi. Hujjatlarning muayyan majmui ishlab chiquvchi tomonidan belgilanadi.

UML hamda o'sha mualliflar taklif etgan Rational Unified Process metodikasi Rational Software Corporation firmasining Rational Rose paketi bilan quvvatlanadi. UML ning qator diagrammalarini, shuningdek Microsoft Visual Modeler dasturi vositalari hamda boshqa CASE vositalar bilan ham qurish mumkin hozirgi paytda «USA Today» ma'lumotlariga ko'ra, 50 ta yetakchi kompyuter kompaniyalaridan 49 tasi dasturiy ta'minotni obyektli yondashuv bilan birgalikda ishlab chiqishda UML dan foydalanishadi, bu esa bugun UML amalda bunday ishlab chiqishlar bayonining standarti bo'lib qolganligi haqida gapirishga imkon beradi.




**6.2-rasm.** Obyektli yondashuvda (UML) ishlab chiqiluvchi dasturiy ta'minlashning to'liq spetsifikatsiyasi.

## 6.2. «Foydalanish variantlarini» belgilash

Dasturiy ta'minot spetsifikatsiyasining ishlab chiqilishi texnik topshiriqda ko'rsatilgan funksionallikka talablar tahlilidan boshlanadi.

Tahlil jarayonida ishlab chiqaruvchi dasturiy ta'minotning tashqi foydalanuvchilar va muayyan foydalanuvchilar bilan o'zaro harakatlanish jarayonida uning xatti-harakatlari alohida aspektlari sanoqnomasi aniqlanadi. Dasturiy ta'minotning xatti-harakati aspektlari «foydalanish variantlari» yoki «presedentlar» (USE CASES) deb nomlangandi.

**Eslatma.** Foydalanish variantlari ko'pchilik dasturiy ta'minot ishlab chiquvchilar tomonidan 1980–1990-yillarda qo'llanilgan loyihalanuvchi dasturiy tizimlar funksiya bajarishi ssenariylarining norasmiy bayoniga asoslangan.

Foydalanish varianti muayyan harakatlanuvchi shaxs tomonidan ishlab chiqiluvchi tizim qo'llanishining xarakterli protsedurasini o'zida ifoda etdiki, bunday shaxs sifatida nafaqat odamlar, balki boshqa tizimlar yoki qurilmalar ham namoyon bo'lishi mumkin.

Foydalanish variantini bo'lg'usi tizimning muayyan operatsiyalari bilan adashtirmaslik lozim. Har bir foydalanish varianti mustaqil ahamiyatga ega ayrim maqsad bilan bog'liq, masalan

matniy muharrir uchun sarlavhani shakllantirish – bu foydalanish varianti, sarlavhalarni maxsus uslublar bilan bog'lash esa sarlavhalashni avtomatik ravishda tuzish mumkin bo'lishi uchun bajarilishi zarur operatsiyadir.

Muayyan protseduraning bajarilish maqsadiga bog'liq ravishda quyidagi foydalanish variantlari farqlanadi:

- asosiy – ishlab chiqiluvchi dasturiy ta'minlashning talab qiluvchi funksionalligini ta'minlaydi;
- ko'makchi – tizimning zaruriy sozlanishi va unga xizmat ko'rsatilishi (masalan, axborotni arxivlashtirish va h.k) bajarilishini ta'minlaydi;
- qo'shimcha – foydalanuvchi uchun qo'shimcha qulayliklarni ta'minlaydi (odatda, ishlab chiqishda ham, tasarrufda ham qandaydir resurslarning jiddiy sarflanishini talab qilmagan holatda amalga oshiriladi).

Foydalanish variantini qisqacha yoki batafsil bayon etish mumkin. Qisqacha bayon shakli quyidagilarni qamrab oladi: foydalanish varianti nomi, uning maqsadi, harakatlanuvchi shaxslar nomi, foydalanish varianti tipi (asosiy, ikkinchi darajali yoki qo'shimcha) va uning qisqacha bayoni. Kombinatorik – optimallashtiruvchi masalalarni hal etish tizimi topshirig'ini bajarish foydalanish variantini quyidagicha tarzda taqdim etishi mumkin:

| | |
|---|---|
| Variant nomi | Topshiriqni bajarish |
| Maqsad | Masala hal etilishi natijalarini olish |
| Harakatlanuvchi shaxslar | Foydalanuvchi |
| Qisqacha bayon | Masala hal etilishi topshiriq tanlovini, algoritm tanlovini, ma'lumotlar topshirig'ini va hal etish natijalari olinishini nazarda tutadi |
| Variant tipi | Asosiy |

Asosiy foydalanish variantlari odatda ishlab chiqiluvchi dasturiy ta'minot predmeti sohasini aks ettirishga intilgan holda batafsil

bayon ettiriladi. Batafsil shakl yuqorida ko'rsatilgan axborotdan tashqari hodisalar tipik bajarishning va mumkin bo'lgan muqobilliklarning bayonini ham qamrab oladi. Hodisalarning tipik borishi ketma-ket raqamlagan holda foydalanuvchi bilan tizim o'rtasidagi muloqot tarzida taqdim etiladi. Agar foydalanuvchi variantlarni tanlashi mumkin bo'lsa, ular alohida jadvalda bayon etiladi. Shuningdek, hodisalarning tipik borishi buzilishi bilan bog'liq muqobilliklar ham keltiriladi. Quyida topshiriqni bajarish foydalanish variantining batafsil bayoni taqdim etiladi:

**Topshiriqni bajarish foydalanish varianti**
**hodisalarning tipik borishi**

| Ijrochi amallari | Tizim javobi |
|---|---|
| 1. Foydalanuvchi yangi topshiriqni shakllantiradi | 2. Tizim yangi topshiriqni qayd etadi va masalalar tiplari ro'yxatini taklif etadi |
| 3. Foydalanuvchi masala tipini tanlaydi | 4. Tizim masala tipini qayd etadi va ma'lumotlar topshirilishi usullari ro'yxatini taklif etadi |
| 5. Foydalanuvchi ma'lumotlar topshirilishi usulini tanlaydi<br>a) Agar klaviaturadan kiritish tanlanilsa, ma'lumotlarni kiritish bo'limiga qaralsin<br>b) Agar ma'lumotlar bazasidan kiritish tanlanilsa, ma'lumotlarni bazadan tanlash bo'limiga qaralsin | 6. Tizim ma'lumotlarni qayd etadi va hal etish algoritmlari ro'yxatini taklif etadi |
| 7. Foydalanuvchi algoritmni tanlaydi | 8. Tizim algoritmni qayd etadi va hal etishni boshlashni taklif etadi |
| 9. Foydalanuvchi hal etish jarayonini shakllantiradi | 10. Tizim topshiriq belgilanishi to'liqligini tekshiradi va masalani hal etish tagdasturini yuritadi |
| 11. Foydalanuvchi kutadi | 12. Tizim foydalanuvchiga natijalarni namoyish etadi va ularni ma'lumotlar bazasida saqlashni taklif etadi |
| 13. Foydalanuvchi natijalarni tahlil qiladi va ularni bazada saqlash kerak-kerak emasligini tanlaydi | 14. Agar ma'lumotlarni saqlash tiklanilsa, u holda tizim topshiriq ma'lumotlarini bazaga yozishni bajaradi |
| | 15. Tizim kutish holatiga o'tadi |

*Muqobillik*

1. Agar dasturni bajarish vaqti foydalanuvchi nuqtayi nazaridan ko'p bo'lsa, u bajarish jarayonini to'xtatadi.

2. Tizim hisob-kitoblarni to'xtatadi, hal etish algoritmlari ro'yxatini taklif etadi va 7-qadamga qaytadi.

*Qo'shimcha axborot*

1. Topshiriq tipi, ma'lumotlar va algoritm tanlovining ixtiyoriy bosqichliligini ta'minlashi zarur.

2. Har qanday bosqichda variantdan chiqish imkoniyatini ta'minlash zarur.

*Ma'lumotlarni kiritish bo'limi*
*Hodisalarning tipik borishi*

| Ijrochi amallari | Tizim javobi |
|---|---|
| 1. Foydalanuvchi ma'lumotlarni kiritishni tanlaydi | 2. Tizim ma'lumotlar kiritilishini ketma-ket so'raydi |
| 3. Foydalanuvchi ma'lumotlarni kiritadi | 4. Tizim ma'lumotlarni tekshiradi va ma'lumotlarni bazada saqlash yoki saqlamaslikni so'raydi |
| 5. Foydalanuvchi so'rovga javob beradi | 6. Agar ma'lumotlarni saqlash varianti tanlanilsa, tizim ma'lumotlarning bazaga yozilishini bajaradi va ularni joriy topshiriqda qayd etadi |

*Muqobillik*

3. Agar yanglish ma'lumotlar aniqlansa, u holda tizim xato haqida xabar beradi va avvalgi qadamga qaytgan holda uni tuzatishni taklif etadi.

**Foydalanish variantlari diagrammalari.** Foydalanish variantlari diagrammalari tizimning kutiluvchi xatti-harakatini ko'rgazmali tasavvur etishga imkon beradi. Harakatlanuvchi shaxs, foydalanish varianti, aloqa – foydalanish variantlari diagrammalarining asosiy tushunchalaridir.

*Ma'lumotlarni bazadan tanlash bo'limi*
*Hodisalarning tipik borishi*

| Ijrochi amallari | Tizim javobi |
|---|---|
| 1. Foydalanuvchi ma'lumotlarni bazadan tanlashni tanlaydi<br>3. Foydalanuvchi ma'lumotlarni tanlaydi | 2. Tizim bazadagi ma'lumotlar ro'yxatini namoyish etadi<br>4. Tizim ma'lumotlarni o'qiydi va ularni joriy topshiriqda qayd etadi |

*Harakatlanuvchi shaxs* – ishlab chiqiluvchi dasturiy ta'minotga munosabat bo'yicha tashqi mavjudod bo'lib, u bilan qandaydir axborotni olish yoki taqdim etish maqsadida o'zaro harakatlanadi. Yuqorida eslatib o'tilganidek, ishlab chiqiluvchi dasturiy ta'minot bilan o'zaro harakatlanuvchi foydalanuvchilar, boshqa dasturiy ta'minotlar yoki qandaydir texnik vositalar harakatlanuvchi shaxslar bo'la olishlari mumkin.

*Foydalanish varianti* – harakatlanuvchi shaxs uchun uning muayyan topshirig'ini hal etuvchi ayrim ravshan protsedura. Barcha foydalanish variantlari u yoki bu darajada ishlab chiqiluvchi tizim funksionalligiga bog'liq va bajariluvchi ish hajmi bo'yicha kuchli farqlanishi mumkin.

*Aloqa* – harakatlanuvchi shaxslar va tegishli foydalanish variantlarining o'zaro harakatlanishi.

Foydalanish variantlari, shuningdek o'zaro bog'liq bo'lishi ham mumkin. Bundan foydalanish va kengayish aloqalari qayd etiladi.

Foydalanish ishlab chiqiluvchi dasturiy ta'minot xatti-harakatining bir necha foydalanish variantlarida takrorlanadigan ayrim fragmenti mavjudligini e'tiborda tutadi. Mazkur fragment alohida foydalanish varianti sifatida rasmiylashtiriladi va u bilan «foydalanish» tipidagi aloqa ko'rsatiladi.

Kengayish birida ayrim qo'shimcha amallar mavjudligi bilan farqlanadigan ikkita o'xshash foydalanish varianti bor bo'lgan taqdirda qo'llaniladi. Mazkur holatda qo'shimcha amallar alohida foydalanish varianti sifatida belgilanadi va u asosiy variant bilan bog'liq bo'ladi.

## 6.3. Predmetli soha konseptual modelini tuzish

Sinflar diagrammalari dasturiy ta'minot ishlab chiqish obyektli-mo'ljalli usullarning markaziy bo'g'inidir, shu bois barcha mavjud usullar mashhur notatsiyalardan birida sinflar diagrammalaridan foydalanadi. Biroq ushbu usullarda sinflar diagrammalarini asosan loyihalash bosqichida qo'llaniladiki, bu muayyan sinflar tuzilishining xususiyatlarini ko'rsatish uchun amalga oshiriladi. Avval mavjud bo'lgan notatsiyalardan farqli o'laroq, UML sinflar diagrammalarining uch darajasidan ularni detallashtirish pog'onasiga bog'liqlikda foydalanishni taklif etadi:

- konseptual daraja, bunda mazkur holatda kontekstli deb nomlanuvchi sinflar diagrammalari predmetli sohaning asosiy tushunchalari o'rtasidagi aloqani namoyish etadi;
- spetsifikatsiyalar darajasi, bunda sinflar diagrammalari predmetli soha sinflarining interfeyslarini, ya'ni ushbu sinflar obyektlarining aloqalarini aks ettiradi;
- amalga oshirish darajasi, bunda sinflar diagrammalari bevosita muayyan sinflarning maydonlari va operatsiyalarini ko'rsatadi.

Amalda bular uchta turli model bo'lib, ular o'rtasidagi aloqa bir xil emas. Xususan, agar konseptual model predmetli sohaning ayrim tushunchasini sinf sifatida belgilash, bu mazkur tushunchani amalga oshirish uchun alohida sinfdan foydalanilishini anglatmaydi. Biroq barcha uch modelda va ularning statik nisbati qiziqtiradi, bu esa yagona notatsiyadan foydalanishga imkon beradi.

Sanab o'tilgan modellarning har biridan dasturiy ta'minot ishlab chiqishning muayyan bosqichida foydalaniladi:

- konseptual modeldan – tahlil bosqichida;
- spetsifikatsiya darajasi sinflar diagrammalaridan – loyihalash bosqichida;
- amalga oshirish darajasi sinflar diagrammalaridan – amalga oshirish bosqichida.

Konseptual model ta'rifiga muvofiq ravishda predmetli soha tushunchalari, ushbu tushunchalarning atributlari va ular o'rtasidagi munosabatlar bilan operatsiyalarni bajaradi. Ishlab chiqiluvchi dasturiy ta'minot predmetli sohasining tushunchasiga moddiy predmetlar ham, predmetli soha mutaxassislari qo'llaydigan abstraksiyalar ham to'g'ri kelishi mumkin.

Modeldagi asosiy tushunchalarga muvofiq ravishda sinflar qo'yiladi. Bunda sinf deyilganda predmetli sohaning topshirilgan obyektlari guruhiga xos umumiy alomatlar jamlanmasi tushuniladi. Mazkur ta'rifga muvofiq sinflar diagrammasida har bir sinfga umumiy alomatlarni sinf qayd etadigan obyektlar guruhi muvofiq keladi. Xususan, Talaba sinfi oliy o'quv yurtlarida tahsil olayotgan odamlar guruhining umumiy alomatlarini birlashtiradi. Sinf nusxasi yoki obyekt (masalan, I.I. Ivanov) o'z sinfining umumiy alomatlariga muqarrar ravishda ega va sinfda qayd etilmagan xos alomatlariga ega bo'lishi mumkin. Jumladan, I.I. Ivanov talaba bo'lish bilan birga yana sportchi, musiqachi va h.k. bo'lishi mumkin. Qat'iy aytganda, talabani identifikatsiyalovchi nom ham shunday xos alomat bo'la oladi.

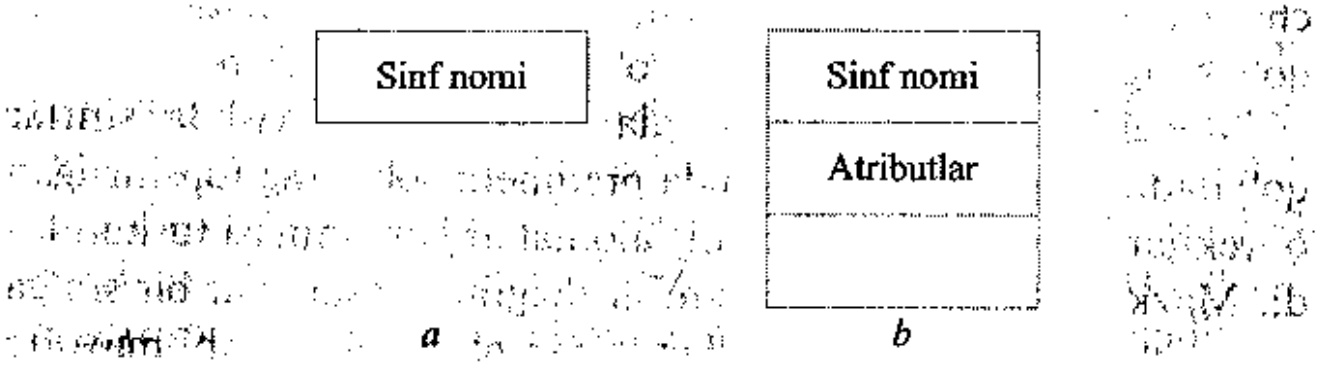
Sinf diagrammalar ichida sinf nomi ko'rsatilgan to'g'riburchak tarzida tasvirlanadi (6.3, a-rasm). Zaruratga ko'ra sinf tavsifnomasini, masalan atributlarni shartli ifodalarning maxsus shu'balaridan foydalangan holda ko'rsatishga ruxsat etiladi (6.3, b-rasm).

Atributlar sifatida obyektlarning hal etilayotgan masala nuqtayi nazaridan ayrim jiddiy tavsiflari, masalan identifikatsiyalovchi birliklari (ism, raqam) taqdim etiladi. Atribut muayyan obyekt uchun doimo belgilangan ifodaga ega bo'ladi. Atributlarni sinflar diagrammasida, odatda atributlar shu'basida ko'rsatiladi.

Sinflar munosabati deyilganda statik, ya'ni sinflararo vaqtga bog'liq bo'lmagan ikki asosiy turi farqlaniladi: assotsiatsiya va umumlashma.

Assotsiatsiya munosabati sinflar nusxalari yoki obyektlar o'rtasida aloqa mavjudligini anglatadi, masalan Talaba sinfi Institut sinfi bilan assotsiatsiyalanadi. Assotsiatsiya nomga ega

bo'lishi ham mumkin, masalan Tahsil oladi. Assotsiatsiya nomi bilan bir qatorda odatda nom o'qilishi yo'nalishini ko'rsatuvchi strelka qo'yiladi («Talaba institutda tahsil oladi», aksincha emas).




**6.3-rasm.** Sinflarning konseptual diagrammasida sinf ifodalanishi: a – tavsifnomalar aniqlashtirilishsiz; b – atributlar aniqlashtirilishi bilan birga.

Sinflar nusxalari o'rtasidagi aloqa tegishli obyektlar bir-biriga munosabat bo'yicha o'ynaydigan ayrim rollarni e'tiborda tutadi. Rol assotsiatsiya yo'nalishi bilan bog'liq. Xususan, talabalarga nisbatan institut ularning tahsil olishini amalga oshiruvchi tashkilot, ya'ni institut rolini o'qish joyi deb nomlash mumkin. Institut uchun talaba institut tahsil faoliyatining obyekti, ya'ni tahsil oluvchi.

Agar rol xos nomiga ega bo'lmasa, uning nomi munosabat bo'yicha mazkur rol belgilanadigan sinf nomi bilan to'g'ri keladi deb hisoblashi mumkin. Ko'rilayotgan misol uchun bu tegishli ravishda Talaba hamda Institut rollari (6.4, a-rasm), biroq rolni ravshan ko'rsatish ham mumkin (6.4, b-rasm).

Rol, shuningdek har bir tomondan bir aloqada qancha obyekt qatnashishi mumkinligini ko'rsatuvchi ko'plik tavsifga ham ega. Ko'plikni quyidagicha ko'rsatishga ruxsat etiladi:

* – 0 dan cheksizlikkacha;

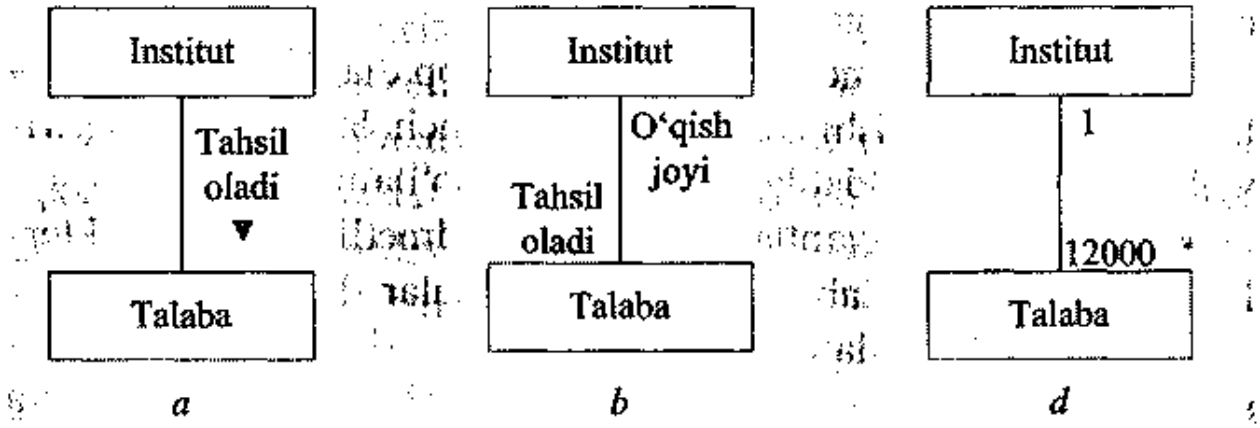
<yaxlit>.. * – topshirilgan miqdordan cheksizlikkacha;

<yaxlit> – obyektlarning aniq belgilangan miqdori;

<yaxlit1>, <yaxlit2> – obyektlar aniq miqdorining bir necha variantlari;

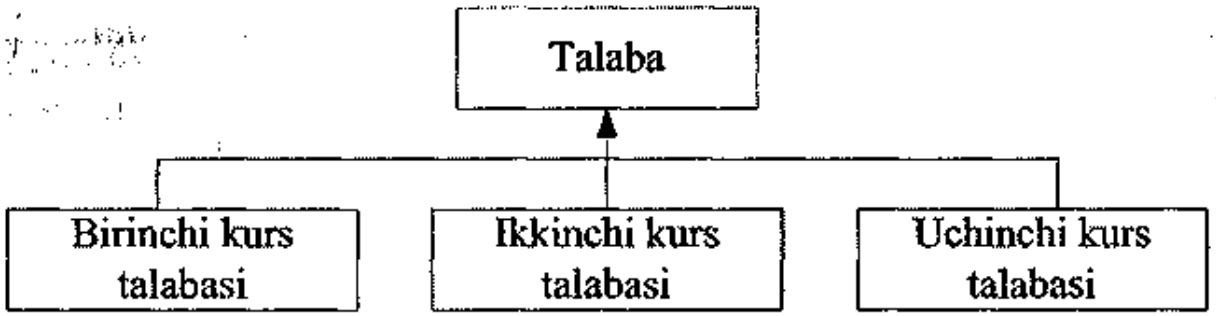
<yaxlit1>..<yaxlit2> – obyektlar diapazonlari.

Nazariy nuqtayi nazardan atribut ham nusxalari ko'rilayotgan sinf bilan qat'iy assotsiatsiyalanuvchi sinfdir. Konseptual modelda tegishli munosabatlarni aks ettirish uchun assotsiatsiyalar ham qo'llanishi mumkin. Masalan, Talaba hamda Nom ikki tushunchasining munosabatini tegishli sinflarning assotsiatsiyalari tarzida ham Talaba sinfiga Nom atributi muvofiqlikda qo'yiladigan variantda ham taqdim etish mumkin.




**6.4-rasm.** Assotsiatsiyalar ifodalanishi: a – assotsiatsiyasi nomi va uning yo'nalishi ko'rsatilishi bilan; b – rollar nomlari ko'rsatilishi bilan; d – ko'plikni ko'rsatish bilan.

Ortiqcha g'ovlardan saqlanishi uchun oddiy qoidaga rioya qilish tavsiya etiladi: agar X ayrim obyekti real dunyoda raqam yoki matn bo'lmasa, demak u tushunchadir. Aks holda esa bu atribut.




**6.5-rasm.** Umumlashmaning ifodalanishi.

Bir sinfning (tagtipning) har qanday obyekti mazkur konspektda supertip deb nomlanuvchi boshqa sinfning ham obyekti

bo'lishi shart holatda sinflararo munosabat umumlashma deyiladi. Xususan, agar ayrim muayyan talaba I.I. Ivanov Talaba supertipining birinchi kurs talabasi, o'sha I.I. Ivanov ko'rsatilgan supertip obyekti ham bo'ladi. Binobarin, supertip obyektlari haqidagi ma'lum barcha narsalar (assotsiatsiyalar, atributlar, operatsiyalar) tagtip obyektlariga ham daxldordir. Sinflar diagrammasida umumlashma supertipga boruvchi uchi uchburchak strelkali chiziq bilan belgilanadi (6.5-rasm).

Amaliyotda predmetli sohaning matnbop diagrammasida sinflar tarzida ifodalanishi shart asosiy tushunchalarini belgilash trivial masala emas. Odatda quyidagi usul qo'llaniladi:

- foydalanish variantlari bayonida predmetli sohani xarakterlovchi otlardan ko'plab nomzod-tushunchalar shakllantiriladi;
- berilgan foydalanish varianti uchun jiddiy bo'lmagan tushunchalar, masalan avvalgi misolda «axborot», «kiritish» va h.k.lar chiqarib tashlanadi.

## 6.4. Xatti-harakat bayoni. Tizimiy hodisalar va operatsiyalar

Konseptual model ishlab chiqiluvchi dasturiy ta'minotning statik xossalarini xarakterlaydi. Uning xatti-harakati xususiyatlarini, ya'ni tizimning mumkin bo'lgan amallarini bayon etish uchun tizim ketma-ketliklari diagrammalaridan, tizimiy hodisalar, tizimiy operatsiyalar, faoliyatlar diagrammalaridan, zaruriyat bo'lganda obyektlar holatlari diagrammalaridan ham foydalanish maqsadga muvofiqdir.

### *Tizim ketma-ketliklari diagrammasi. Tizimiy hodisalar va operatsiyalar*

Tizim ketma-ketliklari diagrammasi — grafik model bo'lib, foydalanish variantining belgilangan ssenariysi uchun harakatlanuvchi shaxslar bo'laklaydigan hodisalarni va ularning tartibini ko'rsatadi. Bunda tizim yagona yaxlitlik sifatida qaraladi.

Tizim ketma-ketliklari diagrammasi tizimi uchun quyidagilar zarur:

• tizimni «qora quti» sifatida tasavvur etish va uning uchun hayot chizig'ini blokka pastdan boruvchi vertikal punktir chiziqni tasvirlash lozim;

• har bir harakatlanuvchi shaxsni identifikatsiyalash va uning uchun hayot chizig'ini tasvirlash kerak (dasturiy ta'minlashdan birgalikda foydalanish variantlarida ko'plab harakatlanuvchi shaxslar bo'ladi);

• foydalanish varianti bayonidan tizimiy hodisalar ko'pligini va ularning ketma-ketligini belgilash darkor;

• tizimiy hodisalarni oxirida strelkani chiziqlar tarzida harakatlanuvchi shaxslar va tizimlar hayot chiziqlari o'rtasida tasvirlash, hamda hodisalarning nomlarini va beriluvchi ifodalar ro'yxatlarini ko'rsatish shart.

Ichki hodisalardan farqli o'laroq, tizim uchun harakatlanuvchi shaxslar tomonidan bo'laklanadigan hodisalar tizimiy hodisalar deyiladi. Tizimli hodisalar tegishli ko'plab operatsiyalar (ular ham tizimiy deyiladi) bajarilishini boshlaydi. Har bir tizimiy operatsiyani tegishli xabar nomi bo'yicha nomlanadi.

Barcha tizimli operatsiyalarning ko'pligini barcha foydalanish variantlarining tizimiy hodisalarini identifikatsiyalagan holda belgilanadi. Ko'rgazmalilik uchun tizimiy operatsiyalarni System abstrakt sinfi (tipi) operatsiyalari tarzida tasvirlanadi. Agar operatsiyalar ko'pligini turli foydalanuvchilar boshlanuvchi tagko'pliklarga ajratish zarur bo'lsa, u holda bir necha System 1, System 2 va h.k. abstrakt sinflardan foydalaniladi.

Har bir tizimli operatsiyalarni bayon etish zarur. Odatda tizimli operatsiya bayoni quyidagilarni qamrab oladi:

• operatsiya nomi va uning parametrlari;
• majburiyat bayoni;
• tip ko'rsatmasi;
• u foydalaniladigan foydalanish variantlari nomlari;
• algoritmlar ishlab chiquvchilari uchun eslatmalar va h.k.lar;
• mumkin bo'lgan istisnolarga ishlov berish bayoni;
• nointerfeys xabarlarni chiqarish bayoni;

• operatsiya bajarilgunga qadar tizim holati haqida taxmin (old shart);

• operatsiya bajarilgandan so'ng tizim holati o'zgarishining bayoni (so'ng shart).

**Nazorat savollari:**

1. Obyektiv dekompozitsiyaning mohiyati nimada?
2. UML tilidan nima uchun foydalaniladi? Nima uchun u modellashtirish tili deb yuritiladi? Obyekt ishlanmalari ta'riflash standarti sifatida nima uchun aynan shu til tanlangan?
3. Obyektiv yondashuvda dasturiy ta'minot tavsifi sifatida qanday diagrammalardan foydalaniladi?
4. «Foydalanish varianti» nima? Foydalanish variantlari diagrammasi qanday tuziladi va u o'zida qanday ma'lumotlarni jamlaydi?
5. Predmet sohasining konseptual modeli nima uchun kerak? Ularning tuzilish metodikasini tushuntirib bering.
6. Konseptual modellar predmet sohalarining asosiy tushunchalari o'rtasidagi qanday munosabatlarni aks ettiradi?
7. Ishlab chiqilayotgan dasturiy ta'minotning harakatini bayon etish uchun UML ning qanday diagrammalari qo'llaniladi?
8. Muntazam voqealar va operatsiyalar deganda nima tushuniladi?
9. Vektor grafikasini qo'llovchi sodda grafik muharrir spetsifikatsiyasini ishlab chiqing. Ushbu holatda qanday diagrammalarni tuzish maqsadga muvofiq?

# 7. OBYEKTLI YONDASHUVDA DASTURIY TA'MINOTNI LOYIHALASHTIRISH

Obyektli yondashuvda mantiqiy loyhalashtirishning asosiy vazifasi – dekompozitsiyalashda olingan obyektlarni amalga oshirish uchun sinflarni ishlab chiqishdan iborat. Bunda har bir sinf maydoni va usulini to'liq bayon etish mo'ljallanadi.

Obyektli yondashuvda jismoniy loyihalashtirish dasturiy komponentlarga sinflarni va boshqa dasturiy resurslarni birlashtiradi, shuningdek, ushbu komponentlarni ma'lum bir hisoblash qurilmalariga joylashtiradi.

## 7.1. Obyektli yondashuvda dasturiy ta'minot tuzilmasini ishlab chiqish

Ko'pchilik sinflarni ma'lum bir tiplarga ajratish mumkin. Ushbu yondashuvga qo'llash mumkin bo'lgan sinflar stereotip deb yuritiladi. Masalan:

- sinflar-mohiyatlar (Predmet sohasi sinflari);
- chegara (interfeys) sinflar;
- boshqaruvchi sinflar;
- xoli sinflar va hokazo (7.1-rasm).

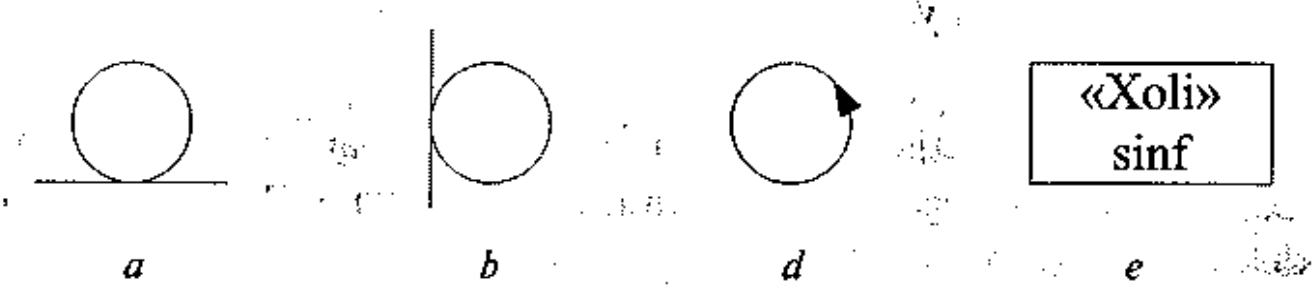
Sinflar-mohiyatlar mavjud dunyo mohiyatini yoki tizimning ichki elementlarini, masalan ma'lumotlar tuzilmasini tasavvur etish uchun foydalaniladi. Odatda, ular atrof-muhitga bog'liq emas va ulardan turli ilovalarda foydalaniladi. Sinflarning mohiyatlarini aniqlash uchun foydalanish variantlarini konseptual model va faoliyat diagrammasini ta'rifi o'rganiladi. Shu tarzda olingan sinflarning nomzodlar ro'yxati predmet sohasiga, til ifodasiga tegishli bo'lmagan so'zlarni olib tashlagan holda saralanadi. Shundan so'ng obyektlarining ham holat, ham harakatga ega bo'lgan sinflarning nomzodlarning tanlab olinadi.

Chegara sinflar harakatdagi shaxslar va tizimning ikki elementlari o'rtasidagi o'zaro harakatni ta'minlaydi. Bunday turga foydalanuvchi interfeyslarni amalga oshiruvchi sinflar ham,

shuningdek, interfeysni apparat vositalari yoki dasturiy tizimlar bilan ta'minlovchi sinflar ham kiradi. Chegara sinflarni topish uchun juftlik – «harakatlanuvchi shaxs – foydalanish varianti» o'rganiladi.

Boshqaruvchi sinflar bir yoki bir necha foydalanish variantlariga kiritilgan ketma-ket harakatlanishni modellashtirish uchun xizmat qiladi.

Agar sinflarning nomzodlar va boshqa resurslar soni ko'p bo'lsa, ularni guruhlarga, ya'ni paketga birlashtirish lozim. Obyektli yondashuvda sinflar va boshqa dasturiy resurslar ta'rifining majmui paketlar deb yuritiladi.




**7.1-rasm.** Sinflar stereotiplarining shartli belgilanishi: a – sinfning mohiyati; b – chegara sinf; d – boshqaruvchi sinf; e – stereotipning aniq ko'rsatilishi.

Paketlarga birlashtirish faqat soni ko'p bo'lgan sinflar, katta loyihalarni yaratishga qulay bo'lishi uchun foydalaniladi. Bunda bitta paketga bir xil ahamiyatga ega sinflar va boshqa resurslar to'planadi.

*Paketlar diagrammasi* loyihalashtirilayotgan dasturiy tizim qanday qismlardan tuzilgani, ushbu qismlar bir-biri bilan qanday bog'langanligini ko'rsatadi.

Agar bitta paketdagi o'zgarish boshqasining o'zgarishiga olib kelsa, paketlar o'rtasidagi aloqa qayd etiladi. Bu, paketga birlashgan sinflar va boshqa resurslarning tashqi aloqalarini belgilab beradi. Ayni paytda sinflar bog'liqligining turli xillari bo'lishi mumkin. Masalan:

- bir sinf obyektlari boshqa sinf obyektlariga xabar jo'natadi;

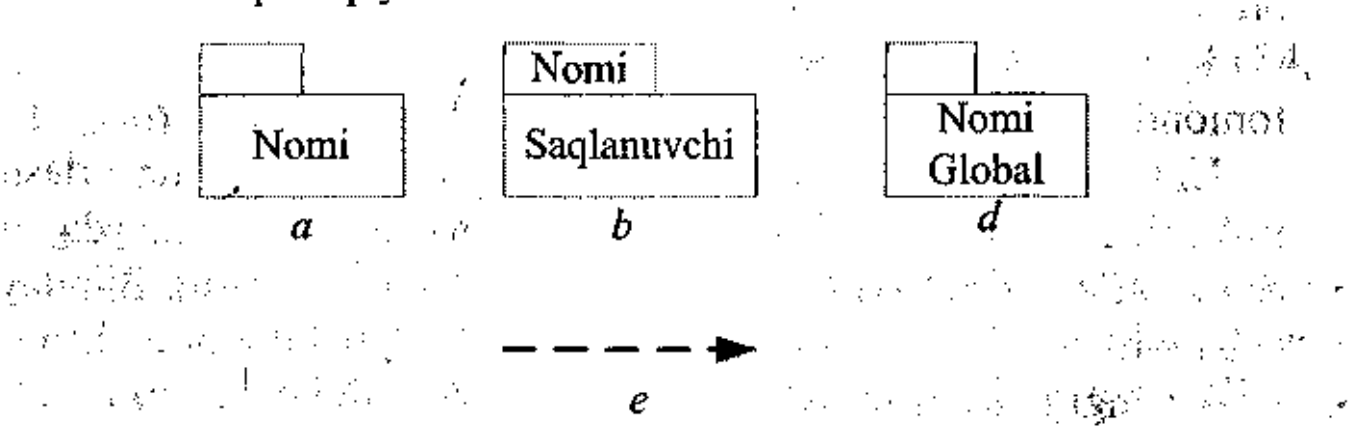
• bitta sinf obyekti boshqa obyektlar komponentlariga murojaat qiladi;

• bitta sinf obyektlari usullar parametrlari boshqa sinf obyektlaridan foydalaniladi va h.k.;

• har bir paket paketning barcha resurslari bayonini saqlovchi interfeysni o'z ichiga oladigan variant eng yaxshi texnologik tavsifligi bilan ajralib turadi. Bu holatda paket resurslaridan foydalanishning o'zgarishi boshqa paketlarga ta'sir ko'rsatmaydi. Faqat interfeysdagi o'zgarishgina ushbu paket resurslaridan foydalanuvchi paketlarni o'zgartirishni talab qilishi mumkin va paketlarning o'zaro harakati faqat shu interfeys orqali amalga oshiriladi.

Dasturiy tizimning barcha paketlari bilan bog'langan paketlar global deb yuritiladi. Bunday paketlar interfeyslari alohida aniq loihalashtirilishi lozim. Chunki, ulardagi o'zgarish ishlab chiqilayotgan barcha paketlarni tekshirishni talab qiladi.

7.2-rasmda paketlar diagrammasida foydalanish mumkin bo'lgan UML notatsiyalari belgilari keltirilgan. Paketlar diagrammasida ko'rsatilgan belgilardan tashqari to'ldirilgan belgilarni ham qayd etish mumkin. Chunki, qoidaga ko'ra, bir necha paketlarning yagona interfeysi mavjudligi taxmin qilinadi (7.3-rasm). Bunday holatda, paketdan paketga o'xshash super turga nisbatan aloqasi qayd etiladi.




**7.2-rasm.** Paketlar diagrammasi qo'llaniladigan shartli belgilar: a – paket; b – mavjud ma'lumotlar belgilangan paket; d – global paket; e – sinflar bog'liqligi (yoy chaqiruvlar yo'nalishini ko'rsatadi).

## 7.2. Obyektlar o‘rtasidagi munosabatlarni aniqlash

Ishlab chiqilayotgan dasturiy ta’minotning asosiy paketlari aniqlangandan so‘ng, har bir paketga kiruvchi sinflarni loyihalashtirishga o‘tiladi. Aniq bir paketga kiritilishi lozim bo‘lgan sinflar nomzodlar loyihalashtirish bosqichini sinflar bosqichida ko‘rsatadi va ko‘rsatilgan sinflar obyektlari o‘rtasidagi obyektlarni aniqlaydi.

**Loyihalashtirish bosqichida ketma-ketlik diagrammasi.** Loyihalashtirish bosqichining ketma-ketlik (uzviylik) bosqichi vaqt bo‘yicha tartibga solingan obyektlarning o‘zaro harakatini aks ettiradi. Tahlil bosqichining ketma-ketlik bosqichi diagrammasidan farqli ravishda unda ichki obyektlar, shuningdek, xabarlar ketma-ketligi ko‘rsatiladi.

Obyektlar to‘g‘riburchak ko‘rinishida tasvirlanadi. Uning ichida obyektni identifikatsiyalovchi axborotlar nomi, obyekt nomi va sinf nomi yoki faqat sinf nomi ko‘rsatiladi (7.3-rasm).

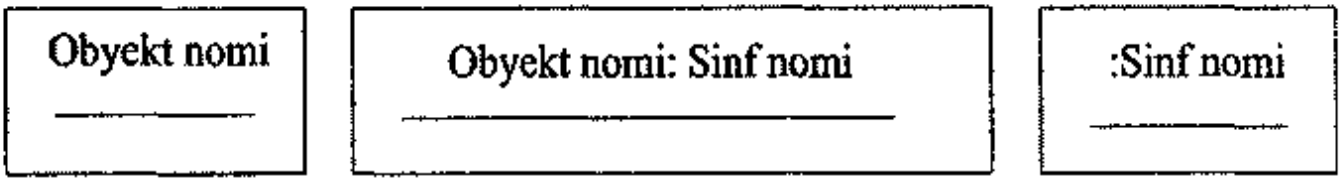
Har bir xabar ikkita obyekt hayot chizig‘ini birlashtiruvchi yoy ko‘rinishida taqdim etiladi. Bu chiziqlar diagrammaga xabar generatsiyasi tartibida (yuqoridan pastga va chapdan o‘ngga) joylashadi. Xabarga nom beriladi, ammo argumentlarni va boshqaruvchi axborotlarni, masalan shakllanish shartlari yoki iteratsiya markerini ko‘rsatishi mumkin (*). Sinxron xabar uning qaytganligini hech bir belgi berilmaganidan ehtimol qilish mumkin.

Agar obyekt xabar orqali yaratilsa, xabar strelkasi unga chap tomondan kiritilgan holda chiziladi.

Uzviylik diagrammasi ham parallel jarayonlarni tasvirlash imkonini beradi. Chaqiruvchi obyekt ishini to‘sib qo‘ymaydigan asinxron xabarlar yarim yoy bilan ko‘rsatiladi (7.4-rasm). Bunday xabarlar:

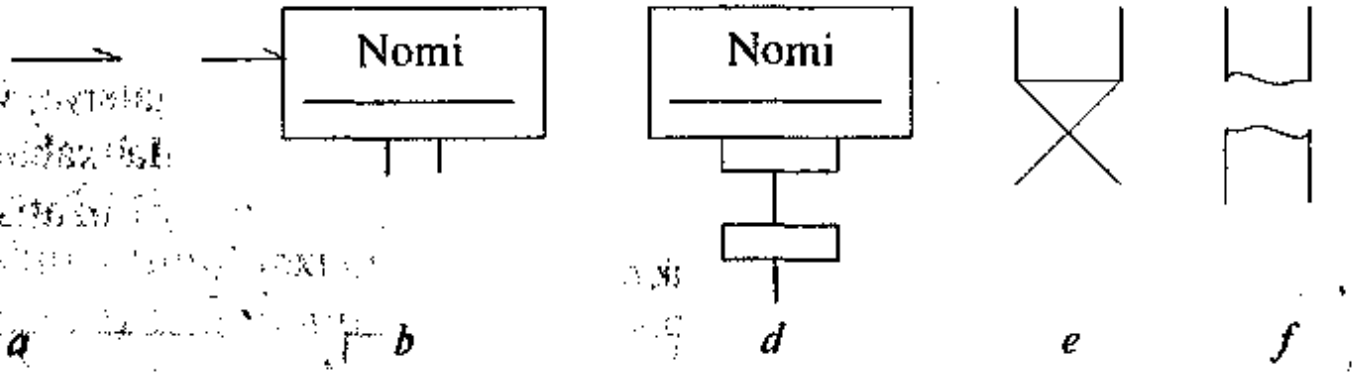
- yangi jarayon shoxchasini hosil qilishi;
- yangi obyektni hosil qilishi (7.4, b-rasm);
- endilikda jarayon shoxchasi bilan aloqani o‘rnatishi mumkin.

Hayot chizig'ida bunday holatda qo'shimcha ravishda faollashtirish ko'rsatiladi. U hayot chizig'i ustiga qo'yilgan to'g'ri to'rtburchak bilan belgilanadi (7.4, d-rasm).




*a* *b* *d*

**7.3-rasm.** UML da obyektlarning shartli belgilari: a – obyekt; b – sinflar aniqlangan obyektlar; d – ko'rsatilgan sinfning nomlanmagan obyekti.




**7.4-rasm.** Boshqaruvni asinxron uzatishning shartli belgilari: a – asinxron xabar; b – obyektni yaratish (asinxron bo'lishi shart emas); d – obyektni faollashtirish; e – obyektni yo'q qilish; f – bo'linish (boshqa ishlanmalarni bajarish).

Obyektni «yo'q» qilish katta «X» belgisi bilan ko'rsatiladi (7.4, d-rasm).

## 7.3. Sinflar munosabatlarini aniqlash

Sinflarni loyihalashtirish jarayoni ular o'rtasidagi munosabatlarni aniqlashdan boshlanadi. Loyihalashtirish bosqichida assotsiatsiya va to'ldirishdan tashqari sinflar o'rtasidagi munosabatlarning ikki turi farqlanadi. Bular – agregatsiya va kompozitsiya.

Afsuski, hozirgacha obyektga mo'ljallangan loyihalashtirishning yagona mustahkam atamasi mavjud emas. Agar «butun-qism» munosabati ma'lum bir holatda muhim bo'lsa, agregatsiya

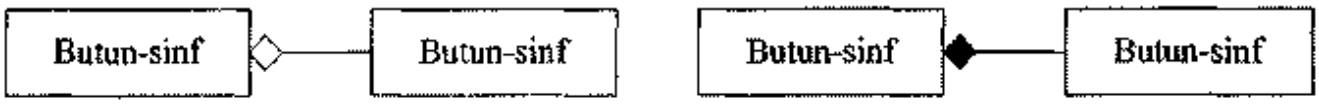
assotsiatsiya bilan birga ko'rsatiladi. Masalan, g'ildirak bizni avtomobilning bir qismi sifatidagina qiziqtirsa, u holda tegishli sinflar o'rtasida agregatsiya munosabatini ko'rsatish maqsadga muvofiq. Agar g'ildirak xuddi avtomobil kabi tovar sifatida qiziqtirsa, «butun-qism» degan aloqa muhim emas.

*Kompozitsiya* – agregatsiyaning yanada kuchli xilma-xilligi. U obyekt-qism faqat yagona butunga tegishli bo'lishi mumkin deb hisoblaydi.

*Obyekt*-qism bu holda hosil qilinadi va faqat o'z butuni bilan birgagina yo'q qilinishi mumkin.

Sinflar o'rtasidagi aniqlangan munosabatlar sinflar diagrammasida qayd etiladi. Buning uchun maxsus shartli belgilar qo'llaniladi (7.5-rasm).

Chunki assotsiatsiya va uning kichik turlari (agregatsiya va kompozitsiya) munosabatlari sinflar obyektlari o'rtasida xabarlar almashinuvi mavjudligini bildiradi. Bunda xabarlarni uzatish yo'nalishini aniqlash maqsadga muvofiqdir.




*a* *b*

**7.5-rasm.** Assotsiatsiyaning maxsus ko'rinishdagi shartli belgilari: a – kompozitsiya; b – agregatsiya.

Navigatsiya (assotsiatsiya yo'nalishi) assotsiatsiya chizig'i oxirida strelka bilan ko'rsatiladi. Agar strelka har ikki tomondan ko'rsatilgan bo'lsa, bu ikki yo'nalishdagi assotsiatsiyani anglatadi.

Loyihalashtirish bosqichining sinflar diagrammasidagi maxsus belgilardan abstrakt sinflarni ko'rsatish uchun foydalaniladi. Sinflar diagrammasida ularning nomi kursiv bilan ajratiladi yoki sinf nomi oldidan «abstract» stereotipi ko'rsatiladi.

UML ham parametrlangan sinflar yoki shablonlarni belgilash uchun maxsus notatsiyani kiritadi (7.6, a-rasm). Shunday usul bilan olingan konkret tipdagi elementlarga ega sinflar bog'lovchi

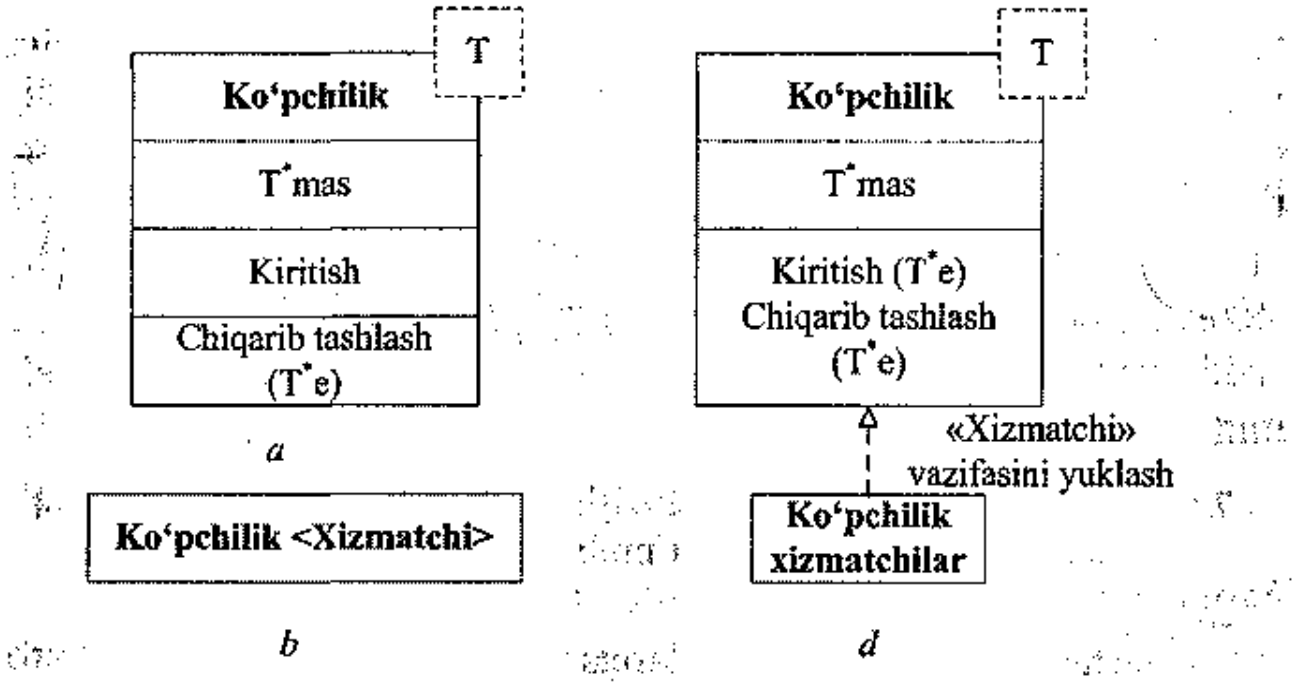
deb yuritiladi. Bog'lovchilarni ikki xil usul bilan belgilash mumkin. Ya'ni, parametr turini aniq ko'rsatish (7.6, b-rasm) va aniqlashtirish shartli belgisini qo'llash orqali (7.6, d-rasm).

Sinflar diagrammasi shuningdek, faqat yuqorida ko'rib chiqilgan tushunchalardan (assotsiatsiya, to'ldirish, atributlar, operatsiyalar) foydalangan holda ko'rsatib bo'lmaydigan cheklashlarni aks ettirishi mumkin.

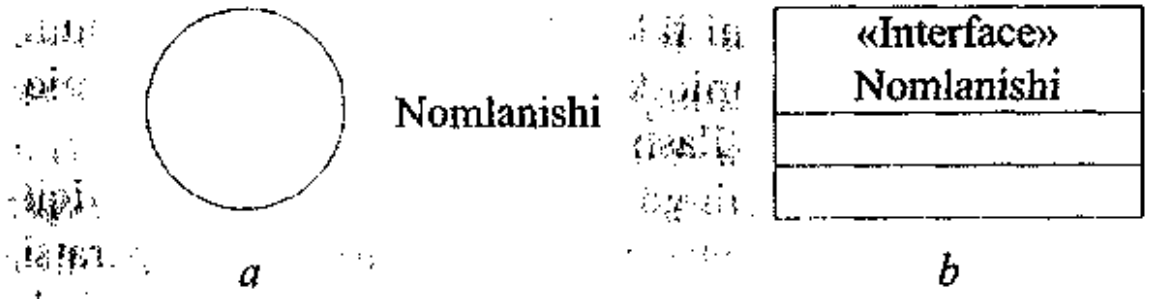
Masalan, talabaning o'rtacha balli tegishli formula bo'yicha belgilanishini ko'rsatishi mumkin.

Sinflar diagrammasida bunday axborotlarni tabiiy tildagi yozuv ko'rinishida yoki figurali qavs ichiga joylashtirilgan matematik formula shaklida taqdim etish mumkin.

Sinflarni loyihalashtirish jarayonida interfeyslarni loyihalashtirish alohida rol o'ynaydi. UML da faqat operatsiyalar e'lonini o'z ichiga olgan sinflargina interfeyslar deb yuritiladi. Interfeyslarning ayrim ta'riflarini loyihalashtirilayotgan dasturiy ta'minotning texnologik sifatini yaxshilaydi. Interfeyslar tarmoq dasturiy ta'minotni ishlab chiqishda keng qo'llaniladi.




**7.6-rasm.** UML da interfeysning shartli belgilari:
a – parametrlangan sinf; b – bog'lash paytida parametr turining aniq ko'rsatilishi; d – aniqlashtirishni qo'llash.

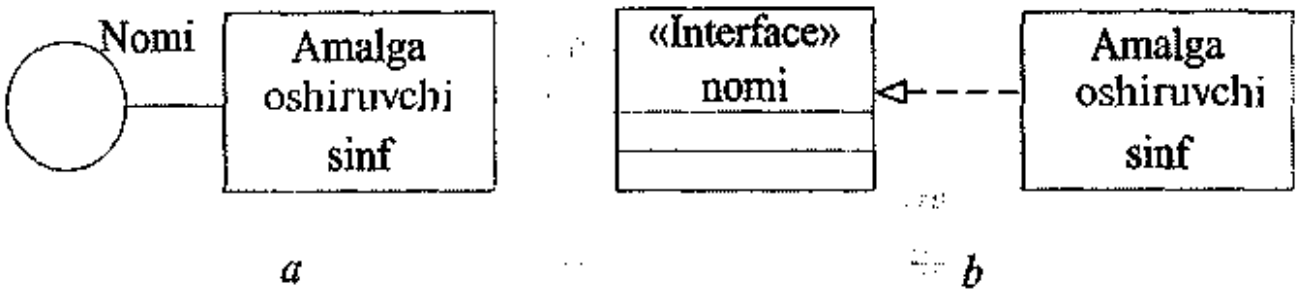



**7.7-rasm.** UML da interfeysning shartli belgilari: a – maxsus belgilar; b – stereotipni ko'rsatish orqali.

Obyektga mo'ljallangan dasturlash nuqtayi nazaridan interfeys abstrakt sinfning alohida turini ifodalaydi. U ko'rsatilgan operatsiyalar va maydonlar e'lonini amalga oshiruvchi usullarni saqlamasligi bilan ajralib turadi. Boshqacha aytganda abstrakt sinflar ayrim usullarni amalga oshirish imkonini beradi, interfeyslar esa barcha usullarni aniqlashtirishni keyinga qoldirishni talab etadi.

Sinflar diagrammasida interfeysni ikki xil usulda ko'rsatish mumkin: qisqartirilgan (7.7, a-rasm), yoki amalga oshirish munosabatini qo'llash orqali (7.7, b-rasm).

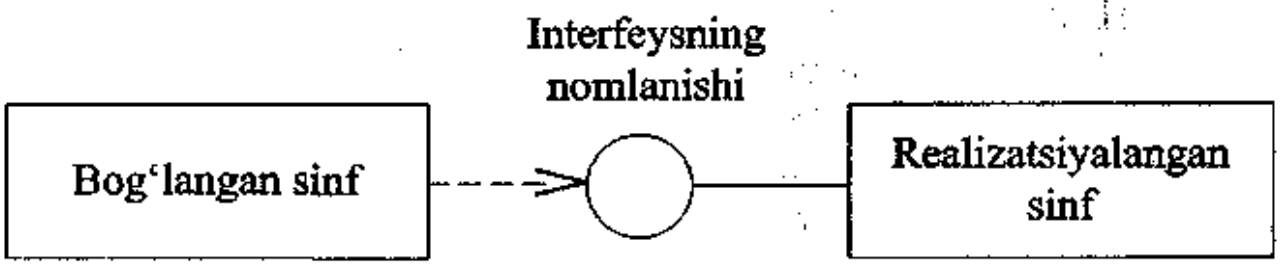
Interfeysni amalga oshirishni ham ikkita uslub bilan ko'rsatish mumkin: qisqartirilgan (7.8, a-rasm), yoki amalga oshirish munosabatini qo'llash orqali (7.8, b-rasm).




**7.8-rasm.** Interfeysni amalga oshirishning shartli belgilari: a – siqiq shakl; b – amalga oshirish munosabati ko'rsatilgan shakl.

Interfeys bilan assotsiatsiyalangan boshqa sinflar uchun o'zaro bog'liqlik munosabatini ko'rsatgan holda assotsiatsiyani aniqlashtirish lozim. Bu munosabatlar mazkur holatda sinf ko'rsatilgan interfeysdan foydalanayotganligini anglatadi (7.9-rasm).

Ayni paytda paketdagi sinflar munosabatini aniqlashtirgan holda turli paketlarga kiritilgan sinflar o'rtasidagi munosabatni ham o'ylab ko'rish lozim.




**7.9-rasm.** Interfeys bilan bog'liq sinf belgilari.

## 7.4. Sinflarni loyihalashtirish

Umuman olganda sinflarni loyihalahtirish tuzilmalar va ular obyektlari xatti-harakatini yakuniy belgilashni nazarda tutadi. Obyektlar tuzilmasi atributlar majmui va sinflar operatsiyalari orqali belgilanadi. Har bir atribut bu sinf obyektida saqlanuvchi ma'lumotlar maydoni.

*Sinf obyektlarining* harakati bajariladigan majburiyatlar bilan aniqlanadi. Majburiyatlar sinf operatsiyalari vositasida bajariladi.

Shu tariqa, sinfni loyihalashtirishda nomlanishi va atributlarning to'liq ro'yxatidan tashqari uning mas'uliyati hamda operatsiyalarini aniqlashtirish zarur. Holbuki, atributlar kabi operatsiyalarni ham loyihalashtirish jarayonida qo'shimcha tasniflash maqsadga muvofiq.

Sinflar diagrammasini detallashtirish darajasiga ko'ra, atributlar belgilari sifatida nomlanishdan tashqari quyidagilarni kiritish mumkin: tip, ravshanlik bayoni va ahamiyati. Buning uchun quyidagi format qo'llaniladi:

*<ravshanlik alomati> <nomi>: <tip>=<jimlik ahamiyati>,*

bu yerda ravshanlik (ko'rinish) belgisi quyidagi uchta sifatdan birini qabul qilishi mumkin: <<+>> – umumiy; <<#>> – himoyalangan, <<->> – yashirin.

Yuqorida eslatib o'tilganidek, sinflar tomonidan amalga oshiriladigan asosiy amallar operatsiyalar deb yuritiladi. Usullardan

farqli ravishda operatsiyalar sinflar tomonidan har doim ham bevosita amalga oshirilavermaydi. Masalan, sonlarni kiritish operatsiyasi <<kiritish darchasi>> amalga oshirilishi mumkin.

UML da sinf diagrammasidagi operatsiyalarning to'liq bayoni quyidagicha ko'rinish olishi:

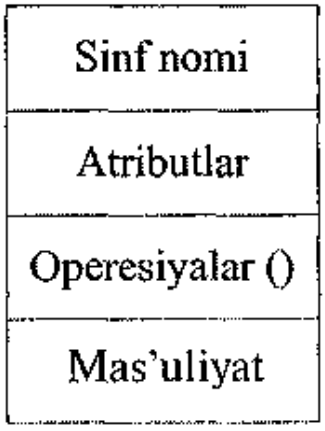
*<ravshanlik alomati> <nomi> (<parametrlar ro'yxati>):*
*<qaytariladigan belgi turi>.*

**Sinf mas'uliyati** deb sinf obyektlari asosiy funksiyalarining qisqacha norasmiy ro'yxatiga aytiladi. Sinf atributlari va operatsiyalari hali belgilanmagan paytda, loyihalashtirishning boshlang'ich bosqichlarida sinf mas'uliyati belgilanadi. Bu axborot sinf diagrammasida, sinfning shartli tasviridagi maxsus seksiyalarda aks etadi (7.10-rasm).

Sinf operatsiyalarining boshlang'ich ro'yxati faoliyat diagrammasini tahlil etgan holda o'zaro harakat diagrammasini va sinf obyektlari ishtirokida turli ssenariylar uchun tuzilgan harakatlar ketma-ketligi diagrammasini loyihalashtirilayotgani shakllantiradi.

Tegishli sinflar obyektlarining umumiy harakatini belgilovchi asosiy operatsiyalarning faqat nominigina ko'rsatadi. Aniqlanish tartibiga ko'ra, yangi operatsiyalar qo'yiladi, mavjud operatsiyalar to'g'risidagi axborot esa detallashtiriladi.

Ko'pgina atributlar predmet sohasi, texnik topshiriqlar talablari va voqealar oqimi bayonini tahlil qilishda aniqlanadi.




**7.10-rasm.** UML dagi sinflarning to'liq shartli belgilari.

Bundan tashqari, yuqorida ko'rsatilganidek, assotsiatsiya va uning kenja turlari – agregatsiya va kompozitsiyalar munosabati sinf obyektlari o'rtasida xabarlar almashuvi mavjudligini anglatadi. Xabarlarni uzatishni tashkil etish uchun chaqirilayotgan obyekt to'g'risida axborotga ega bo'lish kerak.

Bunday holatda agar loyihalashtirilayotgan sinf obyektlari murakkab amallarni bajarishi lozim bo'lsa, ular uchun holat diagrammasini ishlab chiqish maqsadga muvofiqdir.

**Obyekt holati diagrammasi.** *Holat* diagrammasiga nisbatan obyekt holati deganda, obyektning hayot siklidagi vaziyat tushuniladi. Bu davrda u ayrim obyektlar sharoitini yaxshilaydi, ma'lum bir faoliyatni amalga oshiradi yoki biror holatni kutadi. Sharoit buzilishi, yoki faoliyat tugashi yoki yangi voqealar boshlanishi bilan bog'liq holatlar o'zgarishi «o'tishi» deb ataladi.

Holatlar diagrammasi obyektlar holati, ehtimolli o'tishlar, shuningdek, har bir o'tishni chaqiruvchi voqea xabarlarni ko'rsatadi.

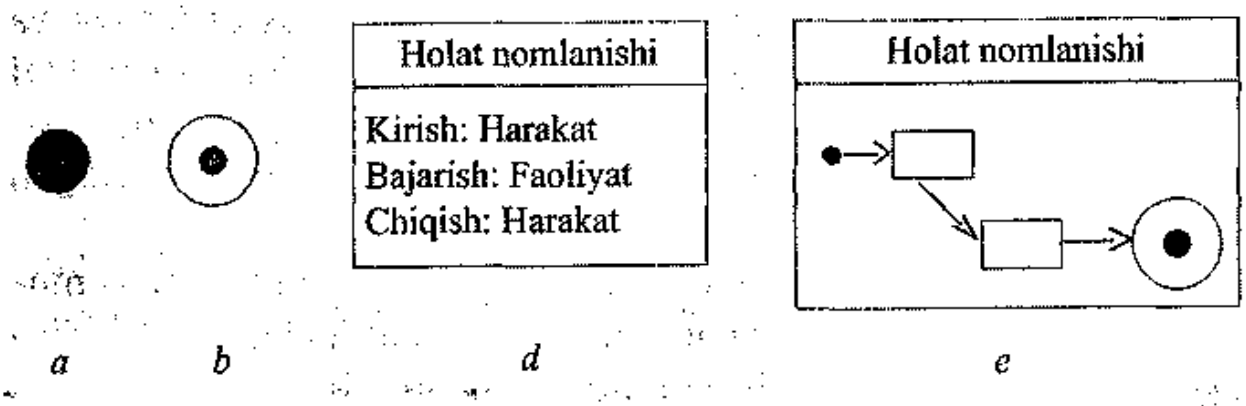
Holatlarning shartli belgilari 7.11-rasmda ko'rsatilgan «Kirish» so'zidan so'ng ko'rsatilgan harakat holatga kirishdan oldin, «chiqish» so'zidan keyin ko'rsatilgan harakat esa undan chiqishda amalga oshiriladi.

O'tish strelkali chiziq bilan belgilanadi va uch qismdan iborat belgi bilan belgilanishi mumkin. Ularning har biri quyidagicha ko'rinishda bo'lishi mumkin:

*<Voqea>[<Shart>]/<Harakat>.*

Agar *voqea* ko'rsatilmagan bo'lsa, bu shuni anglatadiki, o'tish mazkur holat bilan bog'liq faoliyat tugallangach amalga oshiriladi. Agar u ko'rsatilgan bo'lsa, o'tish voqea boshlanishida bajariladi.

Sharoit mantiqiy ifoda shaklida yoziladi. Agar natija ifodasi – «haqiqat» bo'lsa, o'tish amalga oshadi. Obyekt bir paytning o'zida ikkita turli holatga o'tishi mumkin emas. Shuning uchun o'tish sharti har qanday holat uchun o'zaro rad etishili lozim.




**7.11-rasm.** Obyekt holati diagrammasining asosiy shartli belgilari: a – boshlang'ich holat; b – oxirgi holat; d – oraliq holat; e – eng yaxshi holat.

Faoliyatidan farqli ravishda o'tish uchun ko'rsatilgan harakat oxirgisi bilan bog'lanadi va lahzada hamda uzilmagan holda amalga oshadi.

Zarur hollarda, bir necha holatni bittasiga birlashtiradigan super holatni belgilash mumkin. Bunda ichki jarayonni uzishga qaysi vaqtda ruxsat berilishini aniqlab olish lozim. Jarayonni uzish yana uni initsiallashtirish vaqtida mumkin ekanligini ko'rsatish uchun «jarayon» super holatini kiritamiz.

Amalga oshirishda Algoritm faollashigacha bo'lgan jarayonni uzish mumkinligini hisobga olish lozim.

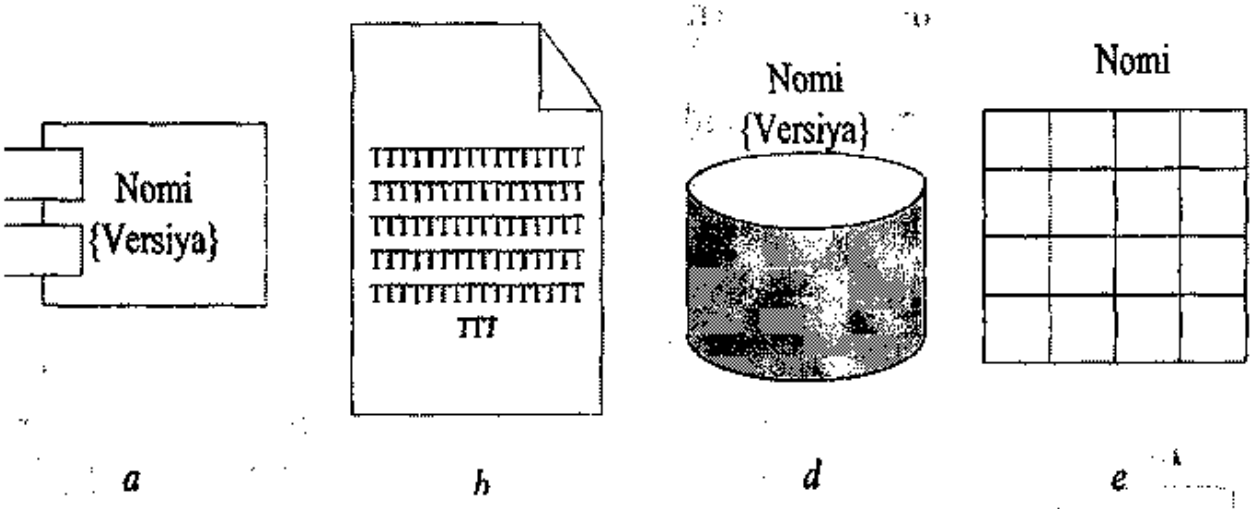
Sinflar obyektlari tuzilmasi va harakatini aniqlash natijalarini sinflar diagrammasida aks ettiramiz.

## 7.5. Dasturiy komponentlarni komponentlash

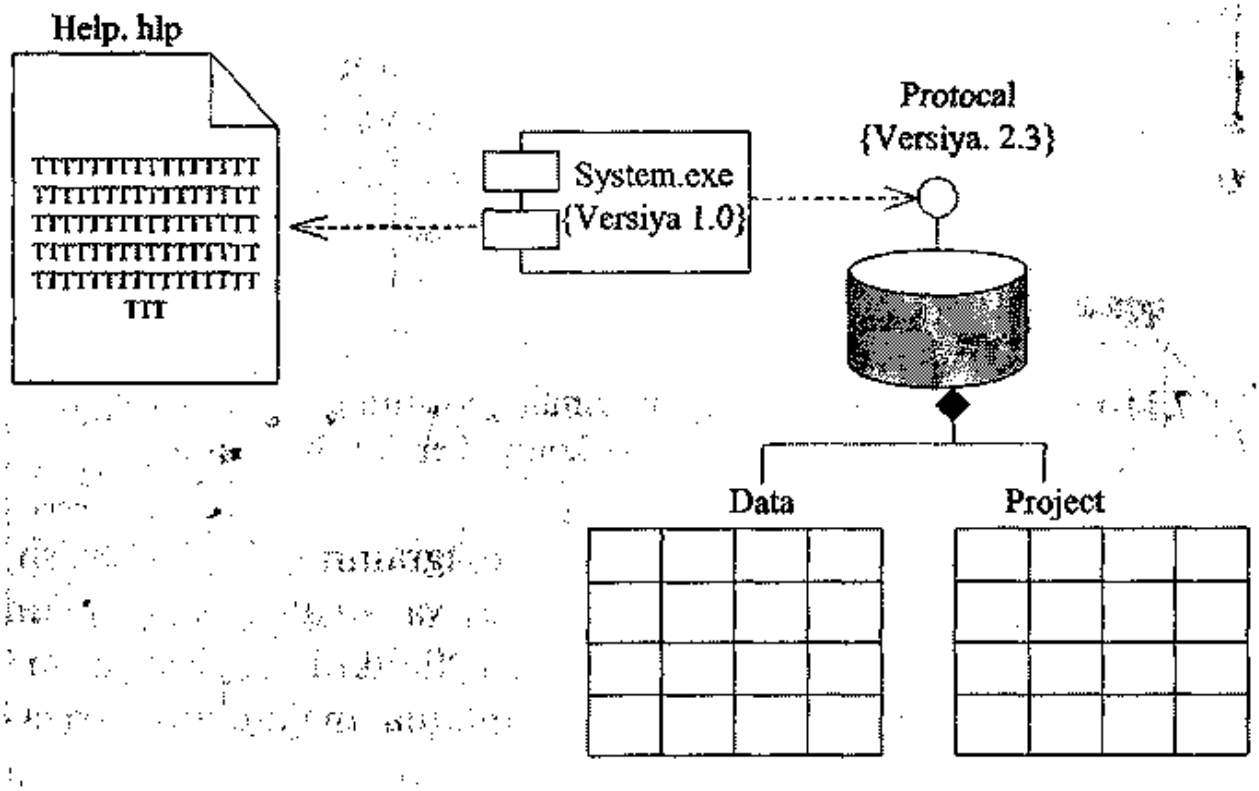
*Komponentlar diagrammasi* ishlab chiqilayotgan dasturiy ta'minotning jismoniy tuzilmasini loyihalashtirishda qo'llaniladi. Bu diagramma dasturiy ta'minot jismoniy darajada qanday ko'rinishini, ya'ni u qanday qismlardan tashkil topishi va bu qismlar o'zaro qanday bog'langanligini ko'rsatadi.

Komponentlar diagrammasi komponent va *bog'liqlik tushunchalarini* qisman o'zgartirib qo'llaydi. Bunda komponentlar deganda, interfeyslar to'plamiga mos keluvchi dasturiy

ta'minotning jismoniy almashtiriladigan qismi tushuniladi. Mohiyatan, turli tiplarning alohida fayllaridir: bajariladigan (exe), matnli, grafik, ma'lumotlar bazasi jadvallari va ishlab chiqilayotgan dasturiy ta'minotni tashkil etuvchi omillardir.



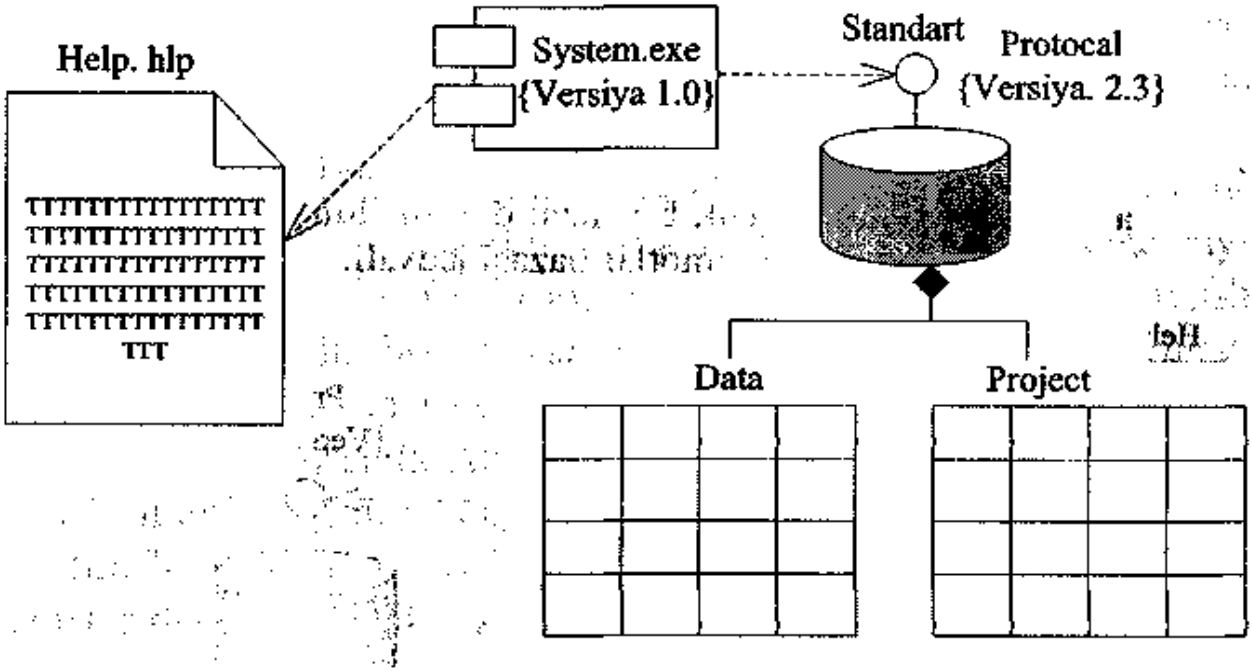

**7.12-rasm.** UML da komponentlarning shartli belgilari:
a – dasturiy komponent; b – fayl; d – ma'lumotlar bazasi;
e – ma'lumotlar bazasi jadvali.




**7.13-rasm.** Kombinatlarining optimallashgan vazifalarini hal etish tizimi komponentlarining diagrammasi.

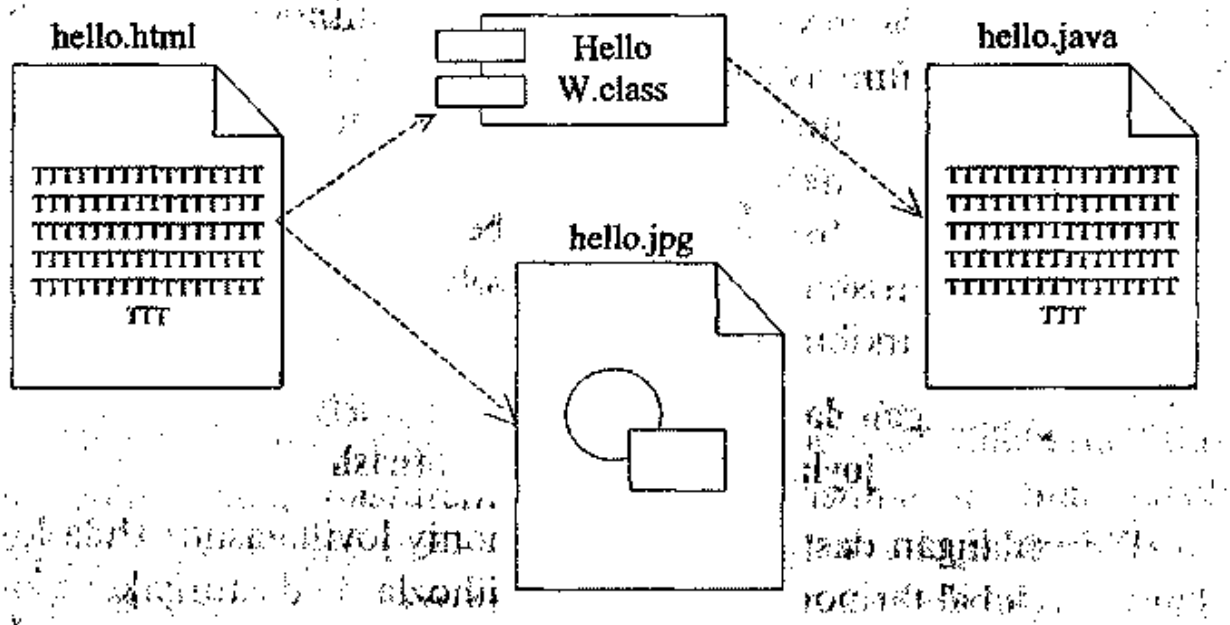
Komponentlar o'rtasidagi bog'liqlik, agar ulardan biri ayrim resurslarga (modul, obyekt, sinf va hokazo) ega bo'lsa, boshqasi undan foydalansa qayd etiladi. Komponentlash sifati soni va komponentlar o'rtasidagi aloqalar turiga ko'ra baholanadi. Komponentlar diagrammasi bog'liqlik punktlar bilan belgilanadi.

7.13-rasmda misol sifatida kombinatlarning optimallashgan tizim komponentlarning diagrammasi keltiriladi. Komponentli yondashuv holatida komponentlar diagrammasida interfeyslarni ko'rsatish maqsadga muvofiq. Ular orqali komponentlar o'zaro bog'langan bo'ladi (7.14-rasm).



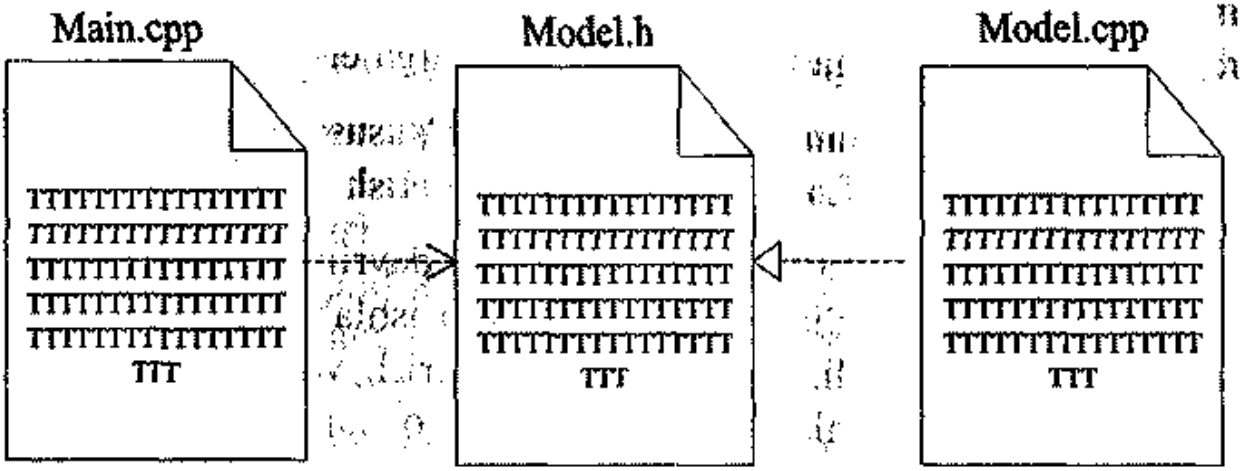

**7.14-rasm.** Interfeys ko'rsatilgan kombinatarlining optimallashgan vazifalarni hal etish tizimining komponentlar diagrammasi.

Bundan tashqari komponentlar diagrammasda to'ldirish, kompozitsiya assotsiatsiyasi, agregatlash va amalga oshirishdan foydalanib komponentlar o'rtasidagi bog'liqlikni aniqlashga yo'l qo'yiladi. Jumladan, 7.13 va 7.14-rasmlarda ma'lumotlar bazasi ikkita jadval (kompozitsiyalar munosabati)ni o'z ichiga olishi ko'rsatilgan.




**7.15-rasm.** «Hello World» matni va fotosuratini chiqaruvchi internet-ilova komponentlash diagrammasi.

UML notatsiyasidan foydalangan holda deyarli har qanday holat uchun komponentlar diagrammasini tuzish mumkin (masalan, Internet-ilova uchun). 7.15-rasmda Internet ilovaning mijoz qismiga oid komponentlar diagrammasi namunasi keltirilgan. U ish jarayonida ayrim rasmlarni namoyish etadigan Java dan foydalanib yozilgan. 7.16-rasmda shu narsa ko'rsatilganki, asosiy Main. cpp fayli Model.h bosh fayliga bog'liq.




**7.16-rasm.** C++ bajariladigan faylni komponentlash diagrammasining namunasi.

«Mijoz-server» arxitekturali dasturiy ta'minot uchun komponentlar diagrammasidan dasturiy ta'minot tomonidan ishlab chiqiladigan arxitekturani belgilovchi tuzilmaviy sxema sifatida foydalanish mumkin. Chunki u tizim qismlarini boshqarish bo'yicha aloqani ko'rsatish imkonini beradi. Biroq, bunday sxemali loyihalashtirishda ishlab chiqilayotgan tizim komponentlarining batafsil tarkibini aniqlashtirish lozim.

## 7.6. Taqsimlangan dasturiy tizim uchun dasturiy komponentlarni joylashtirishni loyihalashtirish

Taqsimlangan dasturiy tizimni jismoniy loyihalashtirishda lokal yoki global tarmoqlardagi ma'lum jihozlarda dasturiy komponentlarni joylashtirishning nisbatan oqilona variantini aniqlashtirish lozim. Buning uchun UML maxsus modeli – joylashtirish diagrammasidan foydalaniladi.

**Joylashtirish diagrammasi** tizimining dasturiy va apparat komponetlari o'rtasidagi o'zaro aloqani aks ettiradi. Tizim apparat tizimining har bir qismiga, masalan, kompyuter yoki datchikka, joylashtirish diagrammasida **tugun** to'g'ri keladi. Tugunlar ulanishi tizimda tegishli kommunikatsiya kanallari mavjudligini anglatadi. Tugunlar ishlab chiqilayotgan dasturiy tizimi dasturiy komponentlarining mazkur jihozda joylashtirishni ko'rsatadi.

Joylashtirish diagrammasi nuqtayi nazaridan lokal va global tarmoqlar – bular ham ayrim spetsifikalarga ega tugunlardir.

## 7.7. Ishlanma spiral modelining xususiyatlari. Loyihani qayta tashkil etish

Dasturiy ta'minot ishlanmasi hayotiy davrining spiral modeli shuni nazarda tutadiki, dasturiy tizimni ishlab chiqish jarayoni iteratsiyali bajariladi. Bunda loyihalashtirish vaqtida navbatdagi faraz topilishi mumkin. Demak, yaxshi o'ylangan boshlang'ich iteratsiyada dastur to'planib qolgan tuzatishlar hisobiga o'zining dastlabki tuzilmasini yo'qotadi. Har bir tuzatish o'z paytida amalga oshirilgan va amalga oshirish jarayonida yuzaga kelgan kichik

nomuvofiqlikni bartaraf etish uchun kiritilgan. Ayni paytda kodlarning bir qismi foydalanib bo'lmaydigan, ayrimlari samarasiz bo'lib qolishi mumkin. Bular hammasi shunga olib keladiki, dasturni hal etish juda murakkab bo'lib qoladi. Bunday vaziyatda dasturni qayta tashkil etish, ya'ni funksiyasini o'zgartirishsiz loyihalashtirish zarur. O'z vaqtida amalga oshirilgan qayta tashkil etish dastur tuzilmasini yanada aniq va tushunarli bo'lishiga olib keladi.

Qayta tashkil etishda o'zgarishlar odatda unchalik katta bo'lmasada, dasturchilar buni yoqtirishmaydi. Shu sabab ko'pincha «ishlayaptimi – tegma» qoidasiga amal qilishadi. Bundan tashqari muddatlar odatda cheklangan bo'ladi va qayta tuzatishga vaqt sarflash maqsadga muvofiq emas. Amaliyot shuni ko'rsatadiki, tuzatishlar to'planib qolganda qayta tuzishdan voz kechish dasturning murakkablashuviga, uning texnologiyasi pasayishiga olib keladi.

Agar dastur funksiyasini kengaytirishda asosiy konsepsiya buzilsa, yoki kod tushunish uchun qiyin bo'lib qolsa qayta tuzishni amalga oshirishi lozim. Bunday holatda loyihalashtirishni to'xtatish va kerakli darajada qayta tuzish lozim.

Qayta loyihalashtirishni uning funksiyalarini kengaytirish bilan birga qo'shib yubormaslik kerak. Dastur qayta tuzilgandan so'ng, biror tartib buzilmaganligiga ishonch hosil qilish uchun testdan o'tkazish va so'ngra uning imkonyatlarini kengaytirish kerak.

**Nazoat savollari:**

1. Obyektli yondashuvda dasturiy ta'minot tuzilmasi qanday bayon etiladi? «Paket» Paketlar diagrammasidan nima uchun foydalaniladi?
2. Sinflarning qanday stereotiplari kiritilgan va nima uchun?
3. 6-bob, 9-vazipredmeti bajarishda o'zingiz bayon etgan grafik muharrirning paketlari kiritilgan va nima uchun? Qanday paketlar o'zaro bog'langan?

4. Siz istagan har qanday paketlar obyektlari uchun xarakterlar ketma-ketligi diagrammasini tuzing. Obyektlar qanday xabarlar bilan o'zaro almashadi? Dasturchi ushbu diagrammani tahlil eta turib qanaqa axborotga ega bo'ladi?

5. Obyektlarning o'zaro harakatini aniqlashtirishda qanday diagrammadan foydalaniladi? Oldingi topshiriq obyektlari uchun ushbu diagrammani tuzing.

6. Sinflarning asosiy komponentlarini sanab o'ting. Bu komponentlar qanday ta'riflanadi?

7. Qanday hollarda obyektlar holati diagrammasidan foydalaniladi? Birorta boshqaruv obyekti uchun holat diagrammasini tuzing.

8. Obyektlar o'zaro ta'sirini tadqiq etish natijalariga ko'ra sinflarning aniqlangan diagrammasini tuzing. Ushbu sinflarni amalga oshirish uchun yana qanday ma'lumotlar kerak bo'ladi?

9. Komponentlar diagrammasi nima? U o'zida qanday ma'lumotlarni aks ettiradi? Qanday hollarda komponentlar diagrammasini tuzish maqsadga muvofiq?

# 8. FOYDALANUVCHI INTERFEYSINI ISHLAB CHIQISH

Hisoblash texnikasi taraqqiyotining ilk bosqichlarida foydalanuvchi interfeysiga insonning operatsion tizimi bilan muloqot vositasi sifatida qaragan. U asosan, vazipredmeti bajarishga tushirish, u bilan ma'lum ma'lumotlarini bog'lash va hisoblash qurilmasiga xizmat ko'rsatishning ayrim turlarini bajarishga imkon bergan.

Vaqt o'tib, apparat vositalari takomillashib borgach, maxsus foydalanuvchi interfeysidan foydalanuvchi interaktiv dasturiy ta'minotni yaratish imkoniyati paydo bo'ladi. Hozirda asosiy muammo professional bo'lmagan foydalanuvchi foydalanishiga mo'ljallangan murakkab dasturiy mahsulotga ega interaktiv interfeyslarni ishlab chiqish feysini tuzishning asosiy konsepsiyasi ta'riflangan va ularni yaratishning bir necha metodikasi taklif qilingan.

## 8.1. Foydalanuvchi interfeys turlari va ularni ishlab chiqish bosqichlari

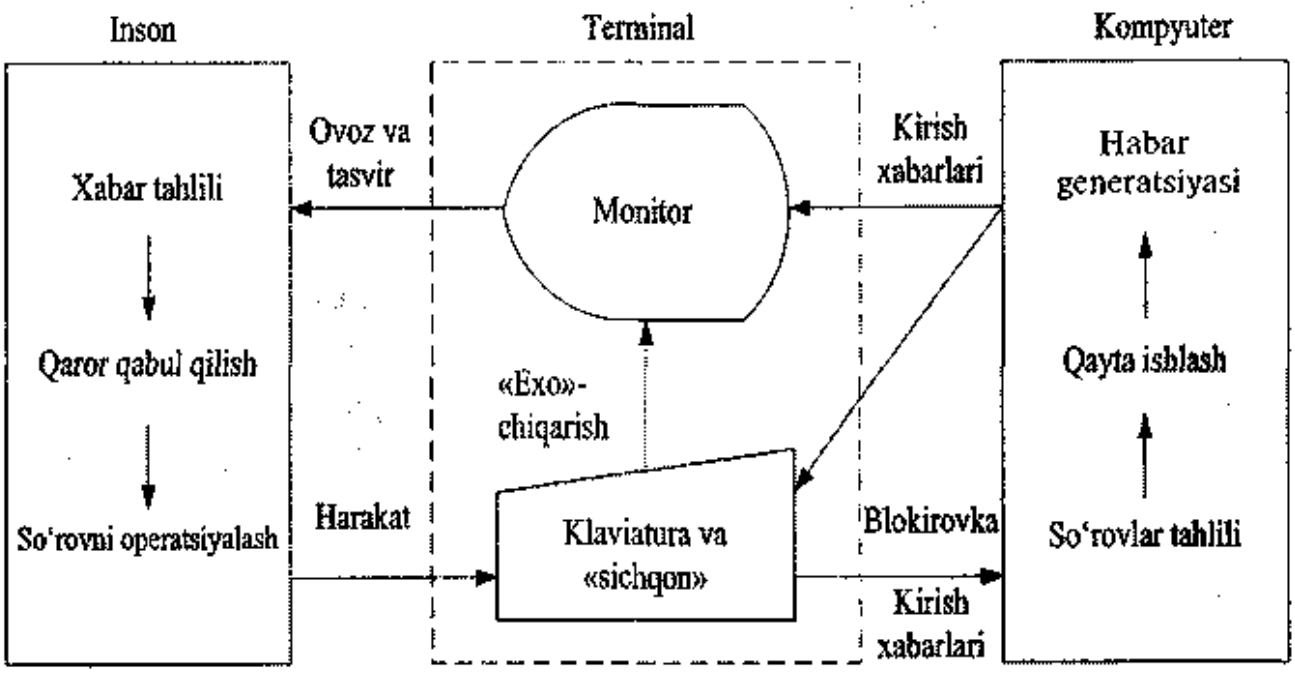
*Foydalanuvchi interfeysi* foydalanuvchi kompyuter bilan o'zaro aloqasini ta'minlovchi dasturiy va apparat vositalari to'plamidan iborat. Bunday o'zaro aloqa asosini dialoglar tashkil etadi. Mazkur holatda dialog deganda inson va kompyuter o'rtasida reglamentlashgan axborot almashinuvi tushuniladi. Bu belgilangan vaqt mobaynida bajariladi va ma'lum bir vazipredmeti hal etishga yo'lnatiriladi. Har bir dialog kiritish-chiqarishning alohida jarayonlaridan iborat bo'lib, foydalanuvchi va kompyuter aloqasini ta'minlaydi.

Axborot almashuvi xabarlar va boshqaruvchi singnallarini uzatish orqali oshiriladi. *Xabar* bu dialog almashinuvida ishtirok etuvchi axborotning bir qismi.

U quyidagi guruhlarga bo'linadi:

- kirish xabarlari. Ular inson tomonidan klaviatura, manipulyatorlar, masalan «sichqon» kabi kiritish vositalari orqali tashkil etiladi;

• chiqish xabarlari. Mazkur ma'lumotlar kompyuter orqali matn, ovozli signallar va (yoki) tasvir shaklida uyushtiriladi hamda foydalanuvchiga monitor ekrani yoki boshqa qurilma vositasida chiqariladi (8.1-rasm).




**8.1-rasm.** Kompyuter va foydalanuvchining o'zaro tasvirni tashkil etish.

Foydalanuvchi asosan quyidagi xabarlar turini uyushtiradi: axborotni so'rash, yordam so'rash, operatsiya yoki funksiya so'rovi, axborotni kiritish yoki o'zgartirish, kadr maydonini tanlash va hokazo. Bunga javoban u quyidagilarga ega bo'ladi: ko'makchi ma'lumot, axborot xabari, harakatni talab etuvchi xatolar to'g'risida xabar, kadr o'lchamini o'zgartirish va hokazo.

Quyida kiritish-chiqarish operatsiyalarini bajarishni ta'minlovchi asosiy qurilmalar sanab o'tilgan.

*Xabarlarni chiqarish uchun:*

• Monoxrom va rangli monitorlar – tezkor matn va grafik axborotni chiqarish;

• printerlar – matnli va grafik axborotning «qat'iy nusxasi»ni olish;

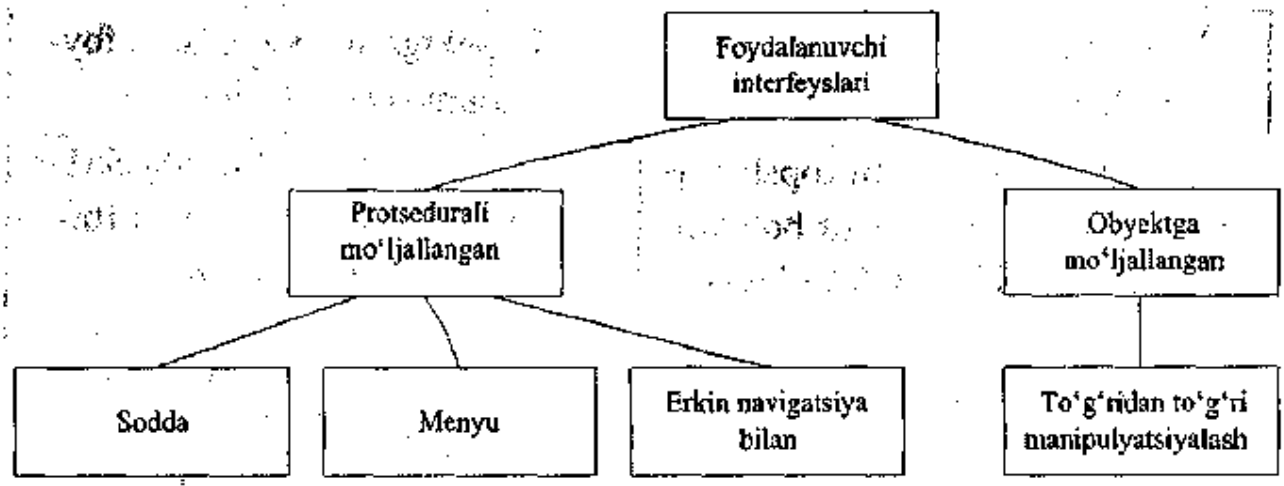
• jadval tuzuvchilar – grafik axborotning qat'iy nusxasini olish;

- nutq sintezatori – nutq chiqishi;
- planshetlar – grafik kiritish;
- skanerlar – grafik kiritish;
- manipulyatorlar, yorug'lik perosi, sensor ekran – axborotni pozitsiyalash va ekranda tanlash va hokazo.

**Interfeyslar turlari.** Dasturlashga nisbatan protsedurlash va obyektiv yondashuv analogiyasi bo'yicha interfeyslarni ishlab chiqishga protsedurali mo'ljallangan va obyektga mo'ljallangan yondashuv farqlanadi (8.2-rasm). Protsedurali mo'ljanllangan interfeyslar «protsedura» va «operatsiya» tushunchalariga asoslangan an'anaviy modeldan foydalanadi.

Ushbu model doirasida dasturiy ta'minot foydalanuvchiga bir qator harakatlarni amalga oshirishga imkon beradi.

Obyektga mo'ljallangan interfeyslar foydalanuvchi bilan o'zaro harakatlanishda predmet sohasining obyektlarini manipulyatsiyalashga mo'ljallangan boshqacharoq modeldan foydalaniladi.




**8.2-rasm.** Interfeyslar turlari.

Ushbu model doirasida foydalanuvchiga har bir obyekt bilan bevosita o'zaro harakat yuritish va bir necha obyekt ishtirok etadigan operatsiyalar bajarilishini initsiallash imkonini beradi. Foydalanuvchi vazifasi ayrim obyektlarni maqsadli ravishda o'zgartirish bilan ifodalanadi. Bu obyekt ichki tuzilishga, ma'lum bir mazmun va tashqi ramziy yoki grafik tasavvurga ega. Bunda

obyekt keng ma'noda tushuniladi. Masalan, aniq bir tizim yoki jarayon modeli, ma'lumotlar bazasi, matn va h.k. Foydalanuvchiga obyektlarni yaratish, ularning parametrlari va boshqa obyektlar bilan aloqasini o'zgartirish, ushbu obyektlarning o'zaro harakatini initsiallash imkonini beradi. Mazkur turdagi interfeyslar elementlari Windows foydalanuvchi interfeysga kiritilgan. Masalan, foydalanuvchi faylni «olishi» va uni boshqa papkaga joylashtirishi mumkin. Shu tarzda u faylni ko'chirish operatsiyasini amalga oshirsa qayta nom beradi.

Bunday holatda protsedurali mo'ljallangan interfeyslar qo'llanilishi tegishli dasturiy ta'minotni ishlab chiqishga tuzilmaviy yondashuvdan foydalanishni anglatadi. Bundan tashqari tuzilmaviy yondashuv bazasida zamonaviy protsedurali mo'ljallangan foydalanuvchi interfeysi juda murakkab va ko'p mehnat talab qiladigan vazifa sanaladi.

*8.1-jadval*

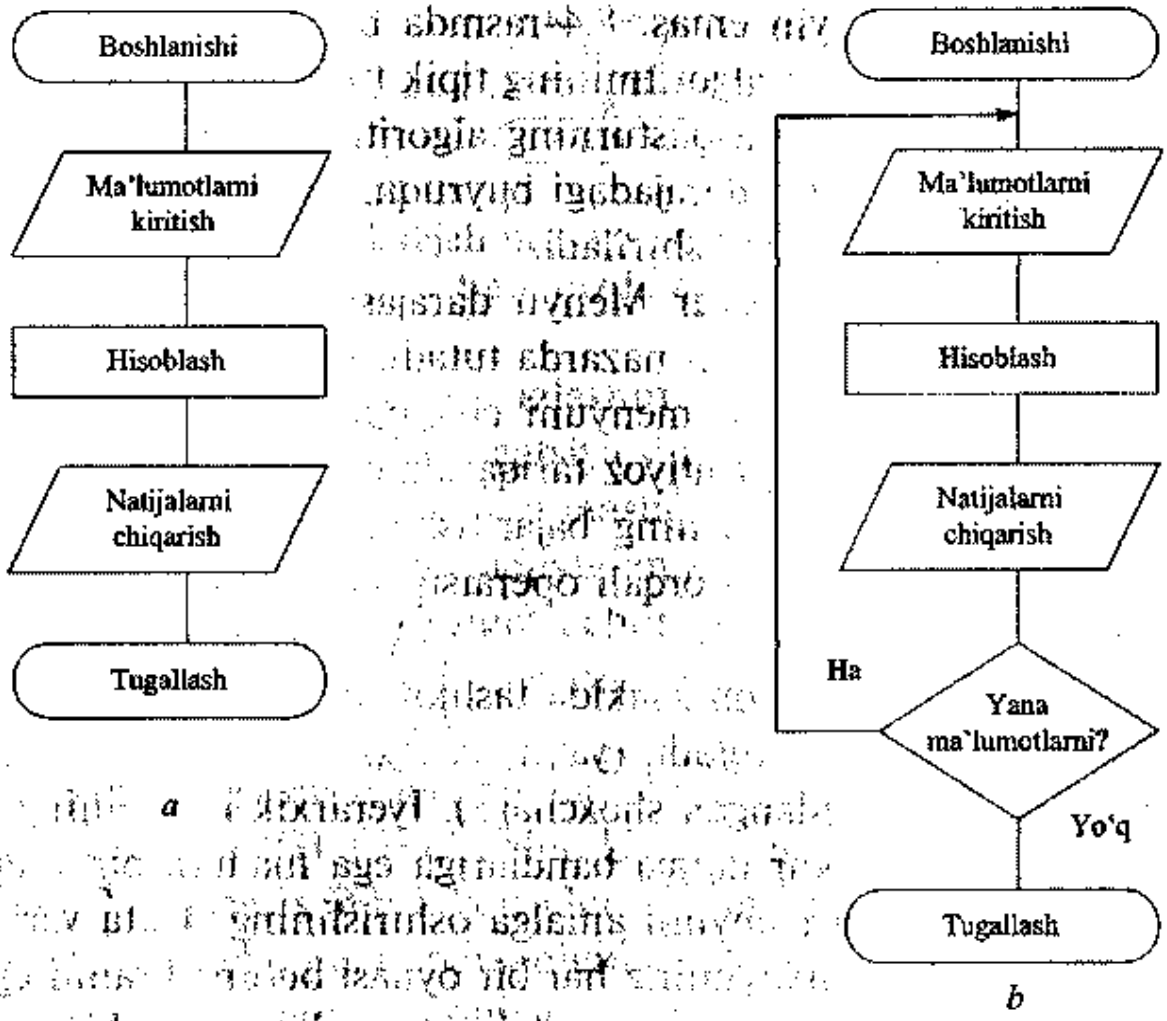
| Protsedurali mo'ljallangan foydalanuvchi interfeyslari | Obyektga mo'ljallangan foydalanuvchi interfeyslari |
|---|---|
| Foydalanuvchilarni topshiriqni bajarish uchun zarur bo'lgan funksiyalar bilan ta'minlaydi. | Foydalanuvchilarga obyektlar bilan o'zaro harakatlanish imkoniyatini beradi |
| Topshiriqqa urg'u beriladi | Kirish ma'lumotlari va natijalariga urg'u beriladi |
| Piktogrammalar ilova, darcha yoki operatsiyalarni taqdim etadi | Piktogramma obyektlarini taqdim etadi |
| Papkalar va ma'lumotnomalar mazmuni jadval hamda ro'yxatlar yordamida aks ettiriladi | Papka va ma'lumotnomalar obyektlarining vizual komponentlari sanaladi |

8.1-jadvalda protsedurali va obyektga mo'ljallangan turdagi foydalanuvchi interfeyslari modellarining asosiy farqi sanab o'tilgan.

Protsedurali mo'ljallangan interfeyslarning quyidagi uchta turi farqlanadi: sodda, menyu va erkin navigatsiyali.

Foydalanuvchi bilan konsol rejimda o'zaro harakatni tashkil etuvchi interfeyslar «sodda» interfeyslar deb yuritiladi.

Odatda bunday interfeyslar dasturiy ta'minotning ish ssenariysini tashkil etadi. Masalan, ma'lumotlarni kiritish, topshiriqni yechish, natijani chiqarish (8.3, a-rasm). Interfeyslarni ma'lumot bilan ta'minlovchi uzviy jarayonni yagona chetlanishi – bu ma'lumotnomalarning bir necha to'plamlarini qayta ishlash uchun siklni tashkil etish sanaladi. (8.3, b-rasm). Hozirda bunday interfeyslar dasturlashni o'qitish jarayonida yoki butun dastur bitta funksiya (masalan, ayrim tizimli utilitlar)ni amalga oshirish uchun foydalaniladi.




**8.3-rasm.** Sodda interfeysga ega dastur algoritmining tipik tuzilmasi: a – ketma-ket; b – takrorlash imkoniyati bilan.

*Interfeys menyu* – sodda interfeysdan farqli ravishda dastur orqali kiritiladigan maxsus ro'yxatdan kerakli operatsiyani tanlash imkoniyatini beradi. Bu interfeyslar ko'pgina ish ssenariylarini amalga oshirishni mo'ljallaydi. Uning oqibatlari foydalanuvchi tomonidan aniqlanadi.

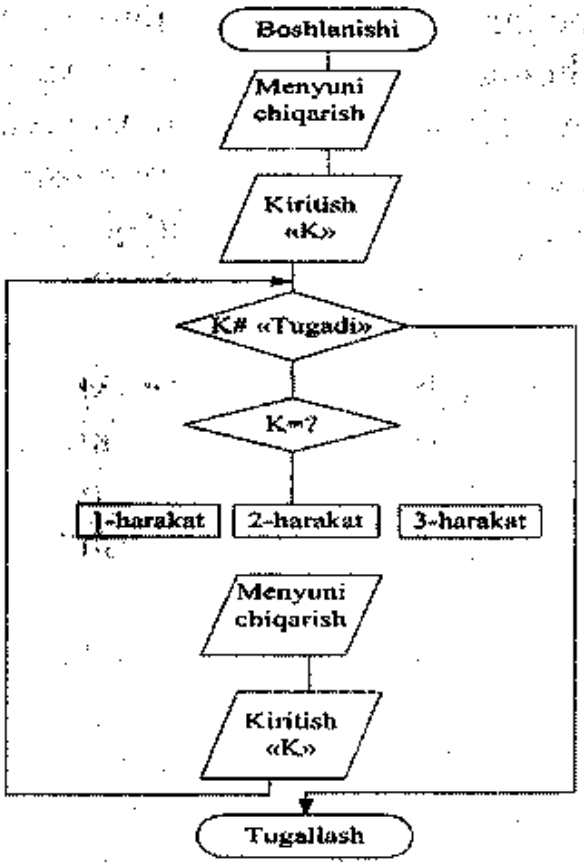
Bitta darajali va iyerarxik menyu farqlanadi. Birinchisi, variantlar kam bo'lganda hisoblash jarayonini nisbatan sodda boshqarish uchun foydalaniladi. Ular bitta turdagi operatsiyalarni o'z ichiga oladi. Masalan: Tuzish, Ochish, Yopish va h.k.

Ikkinchi xil menyu variantlar soni yoki ularning farqlari, masalan, operatsiya va fayl, operatsiya va ushbu fayllarda saqlanuvchi fayllar o'rtasida farq ko'p bo'lganda foydalaniladi. Mazkur turdagi interfeyslarni dasturlash tuzilmaviy yondashuv doirasida amalga oshirish qiyin emas. 8.4-rasmda bitta darajali menyuni tashkil etuvchi dastur algoritmining tipik tuzilmasi ko'rsatilgan.

Ko'p darajali menyu dasturning algoritmi odatda, darajasiga ko'ra tuziladi. Har bir darajadagi buyruqni tanlash bitta darajali menyudagi kabi amalga oshiriladi.

Interfeys menyu dastur Menyu darajasida yoki operatsiyani bajarish holatida bo'lishini nazarda tutadi. «Menyu darajasi» holatida tegishli darajadagi menyuni chiqaradi va menyuning kerakli bandini tanlaydi. Imtiyoz tariqasida ba'zan foydalanuvchiga ssenariy va (yoki) dasturning bajarilish darajasidan qat'i nazar Esc tugmachasini bosish orqali operatsiyani tugallash imkoniyati beriladi.

Menyuni daraxtsimon shaklda tashkil etish qat'iy cheklangan navigatsiyani nazarda tutadi. (ya'ni, «yuqoriga» – daraxt ildiziga yoki «pastga» – tanlangan shoxchaga). Iyerarxik menyuning har bir darajasiga mazkur daraja bandlariga ega ma'lum bir «oyna» mos keladi. Bunda menyuni amalga oshirishning ikkita varianti bo'lishi mumkin. Menyuning har bir oynasi butun ekranni egallaydi yoki ekranda bir vaqtning o'zida turli darajadagi bir necha menyu paydo bo'ladi. Ikkinchi holatda menyu oynasi yuqori darajadagi mos keluvchi bandlarni tanlashda yuzaga keladi.




**8.4-rasm.** Bitta darajali menyuga ega dastur algoritmining tipik tuzilmasi.

Talab etilayotgan bandni izlashni amalga oshirish variantidan qat'i nazar cheklangan navigatsiya sharoitida ikkita darajadan ortiq menyu sodda topshiriq bo'lmasligi mumkin.

Hozirda ham interfeys menyudan kam foydalaniladi faqat nisbatan sodda dasturiy ta'minot uchun yoki tuzilmaviy texnologiya bo'yicha hamda maxsus kutubxonalarsiz bajarilishi kerak bo'lgan ishlanmalarda qo'llaniladi.

*Erkin navigatsiyali interfeyslar,* shuningdek grafik foydalanuvchi interfeyslar (GUI – Graphic User Interface) yoki WYSIWYG (What You See Is What You Get – nimani ko'rib tursang shuni olasan, ya'ni ekrandagi narsanigina qog'ozga chiqarish mumkin) deb ataladi. Bu nom mazkur turdagi interfeyslar yuqori imkoniyatga ega grafik rejimda ekranda foydalanishga mo'ljallanganligini ta'kidlaydi.

Grafik interfeyslar foydalanuvchi bilan vizual qayta aloqani amalga oshirgan holda va ekranda obyektlar va axborotlarni bevosita manipulyatsiya qilish imkoniyatini yaratgan holda dasturiy ta'minot bilan interaktiv o'zaro aloqa konsepsiyasini qo'llab-quvvatlaydi. Bundan tashqari, bunday tipdagi interfeyslar dasturlar o'rtasida axborotni aralashtirishga ruxsat bergan holda, ularning muvofiqlik konsepsiyasini ma'qullaydi.

Interfeys menyudan farqli ravishda erkin navigatsiyali interfeys turli interfeys komponentalari orqali kirish mumkin bo'lgan har qanday operatsiyalar holatini amalga oshirishni ta'minlaydi. Masalan, Windows interfeysini amalga oshiruvchi dastur oynasi odatda quyidagilarni o'zida saqlaydi:

- turli xil menyularni, jumladan: pastlashuvchi, tugmachali, matnli;
- ma'lumotlarni kiritishning turli komponentlari.

**Foydalanuvchi interfeysini ishlab chiqish bosqichlari.** Foydalanuvchi interfeysini ishlab chiqish ham dasturiy ta'minotni ishlab chiqishdagi quyidagi asosiy bosqichlarni o'z ichiga oladi:

- topshiriqning qo'yilishi – interfeys turi va unga nisbatan asosiy talablarni aniqlash;
- talablarni tahlil etish va spetsifikatsiyani aniqlash – foydalanish ssenariysini va interfeysining foydalanish modelini aniqlash;
- loyihalashtirish – kirish-chiqish jarayoni ko'rinishida dialoglar va ularni amalga oshirishni loyihalashtirish;
- amalga oshirish – interfeys jarayonlarini dasturlash va testdan o'tkazish.

## 8.2. Insonning axborotni qabul qilish, eslab qolish va qayta ishlash bilan bog'liq ruhiy-jismoniy xususiyatlari

Foydalanuvchi interfeysini loyihalashtirishda insonning axborotni qabul qilish, eslab qolish va qayta ishlash bilan bog'liq ruhiy-jismoniy xususiyatlarini hisobga olish lozim.

Inson miyasining ishlash tamoyillarini tadqiq etish bilan *kognitiv psixologiya* shug'ullanadi. Bu sohadagi mutaxassislar

8.8-rasmda ko'rsatilgan miyaning soddalashtirilgan axborot-protsessual modelini taklif etadi.

Tashqi dunyo to'g'risidagi axborot miyamizga juda katta hajmda kelib tushadi. Shartli ravishda «qabul qilish protsessori» deb atash mumkin bo'lgan miya qismi ong ishtirokisiz, o'tgan tajribaga taqqoslab uni doimiy qayta ishlaydi va tasvir, ovoz hamda boshqa obrazlarda ma'lumotlar saqlanuvchi joyga kiritadi. Atrof-muhitdagi har qanday kutilmagan yoki muhim sanalgan o'zgarish e'tiborimizni tortadi va shunda bizni qiziqtirgan axborot qisqa muddatli xotiraga kelib tushadi. Agar e'tiborimizni tortmasa, axborot keyingilari bilan aralashib, saqlanadigan joyidan yo'qoladi. Har qanday daqiqada e'tibor fokusi bitta nuqtada qayd etilishi mumkin. Shuning uchun bir vaqtning o'zida bir necha vaziyatni kuzatish zarurati tug'ilsa fokus kuzatilayotgan bitta elementlar boshqasiga ko'chib turadi. Bunda e'tibor «yoyilib» ketadi va qandaydir detal hisobga olinmasligi mumkin. Masalan, Windows oynasidagi chizg'ichidan foydalanib matn yoki rasmni aylantirishda, bir vaqtning o'zida ham matnga, ham chizg'ich sirg'aluvchisiga qarash kerak bo'ladi. Chunki, matn muhimroq hisoblanadi.

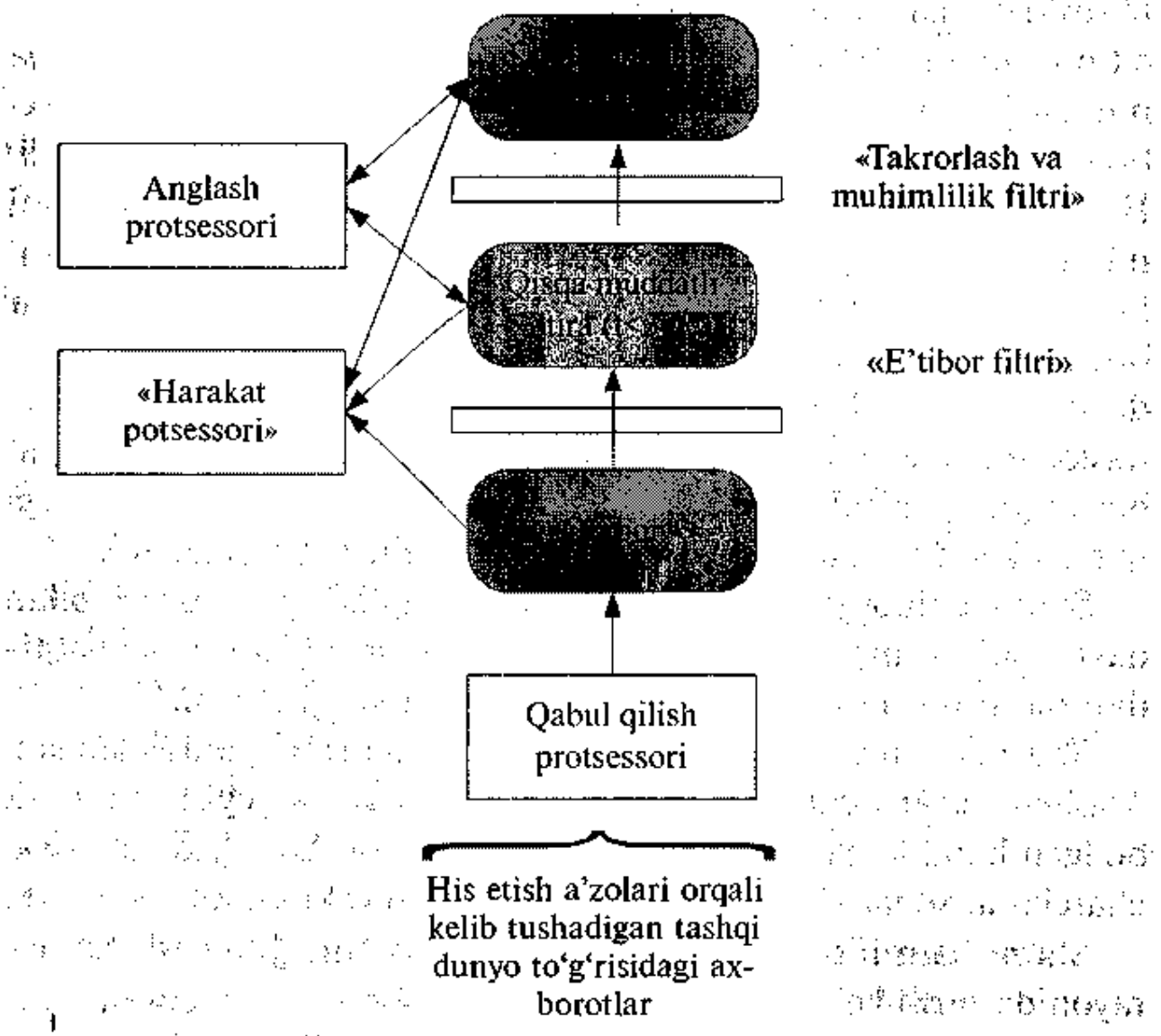
Shuni e'tiborga olish kerakki, qabul qilish protsessori bilan qayta ishlash ma'lum vaqt talab qiladi. Agar qayta ishlash vaqtidan oldinroq signal kelsa, miyamiz uni qabul qilmaydi.

Shu narsa muhimki, qabul qilish ko'proq sababga asoslangan. Masalan, agar odam och qolsa, u birinchi galda yeyish mumkin bo'lgan hamma narsaga e'tiborini qarataveradi. Yoki haddan ziyod charchasa, xonaga kirishi bilan avvalo, divan yoki krovatga qaraydi.

Shuni ham hisobga olish lozimki, axborotni qayta ishlash jarayonida miya kelayotgan ma'lumotlarni avvalgilari bilan taqqoslaydi. Masalan, insonga ramzlar ketma-ketligi A,V,S tarzda ko'rsatilsa, u 13 ning o'rniga V ni qabul qilishi mumkin.

Kadrlar almashganda miya ma'lum bir vaqt qobiqqa o'ralib oladi va nisbatan muhim detallarni ajratib, yangi manzarani «o'zlashtiradi». Demak, agar foydalanuvchining tezkor javob ta'siri zarur bo'lsa, manzarani darhol o'zgartirish kerak emas.

Qisqa muddatli xotira – insonning «axborotni qayta ishlash tizimi» borasidagi eng «tor» joyi. Uning hajmi taxminan bir-biri bilan bog'lanmagan 7±2 obyektga teng. Qisqa muddatli xotira miyaning o'ziga xos tezkor xotirasi sanaladi. Lekin talab etilmagan axborot unda 30 sekunddan ortiq saqlanmaydi. Biz uchun muhim bo'lgan biror axborotni esdan chiqarmaslik uchun biz odatda uni xotirada «yangilab», ichimizda qaytaramiz.




**8.5-rasm.** Miyaning soddalashtirilgan axborot-protsessual modeli.

Shunday qilib, interfeyslarni loyihalashtirishda shuni hisobga olish kerakki, ko'pchilik odamlar uchun beshtadan ortiq raqamni (7–2) yoki ayrim harflarni eslab qolish qiyin. Odamlar o'z tushun-

chasining har bir faoliyatiga o'sha faoliyat qanday bajarishi kerak bo'lsa, ma'lumotni shunday kiritadi. Bu tushuncha – faoliyat modeli – insonning oldingi tajribasiga asoslangan bo'ladi. Bunda ko'pgina modellar insonning uzoq muddatli xotirasida saqlanadi.

Uzoq muddatli xotiraga kuchli his-hayajon bilan bog'liq ma'lumotlar yoki axborotlar yozilib qoladi. Insonnnig uzoq muddatli xotirasi – cheklanmagan hajm va vaqtda axborotlarni saqlash joyi sanaladi.

Biroq, ushbu axborotga kirish oson emas, xotiradan axborotni olish mexanizmi assotsiativ mazmunga ega. Axborotni eslab qolishning maxsus metodikasi (menomonika) xotiraning aynan shu xususiyatidan foydalanadi. Axborotni eslab qolish uchun uni xotirada saqlanib qolingan ma'lumotlarga «bog'lab» qo'yadi.

Demak, uzoq muddatli xotiraga kirish qiyin ekan, foydalanuvchi kerakli axborotni eslab qolishiga emas, uni bilishiga ko'proq e'tibor qaratish maqsadga muvofiq. Shu sababali menyu kabi interfeyslardan keng foydalaniladi.

**Ranglarni qabul qilish xususiyatlari.** Rang inson ongida his-hayajon bilan uyg'unlashib ketadi. Ma'lumki, qizil, sariq, olov ranglar yorqin ranglar sanalib, inson ruhiyatini qo'zg'atadi. Ko'k, binafsha va kul ranglar esa sovuq ranglar bo'lib, tinchlantiradi. Umuman, rang inson uchun kuchli qo'zg'atuvchi omil hisoblanadi. Shuning uchun interfeysda ranglarni ehtiyotkorlik bilan qo'llash lozim.

Shuni hisobga olish lozimki, ranglar ko'pligi e'tiborni tortadi, ammo, tez charchatadi. Shuning uchun foydalanuvchi uzoq vaqt ishlaydigan oynani haddan ziyod yorqin bezamaslik kerak. Ayni paytda, insonning ranglarni qabul qilish xususiyatini hisobga olish kerak. Masalan, har o'nta odamdan biri biror-bir rangni yaxshi farqlay olmaydi. Shu sabab tegishli hollarda foydalanuvchiga ranglarni moslash imkoniyatini berish kerak.

**Ovozni qabul qilishning o'ziga xos xususiyatlari.** Odatda, interfeyslarda ovoz turli maqsadlarda, ya'ni e'tiborni tortish uchun, foydalanuvchining ayrim holatini ta'minlovchi predmet sifatida,

qo‘shimcha axborot manbai sifatida foydalanishi mumkin. Ovozdan foydalanganda shuni hisobga olish kerakki, ko‘pchilik insonlar ovoz signallariga juda sezgir. Shu bois ovoz bilan bog‘liq ko‘makchi vositani yaratishda uni o‘chirish imkoniyatini hisobga olish lozim.

**Vaqtni subyektiv qabul qilish.** Vaqtni subyektiv qabul qilish insonga xos xususiyat. Qabul qilinadigan va qayta ishlanadigan axborot tezligi hamda hajmi bilan ichki vaqt bog‘liq bo‘ladi. Ish bilan band odam, odatda vaqt o‘tishini sezmay qoladi. Biroq kutish holatida vaqt go‘yo hech o‘tmayotgandek tuyiladi. Bu shu bilan bog‘liqki, bu vaqtda miya axborot vakuumi holatida bo‘ladi (charchash ham shu holatga olib kelishi mumkin: axborot kelib tushadi, ammo ko‘proq qayta ishlanadi, shuning uchun vaqt o‘tishi sekinlashgandek bo‘ladi).

Shu narsa isbotlanganki, kutish paytida 1-2 foydalanuvchi chalg‘ishi, «fikrni yo‘qotishi» mumkin. Bu ish natijalariga salbiy ta’sir ko‘rsatadi va charchoqni kuchaytiradi. Chunki, har gal kutishdan so‘ng ishga qayta kirishishga ko‘p kuch sarflanadi.

Foydalanuvchini ishdan chalg‘itmagan holda kutish vaqtini qisqartirish mumkin. Masalan, unga o‘ylash uchun biror axborot taqdim etilsa bo‘ladi. Imkoniyatga ko‘ra, foydalanuvchiga oraliq natijalarni chiqarish maqsadga muvofiq. Birinchidan, foydalanuvchi ularni o‘ylash bilan band bo‘ladi, ikkinchidan, unga qarab kelajak natijalarini baholash va agar qoniqtirmasa, operatsiyalarni bekor qilishi mumkin bo‘ladi.

Foydalanuvchi «dam olishi» uchun animatsiyadan foydalanish usuli ham ma’lum. Masalan, Windows da fayldan nusxa olishda uchib yurgan barglar roligi namoyish qilinadi. Biroq, shuni hisobga olish lozimki, biror animatsiyani birinchi marta ko‘rganda qiziqarli tuyiladi. Lekin, inernetdan axborot olishda barglar «uchishi»ni yarim soat kuzatish asabga tega boshlaydi.

Kutish paytida yuzaga keladigan asabiy holatni kamaytirish uchun quyidagi asosiy qoidaga rioya etish zarur: foydalanuvchiga shuni ma’lum qilish kerakki, u buyurtma qilgan operatsi-

yani amalga oshirish ma'lum vaqt talab qiladi. Odatda buning uchun qolgan vaqt indikatori animatsiya obyektlari va qum soatda «sichqon» kursori shaklini o'zgartirishdan foydalaniladi. Tizim ishni davom ettirishga tayyor bo'lgan vaqtni aniq ko'rsatish juda muhim. Odatda buning uchun ekran tashqi ko'rinishini ancha o'zgartirishdan foydalaniladi.

Xullas, foydalanuvchi va interfeysning o'zaro harakati nafaqat insonning axborotni qabul qilish va eslab qolish bo'yicha jismoniy imkoniyati, o'ziga xos xususiyati bilan, shuningdek, interfeysning foydalanuvchi modeli hamda turli harakatlarni amalga oshirishi bilan aniqlanadi.

### 8.3. Interfeysning foydalanuvchi dasturiy modeli

Foydalanuvchi interfeysining uchta mutlaqo turli modellari mavjud: dasturchi modeli, foydalanuvchi va dasturiy model. Dasturchi foydalanuvchi interfeysini ishlab chiqishda mazkur interfeysni amalga oshirishda qanday operatsiyalarni boshqarishi va kompyuter resurslari hamda o'zining kuch va vaqtini sarflamay buni qanday amalga oshirishidan kelib chiqishi lozim. Uni dasturiy ta'minotning funksionalligi, samaradorligi, texnologiyasi ichki tuzilishi va foydalanuvchi qulayligi bilan bog'lanmagan boshqa xususiyatlari qiziqtiradi. Aynan shuning uchun mavjud dasturlarning ko'pgina interfeyslari foydalanuvchilarda jiddiy e'tiroz tug'diradi.

Sog'lom fikr nuqtayi nazaridan ish davomida foydalanuvchi o'zi kutgan narsani olsa, o'sha interfeysni yaxshi deb hisoblash qabul qilingan. Foydalanuvchining interfeys funksiyalari to'g'risidagi tasavvurini interfeysning foydalanuvchi modeli ko'rinishida bayon etish mumkin.

*Interfeysning foydalanuvchi modeli* bu bir foydalanuvchi yoki foydalanuvchilar guruhining dastur yoki dastur tizimi ishi vaqtida yuz beradigan jarayonlar to'g'risidagi to'liq tasavvurlar majmuidir. Bu model ma'lum bir foydalanuvchi tajribasining o'ziga xos xususiyatlariga asoslanadi va u quyidagilarni tasvirlaydi:

• ishlab chiqilayotgan dasturiy ta'minotning predmet sohasidagi tayyorgarlik darajasi;

• ushbu predmet sohasida operatsiyalarni bajarishning intuitiv modellari;

• kompyuterdan foydalanish borasidagi tayyorgarlik darajasi;

• kompyuter bilan ishlashdagi mavjud stereotiplar.

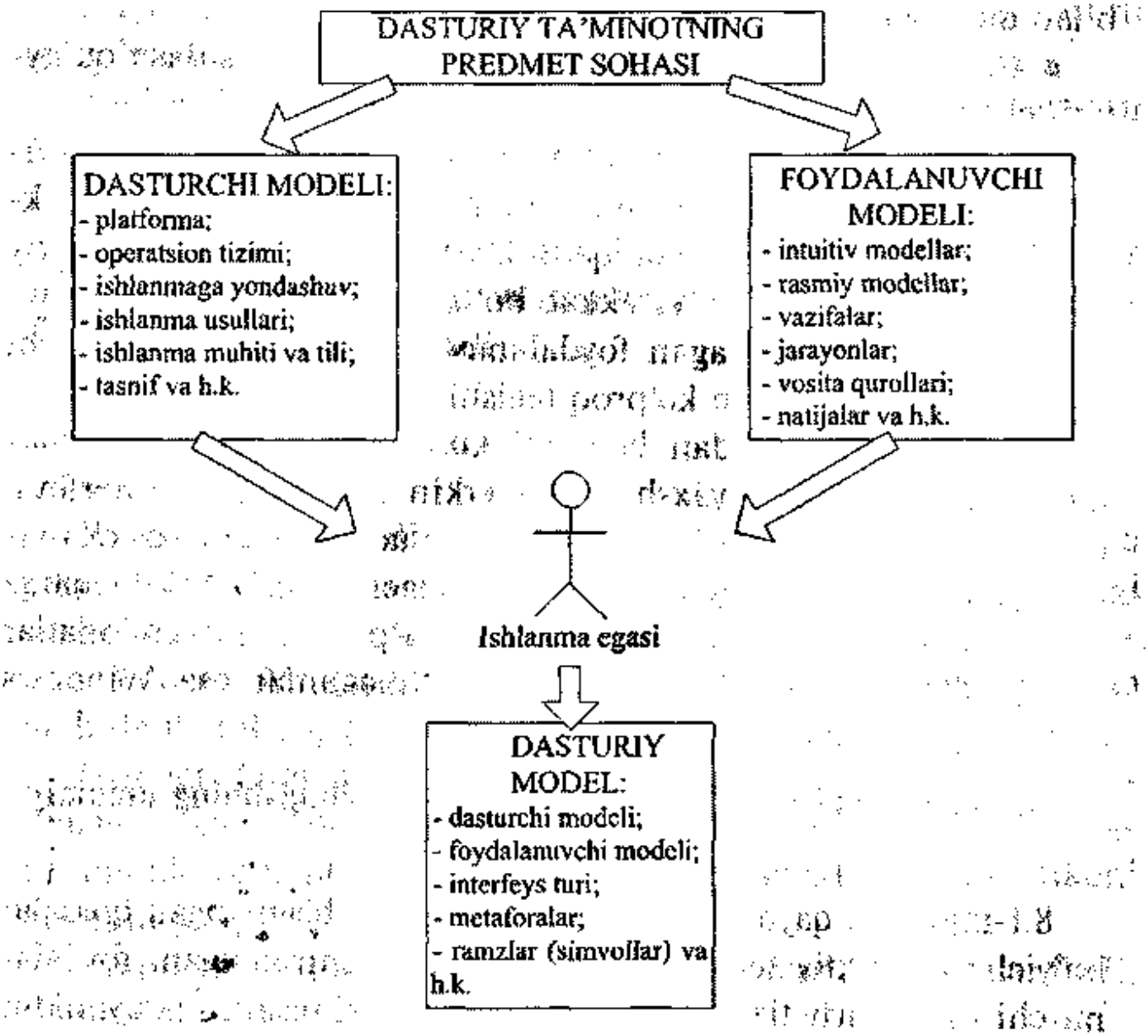
Foydalanuvchi modelini tuzish uchun ushbu tajriba xususiyatlarini o'rganish kerak. Shu maqsadda so'rovlar, testlardan foydalaniladi, hatto tasmada bajariladigan ayrim operatsiyalar jarayonida amalga oshiriladigan harakatlar natijasi qayd etiladi.

Interfeysning dasturiy modelini yaratishda shuni nazarda tutish kerakki, foydalanuvchi modelini o'zgartirish oson emas. Kompyuterni egallash borasida foydalanuvchining kasbiy darajasini va uning tayyorgarligini oshirish dasturiy ta'minotni ishlab chiquvchilar vakolatiga kirmaydi. Ammo interfeysni oqilona tuzish yuz berayotgan jarayonlar mohiyatini aks ettiradi, foydalanuvchining malakasi oshishiga ko'maklashadi.

Predmet sohasida operatsiyalarni bajarishning intuitiv modeli interfeysini ishlab chiqish uchun asos bo'lishi lozim. Shu bois ko'pchilik hollarda ularni almashtirish emas, balki aniqlash va takomillashtirish zarur. Operatsiyalarni amalga oshirishning intuitiv modellariga rioya etmaslikni istamaslik yoki uning imkoniyati yo'qligi foydalanuvchi tomonidan salbiy qabul qilinadigan, sun'iy ravishda o'ylab chiqilgan interfeyslar yaratilishiga olib keladi.

Ba'zan elementning kompyuter bilan ishlash stereotipini o'zgartirish yagona yo'l tuyiladi, ammo stereotiplardan voz kechish og'ir kechadi. Lekin, agar ayrim inqilobiy o'zgarishlar foydalanuvchi imkoniyatlarini kengaytirsa yoki uning ishini yengillashtirsa, bunga jazm qilsa arziydi. Masalan, Windows interfeyslarga o'tish juda ko'p sonli professional bo'lmagan foydalanuvchilarining kompyuter bilan ishlashini osonlashtiradi. Stereotiplarni maydalab buzib yoki qabul qilingan konsepsiyaga noto'g'ri rioya etgan holda, ishlanma muallifi aslida nima yuz berayotganini tushunmagan foydalanuvchini yiroqlashtiradi. Bunga misol tariqasida

ish stolidagi piktogramma bo'yicha dasturni chaqirish uchun «sichqon»ning o'ng tomondagi tugmachasini ikki marta bosish, yoki agar piktogramma Quick Launch (tezkor kirish) Windows paneliga chiqarilgan bo'lsa, tugmachani bir marta bosish bilan bog'liq chalkashlikni eslash kifoya.




**8.6-rasm.** Foydalanuvchi interfeysini loyihalashtirish jarayoni.

**Foydalanuvchi interfeysini baholash mezonlari.** Dasturiy ta'minotni ishlab chiqish bo'yicha yetakchi firmalar tomonidan o'tkazilgan ko'plab so'rovlar va tadqiqotlar shuni ko'rsatadiki, foydalanuvchi interfeysini baholashning asosiy mezonlari quyidagilar sanaladi:

• tizim operatsiyalarini o'zlashtirish va eslab qolishning oddiyligi axborotni o'zlashtirish vaqtini hamda xotirada saqlash davomiyligini baholaydi;

• tizimdan foydalanishda natijaga erishish tezligi «sichqon» bilan kiritiladigan yoki tanlanadigan komanda va sozlashlar soni bilan aniqlanadi;

• tizimni ishga tushirishda subyektiv qoniqish (ishlash qulayligi, charchoq va h.k).

Ta'kidlash joizki, doimo bitta va faqat o'sha paket bilan ishlaydigan professional foydalanuvchilar uchun birinchi o'ringa ikkinchi va uchinchi mezon chiqadi. Dasturiy ta'minot bilan davriy ishlaydigan va nisbatan murakkab bo'lmagan vazifalarni bajaruvchi professional bo'lmagan foydalanuvchilar uchun esa birinchi hamda uchinchi mezon ko'proq tegishli bo'ladi.

Shu nuqtayi nazardan bugungi kunda professional foydalanuvchilar uchun eng yaxshi tavsif erkin novigatsiyali interfeys, professional bo'lmagan foydalanuvchilar uchun esa bevosita manipulyatsiyalash interfeysi sanaladi. Allaqachon ma'lumki, boshqa teng shartlarda fayllarni nusxalashda ko'pchilik professionallar For tipidagi qobiqdan, professional bo'lmaganlar esa Windows obyektlarni tashish turidan foydalanadi.

## 8.4. Dialoglarni tasniflash va ularni ishlab chiqishning umumiy tamoyillari

8.1-mavzuda qayd etilganidek, dialog — bu belgilangan qoidalar bo'yicha interaktiv terminal orqali amalga oshiriladigan, foydalanuvchi va dasturiy tizim o'rtasida axborotlar almashish jarayonidir.

Dialog turi va uning shakllari farqlanadi.

**Dialog turlari.** Dialog turi «suhbatdoshlardan qay biri axborot almashish jarayonini boshqarishini belgilaydi». Muvofiq ravishda dialogning ikkita turi farqlanadi: dastur boshqaradigan va foydalanuvchi boshqaradigan dialog.

Dastur tomonidan boshqariladigan dialog qat'iy, chiziqli va daraxtsimon bo'lib, barcha muqobil variantlarni, dasturiy

ta'minotga kiritilgan dialog ssenariysini o'z ichiga oladi. Bunday dialog odatda, har bir qadamda qanday ma'lumotni kiritishni aniqlab beradigan ko'makchi ko'rsatmalar bilan kechadi.

Foydalanuvchi boshqaradigan dialog uning ssenariysi foydalanuvchiga bog'liq ekanligini nazarda tutadi. U o'ziga operatsiyani bajarish uchun kerakli tizimni qo'llaydi. Bunda tizim turli foydalanish ssenariylarini qo'llash imkoniyatini ta'minlaydi.

**Dialog shakllari.** Agar suhbatdoshlarga tushunarli bo'lmagan til mavjud bo'lmasa hech qanaqa dialogni amalga oshirib bo'lmaydi. Dialog olib boriladigan til tavsifi til yoki uning shakli tuzilishini belgilovchi qoida – *semantikani* o'z ichiga oladi. Ma'lum holatda foydalaniladigan sintaksis va semantikaga ko'ra dialogning quyidagi uchta turi farqlanadi:

- iborali;
- direktiv;
- jadvalli.

Iborali shakldagi dialog foydalanuvchi bilan tabiiy yoki unga yaqin tilda «muloqot» qilishni nazarda tutadi. Bunday shakldagi dialog buyruq, darak va so'roq gaplar hamda savollarga javoblardan tuziladi. Muloqot erkin formatda amalga oshirilishi, ammo alohida iboralar qayd etilishi mumkin.

Hozirgi darajada tabiiy tilda dialogni tashkil etish hal etilmagan vazifa. Chunki tabiiy til juda murakkab va uning sintaksis hamda semantikasini yetarli darajada shaklga solish imkoni bo'lgani yo'q.

Ko'pincha, bitta murakkablikka ega javoblardan foydalaniladi, masalan:

Dastur: Yoshingizni (to'liq yilini) kiriting:
Foydalanuvchi: 48.

Bunday vaziyatda dastur cheklangan – tabiiy tildan foydalanadigan sintaksis va semantikaning cheklangan bayonini o'z ichiga oladi. Mazkur holatda «butun musbat son» sintaksis tushunchasini aniqlash va son ifodasi uchun cheklash qo'yishning o'zi yetarli.

Biroq, asosan intellektual tizim uchun tabiiy tilning cheklangan ko'plik gaplari bazasida interfeyslarni yaratish borasida ayrim tajribalar bor. Bunday interfeyslarda amalga oshiriladigan dialoglar tilining sintaksis va semantikasi murakkab.

Bunday hollarda iboralarni qayta ishlashda «so'z – shakl» tushunchasidan foydalaniladi. So'z-shakl, bu ikki suhbatdosh o'rtasidagi matnning oraliqlar yoki boshqa belgilar qo'yilgan parchalari. To'liq matn mazmuni bilan aloqada bo'lmagan so'z-shaklni qayta ishlash *morfomantiqiy tahlil* deb ataladi.

Marfomantiqiy tahlilining ikki usuli mavjud:

- *deklarativ usul* – lug'atdagi har bir so'zning turli so'z shakllari mavjudligini nazarda tutadi. Shunda tahlil lug'atidagi so'z-shaklni qidirishga keltiriladi. Mazkur usul lotin, rus va boshqa til alifbolaridagi ham bosma, ham yozma harflardan tuzilgan xabarlarni qayta ishlash imkoniyatini ta'minlaydi;
- *protsedurali usul* – joriy so'z-shaklning keyinchalik identifikatsiyalanadigan asosini ajratishni nazarda tutadi.

So'z-shakl ma'nosi aniqlangach, xabar sintaksis tahlil etiladi. So'ngra natijalarga ko'ra, uning sintaksis tuzilmasi belgilanadi, ya'ni gap tahlil qilinadi.

Shundan keyin *semantik tahlil* amalga oshiriladi, ya'ni so'z-shakllar o'rtasidagi mazmuniy munosabatlar aniqlanadi. Ayni paytda gap mazmunini belgilovchi asosiy o'zaklar ko'rib chiqiladi.

Shunday qilib, dialogning ibora shaklini amalga oshiruvchi interfeys quyidagilarni bajarishi zarur: xabarni tabiiy til shaklidan ichki tasavvurga va qayta shaklga solishi, foydalanuvchi va tizim xabarlarini tahlil etish, uyg'unlashtirish, dialogning o'talgan qismini kuzatish va xotirada saqlash.

Tabiiy tilning boyligidan foydalanishda ibora shaklining asosiy kamchiligi quyidagilar sanaladi:

- resurslarning ko'p sarflanishi;
- ta'riflarni bir xil interpretatsiyalash kafolatining yo'qligi;
- grammatik jihatdan to'g'ri uzun iboralarni kiritish zaruriyati.

Ibora shaklining asosiy afzalligi tizim bilan nisbatan erkin muloqotda bo'lishida.

Direktiv shakldagi ibora maxsus ishlab chiqilgan rasmiy til komandalari (direktivlari)dan foydalanishni nazarda tutadi. Bu holatda kombinatsiyalashgan ma'lumotlarni bayon etuvchi ushbu tilning gaplari *komanda* (buyruq) deb ataladi.

Komandani quyidagicha kiritish mumkin:

- maxsus ishlab chiqilgan matn qatorlari ko'rinishida. Masalan, buyruqli qatorlarda kiritiladigan MS DOS buyruqlari;
- ayrim klaviatura klavishlari kombinatsiyalarini bosish orqali. Masalan, zamonaviy Windows – ilovalarining «tezkor ruxsat» kombinatsiyasi;
- «sichqon» bilan manipulyatsiya qilish vositasida. Masalan, piktogrammani «tashib o'tkazish»;
- Ikkinchi va uchinchi usullarni kombinatsiyalash bilan.

Direktiv shaklnning asosiy *afzalliklari:*

- kiritiladigan axborotning nisbatan kichik hajmda bo'lishi;
- moslashuvchanlik – operatsiyani tanlash imkoniyati bu holatda ruxsat etiladigan buyruqlar majmui bilangina cheklangan;
- foydalanuvchi boshqaradigan dialogga mo'ljallanganlik;
- ekranda kichik o'lchamda foydalanish yoki umuman foydalanmaslik;
- boshqa shakllar bilan qo'shilish *imkoniyati.*

Direktiv shakl kamchiliklari:

- ekranda yo'l ko'rsatuvchi buyruqlarning amalda yo'qligi. Bu kiritiladigan buyruqlar va ularning sintaksisini eslab qolishni talab etadi;
- inisiallangan jarayonlar holati to'g'risida javob aloqasining deyarli, to'liq yo'qligi;
- matnli axborotni kiritish ko'nikmasining zarurligi yoki «sichqon» vositasida manipulyatsiyani amalga oshirish;
- foydalanuvchi bilan aloqani sozlash imkoniyatining yo'qligi.

Tadqiqotlar shuni ko'rsatadiki, direktiv shakl odatda buyruq yoki klavishlar kombinatsiyasini tez-tez qo'llovchi professional-foydalanuvchilar uchun juda qulay. Shaklning asosiy afzalligi (moslashuvchanlik va yaxshi, vaqtinchalik tavsif) ayniqsa shu holatda aniqroq ko'rinadi.

*Jadvalli dialog shakli* foydalanuvchi taklif etilgan dasturdan javob tanlashini nazarda tutadi. Bu xil shakldagi dialog tili sodda sintaksis va bir xil semantikaga ega bo'lib, uni amalga oshirish oson. Bu shakl foydalanuvchi uchun ham qulay. Chunki professional foydalanuvchi yoki aniq dasturiy ta'minotni kam qo'llaydigan, professional bo'lmagan foydalanuvchi uchun ham muhim. Biroq jadvalli shaklni har doim ham qo'llash imkoni yo'q. Undan agar ma'lum bir savolga ko'plab javob bo'lsagina foydalanish mumkin. Ayni paytda mumkin bo'lgan javoblar juda ko'p bo'lsa (20 dan ortiq), jadval shaklini qo'llash maqsadga muvofiq bo'lmasligi mumkin.

Jadvalli shakl *afzalliklari* quyidagilar sanaladi:

- ko'makchi ko'rsatmalar mavjudligi. Bu foydalanuvchi xotirasiga ortiqcha yuk tushishini kamaytiradi. Chunki bu shakl eslab qolishga emas, bilishga mo'ljallangan;
- kiritish jarayonida xatolar sonining kamayishi. Chunki foydalanuvchi axborot kiritmaydi, balki uni so'rab ko'rsatadi;
- foydalanuvchini o'rgatish vaqti kam;
- boshqa shakllar bilan uyg'unlashish imkoniyati mavjud;
- ayrim hollarda foydalanuvchi tomonidan sozlash imkoniyati bor.

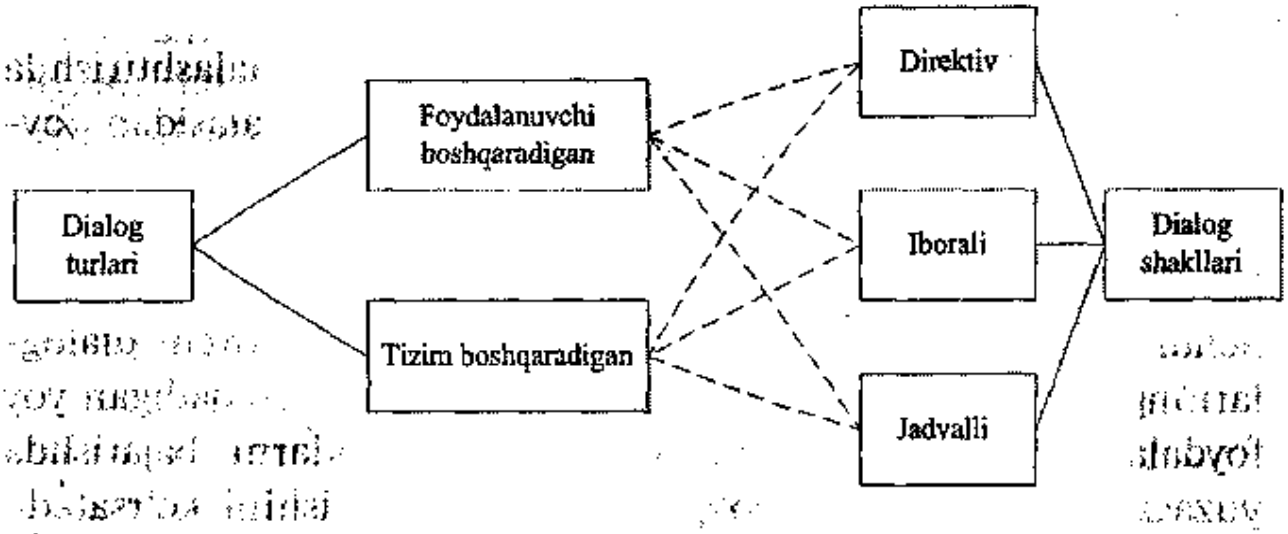
Mazkur shakl *kamchiliklari:*

- ekran bo'ylab navigatsiya ko'nikmasi bo'lishining zarurligi;
- vizual komponentalarni tasvirlash uchun ekranning nisbatan katta maydonidan foydalanish.

Shuni hisobga olish kerakki, dialog turi va shakli bir-biriga bog'liq bo'lmagan holda tanlanadi: har qanday shakl har ikki dialog turi uchun maqbul (8.7-rasm).

Biroq foydalanuvchi boshqaradigan dialog foydalaniladigan ibora shakli dialog tilining nisbatan murakkab sintaksis va se-

mantikasini nazarda tutadi. Chunki, dastur foydalanuvchini «tushunishi» lozim.




**8.7-rasm.** Dialoglar turi va uning shakllari muvofiqligi

Murakkab dasturiy ta'minot, odatda, foydalanuvchi bilan hal etiladigan vazifaga bog'liq holda turli xil tur va shakldagi dialoglar vositasida foydalanuvchi bilan o'zaro harakatlanadi. Ish jarayonida yuz beradigan dialoglardan tashqari jarayon ssenariysi buzilganda tizim yoki foydalanuvchi tashabbusiga ko'ra yuzaga keladigan dialoglar ham mavjud. Ular asinxron dialoglar deb ataladi. Odatda ulardan tizim yoki foydalanuvchidan favqulodda xabarlarni uzatish uchun qo'llaniladi.

**Dialoglarni ishlab chiqish.** Dialoglarni loyihalashtirish va amalga oshirish jarayonini quyidagi bosqichlarga bo'lish mumkin:

- ko'plab zarur dialoglar, ularning asosiy xabarlari va ssenariylarini aniqlash, mavhum dialoglarni loyihalashtirish;
- har bir dialog turi va shaklini, shuningdek, foydalaniladigan tilning sintaksis va semantikasini aniqlash, konkret dialoglarni loyihalashtirish;
- asosiy qo'shimcha qurilmani kiritish va har bir dialog uchun kirish-chiqish jarayonini loyihalashtirish, shuningdek, uzatiladigan xabarlarni aniqlash, texnik dialoglarni loyihalashtirish.

Abstrakt dialoglar asosiga avtomatlashtirilishi uchun dasturiy mahsulot mo'ljallangan texnologik jarayon idealogiyasi singdirili-

shi lozim. Aynan avtomatlashtiriladigan texnologik jarayonni tahlil etgan holda, ishlanmani ishlab chiquvchi dialoglar ssenariysini belgilaydi.

Ssenariydan tashqari abstrakt dialoglarni loyihalashtirishda interfeys holati diagrammasidan yoki dialoglar grafasidan foydalaniladi.

*Dialog grafi* – qat'iy grafga mo'ljallangan, uning har bir katak boshida ekranda aniq holat (kadr) yoki foydalanuvchi uchun mumkin bo'lgan harakatlar to'plamini izohlovchi dialoglarning ma'lum bir holati ko'rsatilgan. Yuqoridan chiqadigan yoy foydalanuvchi tomonidan ko'rsatilgan harakatlarni bajarishda yuzaga kelishi mumkin bo'lgan holatlar o'zgarishini ko'rsatadi. Yoylar tarozisi sifatida holatdan holatga o'tish va o'tish paytida bajariladigan operatsiyalar shartlari ko'rsatiladi.

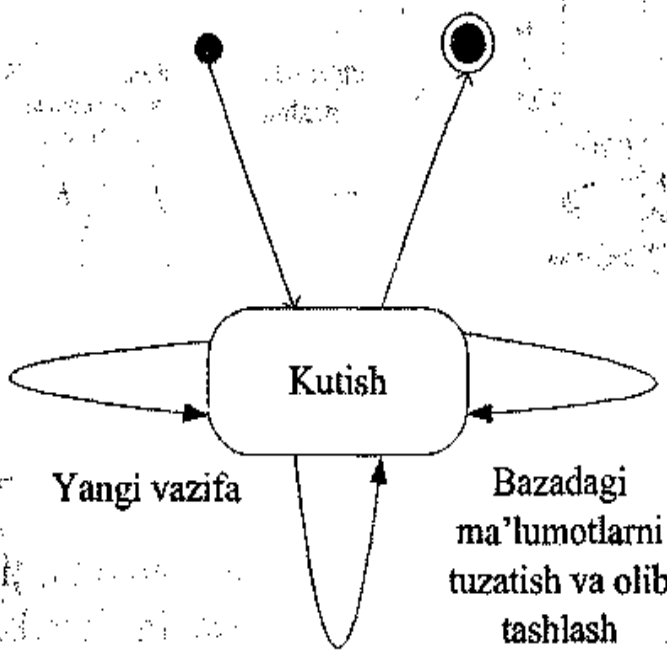
Shunday qilib, grafdagi har bir marshrut dialogining tegishli variantiga mos keladi. Ayni paytda dialogni grafik ko'rinishida ishlanma bog'liq holda turli darajadagi detallashtirish bilan amalga oshirish mumkin.

Mohiyatan, dialog grafi bu foydalanuvchi ta'sir ko'rsatganda dasturiy ta'minotning harakatini modellashtiruvchi avtomat holati grafigidir. Bunday grafiklarni taqdim etish uchun ikkita notatsiya kiritilgan: ishlanmaga tuzilmaviy yondashuv holati diagrammasi notatsiyasi va UML holati diagrammasi notatsiyasi. Ayni paytda UML notatsiyasi nisbatan kuchli sanaladi va boyitilgan dastur holatidan foydalanishga imkon beradi. Shuning uchun dialog grafigini tasavvur etish uchun yangi notatsiyani kiritmaslik maqsadida UML belgisidan foydalanamiz.

**8.2-misol.** Kombinatorli optimallashgan topshiriqlarni yechish tizimi uchun dialog grafigini ishlab chiqish.

Yuqori darajadagi dialog foydalanish variantlari diagrammasini amalga oshirishni ta'minlashi lozim. Shu bois dialog grafigining dastlabki variantini ushbu diagrammani (8.4-rasmga qarang) tahlil etish asosida tuzamiz. Foydalanuvchi natijalarni ko'rib bo'lgach, ularni saqlash yoki olib tashlashga qaror qilishi mumkin. Shu-

ning uchun ushbu operatsiyalarni yagona guruhga birlashtirish lozim. Bundan tashqari, ushbu guruhga natijalarni bosish operatsiyasini ham qo'shish maqsadga muvofiq. Ma'lumotlarni ham aynan shunday ko'zdan kechirib, ularni tuzatib yoki olib tashlab birlashtirish lozim. Ya'ni vazifa operatsiyasini alohida guruhga joylashtirgan ma'qul (8.8-rasm).

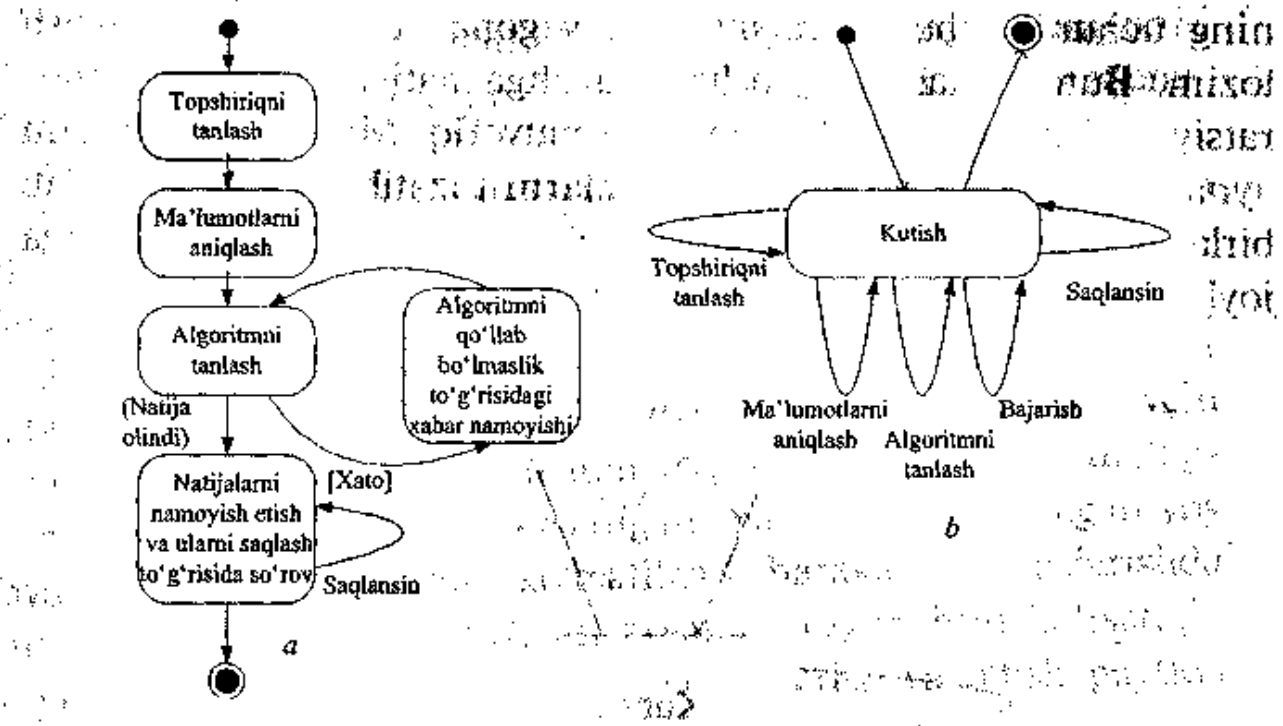



**8.8-rasm.** Kombinatorli-optimallashgan masalalarni yechish tizimining abstrakt dialog grafigi.

Yuqori darajada dialog foydalanuvchi tomonidan boshqarilishi kerak. Direktiv va jadval shakllar muqobil ravishda, foydalanuvchi istagiga ko'ra qo'llanishi mumkin. Iborali dialog shaklini qo'llash esa maqsadga muvofiq emas.

**8.3-misol.** Yangi masala dialogini detallashtirish.

6.2-mazuda masalani yechish ssenariysi keltirilgan. Uning bazasiga tizim bilan boshqariladigan dialog grafigini taklif etish mumkin (8.9, a-rasm).




**8.9-rasm.** Yangi masala abstrakt dialog grafigi: a – tizim boshqaradigan dialog; b – foydalanuvchi boshqaradigan dialog.

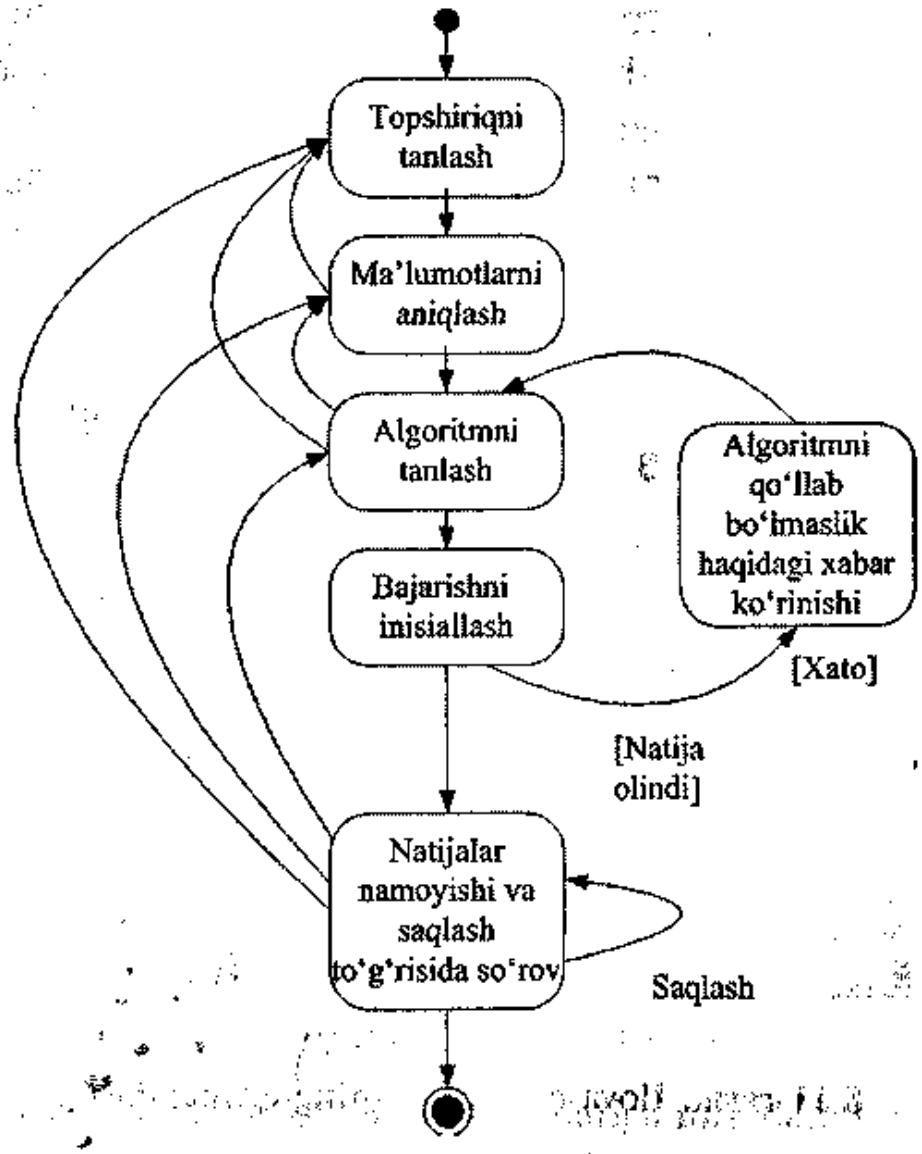
Dialog grafigini tahlil etish shuni ko'rsatadiki, tizim tomonidan boshqariladigan dialog, foydalanuvchini harakatlar variantlarini tanlashda cheklab qo'yadi. Foydalanuvchi boshqaradigan dialog har bir qadamdan so'ng harakatni tanlashni nazarda tutadi.

Shu sababli amalga oshirish uchun kombinatsiyalashgan variantdan foydalangan ma'qul. U ssenariy mavjudligini hisobga oladi, foydalanuvchi istagiga ko'ra undan chetlanishga ham yo'l qo'yiladi. (8.10-rasm).

Endi har bir qadam uchun dialog shaklining qaysi shaklidan foydalanish zarurligini aniqlash lozim. Birinchi qadam, masalani tanlash – uchta variantni o'z ichiga oladi, shu sababli jadvalli dialog shaklidan foydalangan ma'qul.

Ikkinchi qadam, **Ma'lumotlarni aniqlash** – konkretlashtirilgan, demak uning shaklini aniqlab bo'lmaydi. Uchinchi qadam, Algoritmni tanlash – yana tanlashni nazarda tutadi. Variantlar soni ko'p emas, bunda jadvalli dialog shaklini tanlash maqsadga muvofiq. Boshqa holatlarda ham aynan ushbu shakl maqbulroq bo'ladi.

Interfeyslarni loyihalashtirishning oxirgi bosqichi interfeys shaklining xususiyatini hisobga olgan holda, har bir dialog uchun kiritish-chiqarish operatsiyalarini ishlab chiqishdan iborat. Zamonaviy foydalanish interfeyslarida qo'llash mumkin bo'lgan interfeys komponentlarini ko'rib chiqamiz.




**8.10-rasm.** Kombinatsiyalashgan turdagi abstrakt dialog grafigi.

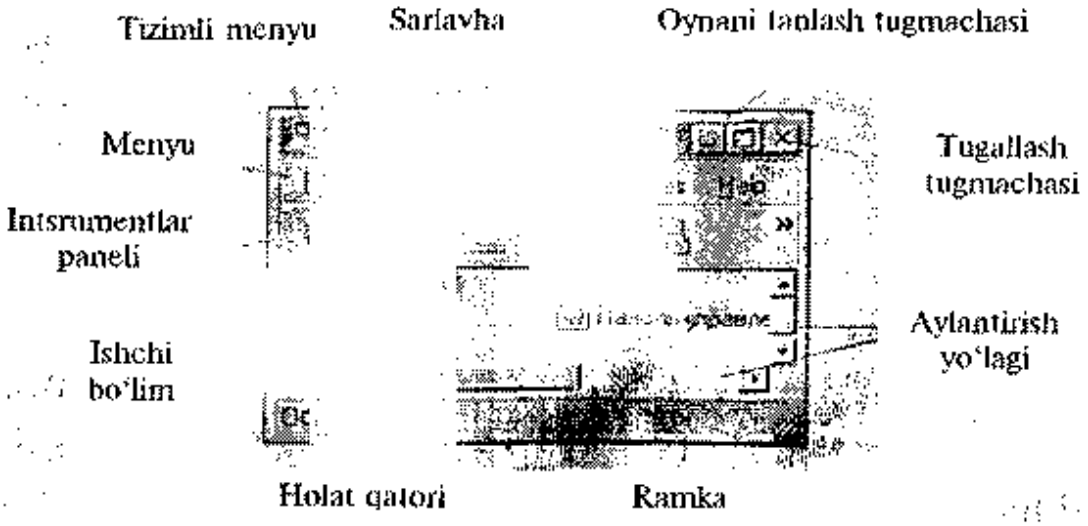
## 8.5. Grafik foydalanish interfeyslarining asosiy tarkibiy qismlari

Grafik foydalanish interfeyslari Windows, Apple, Macintosh, OS/2 va boshqa operatsiyaviy tizimlar doirasida, bunda interfeyslar uchun foydalanuvchi bilan o'zaro ta'sirning standart komponentlari to'plami ishlab chiqilgan.

Ko‘pgina zamonaviy dasturlar foydalanuvchi interfeyslari WIMP texnologiyasi bo‘yicha tuziladi: W – Windows (oyna), I – Icons (piktogramma), M – Mouse («sichqon»), P-Pop-up (tushib keladigan menyu). Shunday qilib grafik interfeyslarning asosiy elementlari quyidagilar sanaladi: oyna, piktogramma, kiritish-chiqarish komponentalari va qurilmani ko‘rsatuvchi hamda ekranda obyektlar bilan bevosita manipulyatsiya qiluvchi qurilma sifatida foydalanadigan «sichqon».

**Oyna.** Oyna – ekranning to‘g‘ri burchakli ramka bilan chegaralangan qismi. U ekran doirasida o‘lchami va joyini o‘zgartirishi mumkin. Butun oynani 4 ta kategoriyaga bo‘lish mumkin:

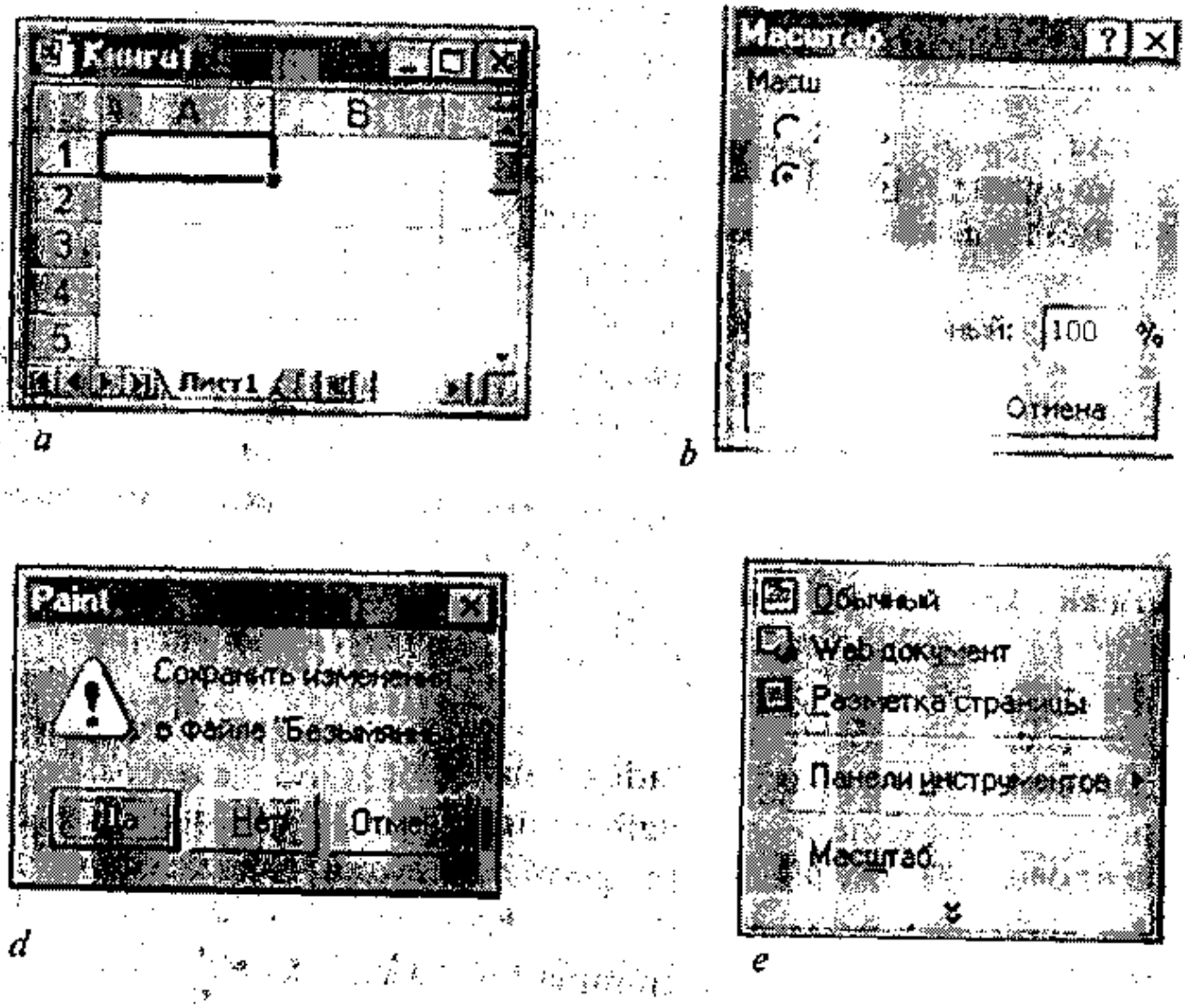
- asosiy oyna (ilova oynasi);
- sho‘ba yoki qarashli oyna;
- dialog oynasi;
- menyu oynasi.




**8.11-rasm.** Ilova oynasi va uning elementlari.

Windows ilova oynasi (8.11-rasm) odatda quyidagilarga ega bo‘ladi: oynaning ishchi sohasini chegaralovchi ramka, tizim menyusining va oyna ko‘rinishini tanlash hamda chiqish tugmachasiga ega sarlavha qatori, menyu qatori, piktografik menyu (instrumentlar paneli), qatorlarni aylantirishning enlama va bo‘ylama polosasi. Windows sho‘ba oynasi (8.12-rasm) ko‘p hujjatli dasturiy interfeyslarda (MDI) foydalaniladi va dasturiy

ta'minot bir vaqtning o'zida bir necha hujjatlar bilan ishlashni ta'minlashni nazarda tutadi. Ilova oynadan farqli ravishda sho'ba oyna menyuga ega emas. Sarlavha qatorida u bilan bog'liq hujjat yoki faylni identifikatsiyalovchi maxsus nom mavjud. Barcha sho'ba piktogrammalar oynasi bir xil.




**8.12-rasm.** Windows ko'makchi oynalariga misollar: a – sho'ba oynasi; b – dialoglar oynasi; d – axborotlar oynasi; e – menyu oynasi.

Windows dialog oynasi (8.12, b-rasm). Turli ish rejimini, kerakli parametrlarni yuklash yoki boshqa axborotlarni ko'rish uchun foydalaniladi.

U quyidagi vositalarga ega bo'lishi mumkin:

- tizim menyusi tugmachasiga ega sarlavha qatori;

• foydalanuvchiga javobni kiritish yoki chiqarish imkoniyatini beruvchi komponentlar;

• ko'rsatmalar bilan ta'minlovchi ko'makchi komponentlar (masalan, oldindan ko'rish maydonchasi yoki ma'lumot so'rash uchun tugmacha).

*Axborot oynasi* ikki xil bo'ladi: xabarlar oynasi va yordam berish oynasi. Xabarlar oynasi (8.12, b-rasm) tizim menyusi tugmachasidan tashqari, odatda xabarlar matni va foydalanuvchi reaksiyasining bitta yoki bir necha tugmachasi (masalan, Yes va No yoki Yes, No va Cancel tugmachalari)ga ega bo'ladi.

*Yordam oynasi* nisbatan murakkab tuzilmaga ega. Unda menyu, satrni aylantirish yo'lagi va axborot qismi mavjud. Ya'ni tuzilmasiga ko'ra, u ilova oynasiga o'xshash, ammo ma'lumotnoma xabari navigatsiyasini ta'minlagan holda tor, maxsus mo'ljalga egaligi bilan farqlanadi.

*Windows menyu oynasi* (8.12, e-rasm) iyerarxik menyu yoki alohida kontekst menyular panelini ochish vositasi sifatida foydalanish mumkin. Quyidagilar oynaning har bir qatoriga mos kelishi mumkin:

• buyruq;

• strelka bilan belgilanadigan navbatdagi daraja menyusi;

• uchta nuqta bilan belgilanadigan dialog oynasi.

Bundan tashqari, ayrim qatorlarga tezkor chaqiruv klavishi/ko'rsatmasi qo'shiladi.

**Piktogrammalar.** Piktogramma bu buferni aks ettiruvchi grafik tasvirga ega bo'lgan kichikroq oyna. Uning quyidagi turlari mavjud:

• dasturiy piktogramma;

• sho'ba oyna piktogrammasi;

• instrumentlar paneli piktogrammasi;

• obyektlar piktogrammasi.

Tegishli dasturlar bilan bog'liq dasturiy piktogrammalarni operatsiya tizimi boshqaradi. Jumladan, ilova oynasini Windows topshiriqlar panelidagi piktogrammaga «o'rash» yoki uni «ishchi stol»ga qayta «yoyish» mumkin.

O‘xshash ko‘p hujjatli dasturiy tizim sho‘ba oyna piktogrammalarini boshqaradi. Bu oyna dasturiy tizim bilan bir vaqtda qayta ishlanadigan turli hujjatlarga kirishni ta’minlaydi.

Instrumentlar paneli piktogramma menyu orqali tezkor chaqiruv bilan tegishli funksiyalarga ruxsat olishni takrorlaydi.

Obyektlar piktogrammasi obyektlar bilan to‘g‘ridan to‘g‘ri manipulyatsiya qilish uchun qo‘llaniladi.

Odatda, barcha piktogrammalarni «sichqon» yordamida aralashtirish mumkin. Bundan tashqari piktogrammalar bilan ishlashni osonlashtirish uchun «ko‘chib chiquvchi» ko‘rsatmalardan foydalanadi. Agar foydalanuvchi ma’lum bir muddat «sichqon»ni instrumentlar paneli piktogrammasi ustida ushlab tursa mazkur ko‘rsatmalar paydo bo‘ladi.

**Tasvirlarni bevosita manipulyatsiyalash.** *Tasvirlar bilan bevosita manipulyatsiya qilish* bu «sichqon» vositasida interfeysdagi ayrim obyektlarga jismoniy harakat bilan ta’sir ko‘rsatish buyruqlarini almashtirish imkoniyatidir. Bunda ekranning har qanday qismiga manzilgoh sifatida qarash mumkin. U kursorni yurgazish va «sichqon» klavishini bosish orqali faollashtirilishi mumkin.

Ta’sirga javob ko‘rsatishiga ko‘ra, manzilgohlar quyidagi turlarga bo‘linadi:

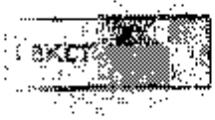
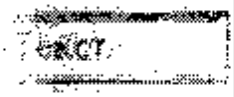
- ko‘rsatma va tanlov (piktogrammani yeyish, faol oynani aniqlash va h.k);
- shatakka olish va «rezinka ip» (obyekt yoki uning chegaralarini ko‘chirish);
- ekran tugmachalari va «sirg‘aluvchan» to‘siqlar (diskret yoki davriy qaytariladigan harakatlarni amalga oshirish. Masalan, ekranning ma’lum bir qismini faollashtirishga mo‘ljallangan operatsiyali rasmlarni bajarish).

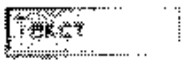
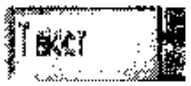
Grafik interfeyslarda dinamik vizual signallar ekrandagi tasvirlarning o‘zgarishidan iborat. Ushbu signallarning asosiy maqsadi foydalanuvchilarga qo‘shimcha ma’lumotlar berishdir. Ma’lum bir operatsiyalarni bajarishda «sichqon» kursori tasvirini

o‘zgartirish bunga misol bo‘la oladi. Yana bir misol – tugmachani bosishda tugmacha tasvirini o‘zgartirish.

**Kiritish-chiqarish komponentlari.** Yuqorida eslatib o‘tganimizdek, ilova oynasiga axborotni kiritish-chiqarish uchun foydalaniladigan maxsus komponentlar joylashtirilishi mumkin. Deyarli har qanday zamonaviy dasturiy ta’minot interfeysi bir necha menyularni o‘z ichiga oladi. Jumladan, asosiy yoki «Pastlashuvchi» iyerarxik menyu, piktografik menyu (instrumentlar paneli) va turli vaziyatlar uchun matn menyusi. Bu menyularning har biri jadval shaklidan foydalangan holda, foydalanuvchi bilan dialogni amalga oshiruvchi kiritish-chiqarish komponentlariga ega.

*8.2-jadval*

| Komponent | Tashqi ko‘rinish | Amalga oshiriladigan shakl | Foydalanishning o‘ziga xos xususiyatlari |
|---|---|---|---|
| Label – belgi (metka) |  | Iborali | Qoidaga ko‘ra, ish jarayonida o‘zgarmaydigan xabarlarni chiqarish |
| Edit – Bir qatorli muharrir |  | Iborali | Qisqa xabarlarni kiritish-chiqarish: so‘zlar, sonlar va h.k. |
| Memo – ko‘p qatorli muharrir |  | Iborali | Matnli axborotni kiritish-chiqarish |
| Button – tugmacha |  | Jadvalli | Operatsiyalarni initsiallash |
| ChekButton – tanlash elementi |  | Jadvalli | Opsiyani tanlash yoki bekor qilish |
| RadioGroup –radiotugmacha guruhi |  | Jadvalli | Agar variantlar soni 6–10 tadan oshmasa, ulardan birini tanlash |

| | | | |
|---|---|---|---|
| ListBox – ro'yxat |  | Jadvalli | Bitta yoki bir nechta variantni tanlash |
| TrackBar – ko'rsatkich |  | Jadvalli | Berilgan interval-dan sonli belgini tanlash |
| TabControl, PageControl – ko'rsatkichlar |  | Jadvalli | Parametrlar soni ko'p bo'lganda navigatsiyani yax-shilash |
| DBNavigator – navigator |  | Jadvalli | Jadval bo'yicha navigatsiyalash |
| ComboBox – qo'shimcha imkoni-yatga ega ro'yxat |  | Kombinatsi-yalash | Ro'yxatdan tanlash yoki belgini kiritish |
| SpinEdit –Belgini ko'paytirish yoki kamaytirish im-koniyatiga ega bir qatorli muharrir |  | Kombinatsi-yalash | Belgini kiritish yoki o'zgartirish |

Iyerarxik menyudan agar operatsiyalar soni 5–8 tadan oshsa (IBM firmasi tavsiyasiga ko'ra) bajariladigan dasturiy ta'minot operatsiyalarini tashkil etish hamda foydalanuvchiga ular sharhi-ni yetkazish uchun foydalaniladi. Instrumentlar paneli va matn mazmuni menyusi tez-tez foydalaniladigan buyruqlarga tezkor kirish uchun qo'llaniladi.

Menyudan tashqari interfeysda kiritish-chiqarishning boshqa komponentlaridan foydalaniladi. Bu komponentlarni qanday dialog shaklini amalga oshirishiga ko'ra uchta guruhga bo'lish mumkin: iborali, jadvalli va aralash dialog. Dialogning direktiv shakli, odatda, klavishlar kombinatsiyasini kiritish yoki pikto-grammani ko'chirishni nazarda tutadi. Chunki kiritish-chiqarish komponentlaridan foydalanishni talab qilmaydi.

8.2-jadvalda Windows interfeysining WINP asosiy komponentlari keltirilgan va ulardan foydalanish bo'yicha tavsiyalar berilgan.

## 8.6. Grafik foydalanish interfeysida dialoglarni amalga oshirish

Qoidaga ko'ra, rivojlangan foydalanish interfeysiga ega murakkab dasturiy ta'minot foydalanuvchi va tizim boshqaradigan ikki xil dialoglarni qo'llaydi.

**Foydalanuvchi boshqaradigan dialoglarni amalga oshirish.** Foydalanuvchi boshqaradigan dialoglarni amalga oshirish uchun turli xil menyular qo'llaniladi. Jumladan, asosiy menyu, instrumentlar paneli menyusi, matn va tugmachalar menyusi. Muqobil menyu sifatida direktiv dialog shaklidan foydalangan ma'qul. Bundan tashqari klaviatura menyusini boshqarish imkoniyatini nazarda tutish lozim. Agar foydalanuvchi tizim bilan ishlash vaqtining asosiy qismini matn yoki ma'lumotlarni kiritishga, ya'ni klaviatura bilan ishlashga sarflasa, bu ayniqsa juda muhim.

*Menyu.* Menyu grafiklar asosida ishlab chiqilayotgan dasturiy ta'minot dialoglarini loyihalashtiradi. Bunda, agar operatsiyalar soni 5 tadan oshmasa, odatda tugmachalardan foydalanadi. Va nihoyat, agar amalga oshiriladigan operatsiyalar 10 tadan oshib ketsa, «Pastlashuvchi» ikki darajali iyerarxik menyu amalga oshiriladi.

*Pastlashuvchi menyu.* Iyerarxik menyuning birinchi darajasi operatsiyalar asosiy guruhining nomlarini o'zida saqlashi lozim. An'anaga ko'ra, birinchi band – Fayl, ikkinchisi – Tuzatish, uchinchisi – Ko'rinish, oxirgisi – Ma'lumotnoma sanaladi. Bandlarning bunday taqsimlanishi faylga joylashtirilgan, ma'lumotlarni qayta ishlash dasturi uchun xos narsa (masalan, matn va grafik muharrirlar). Oxirgi paytlarda bandlarning bunday taqsimlanishi tufayli muammo yuzaga kelmoqda. Chunki, ko'pgina dasturlar an'anaviy usuldagi ma'lumotlar bilan allaqachon ishlamaydi. Masalan, 8.4-misolda ma'lumotlar faylda emas, ma'lumotlar bazasida joylashgan.

Iyerarxik menyu darajasining soni 2–3 tadan oshmasligi kerak. Chunki, daraja soni ko'p bo'lganda, talab etilayotgan operatsiyani

izlab topish qiyin kechadi. Shu sababli menyu oynasidagi operatsiyalar soni 70–80 tadan oshmagani ma'qul.

Agar operatsiyalar soni 70–80 tadan oshsa, bunday ko'p sonli operatsiyalar bilan ko'rgazmali menyuni qanday tuzish borasida muammo kelib chiqadi. Microsoft Word mutaxassislari tomonidan qiziqarli yechim taklif etilgan. Ular adaptiv iyerarxik menyuni amalga oshirishdi. Mazkur menyuning ikkinchi darajasi mundarijasi muntazam o'zgarib, faqat foydalanuvchi foydalanadigan operatsiyalarni aks ettiradi. Agar foydalanuvchi kerakli operatsiyani topa olmasa, bir necha sekunddan so'ng yoki maxsus Word tugmachasi bosilgach menyu oynasi to'liq namoyon etiladi.

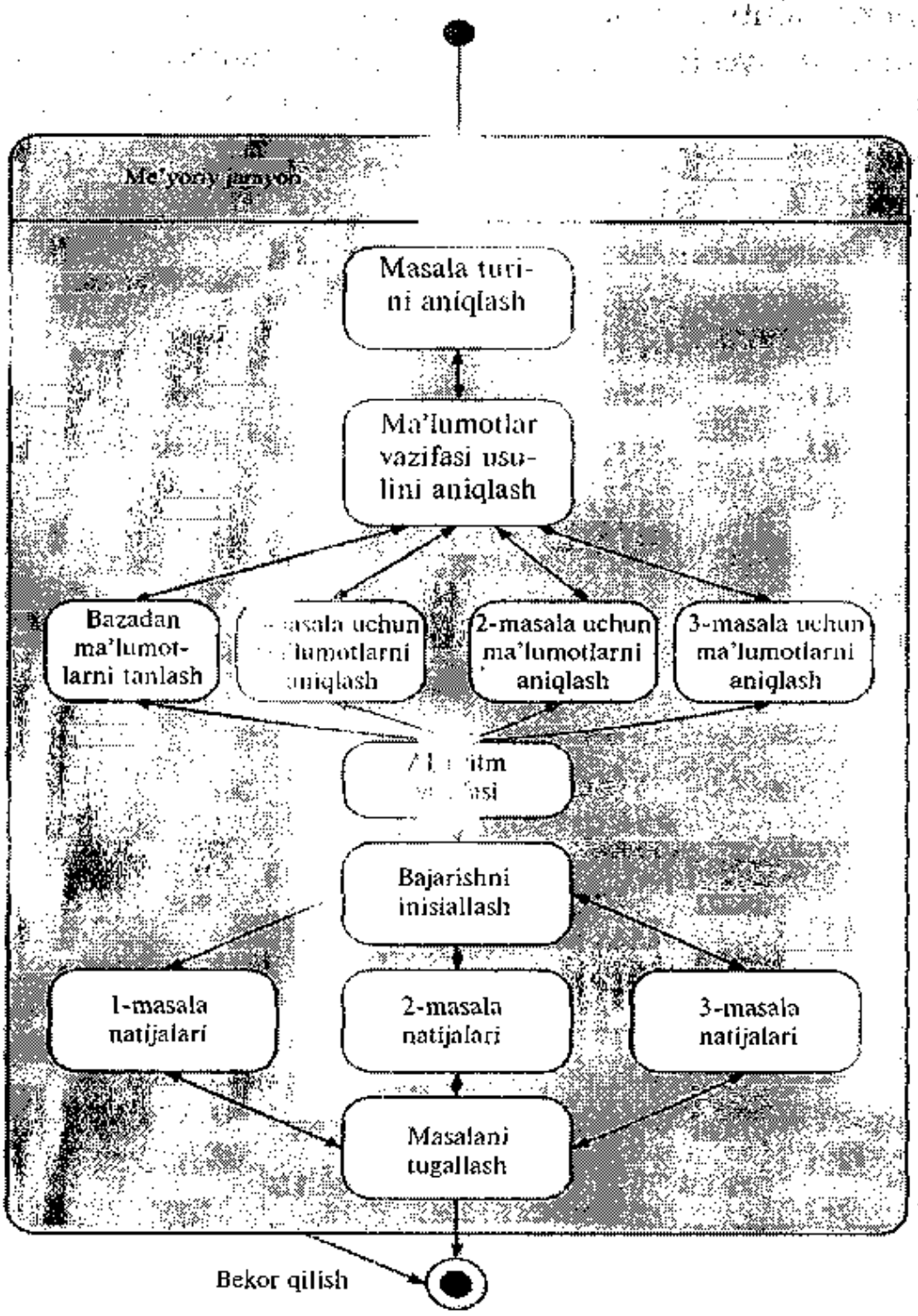
**8.4-misol.** Kombinatorli – optimallashgan topshiriqlarni yechish tizimning asosiy menyusini ishlab chiqish.

Foydalanuvchiga bir necha yechim variantini taqqoslash zarur ekan, demak, har biri vazifaga mos keladigan ko'p hujjatli interfeysni ishlab chiqish maqsadga muvofiqdir. Shuningdek, alohida hujjat sifatida ma'lumotlar ko'rib chiqilishi kerak. Ular turli algoritmlar bilan qayta ishlanishi yoki boshqa topshiriqlarda foydalanish uchun tuzatilishi mumkin.

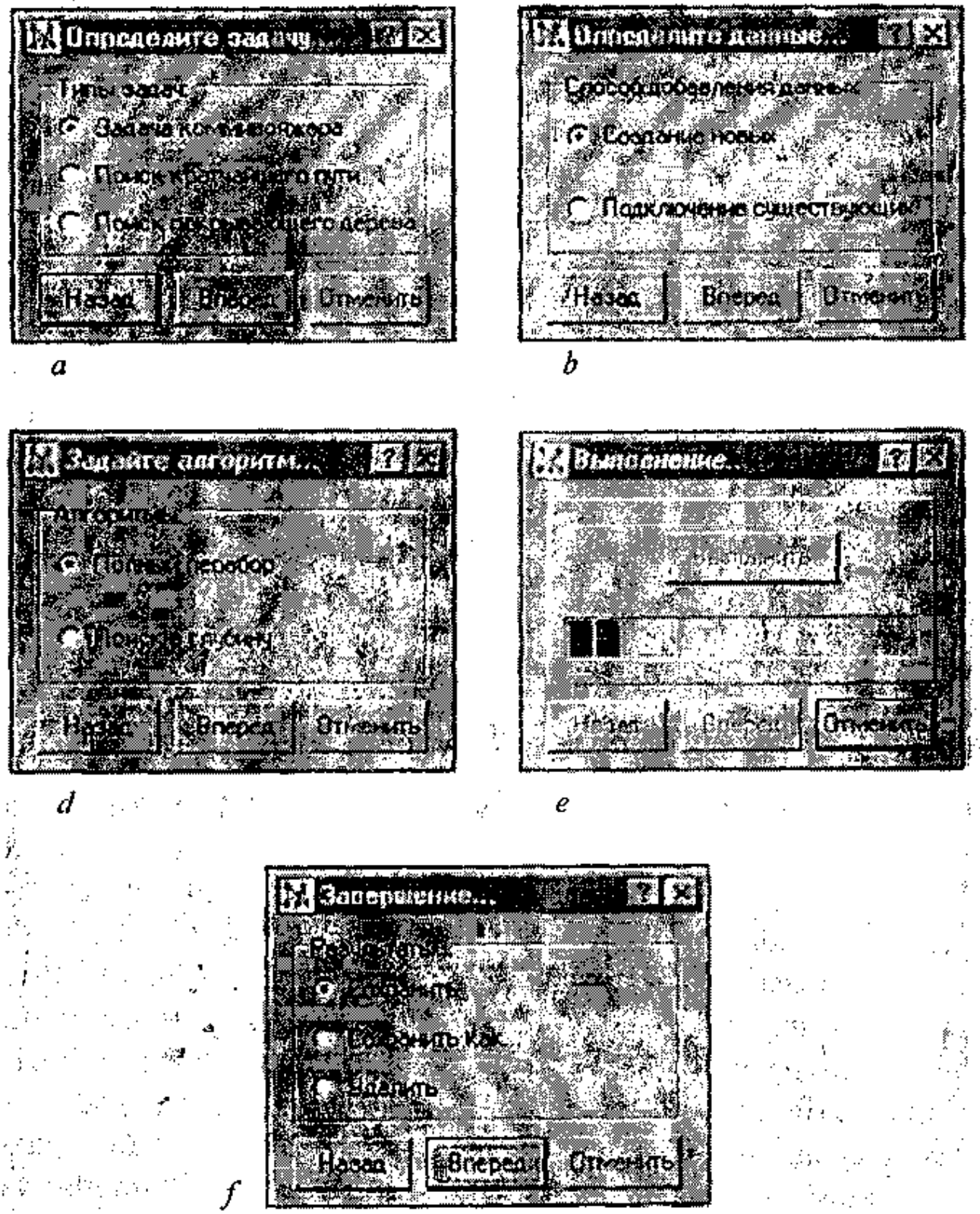
8.8-rasmda tizimning abstrakt dialogi grafikasi berilgan bo'lib, uning asosida menyu tuzilishi lozim. Ko'rib chiqilgan grafik dialogi operatsiyalariga topshiriq fayl xizmat ko'rsatish operatsiyalari, sho'ba oynalarini boshqarish va ma'lumotnoma axborotlari bilan ishlashni qo'shish so'ngra olingan ko'plab operatsiyalarni guruhlarga bo'lish lozim.

Birinchi oyna masala turiga o'xshash tanlovni bajaradi (8.14, a-rasm). Tanlash natijasi maxsus hujjat – Bayonnomada qayd etiladi. Ikkinchi oyna – ma'lumotlarni berish usulini belgilash (8.14, b-rasm), uchinchisi – tanlangan usulga bog'liq holda ma'lumotlarning bevosita vazifasi (masalan, rasmda yo'q, chunki ma'lumotlarni aniqlash shakli topshiriqqa bog'liq va natijalarni chiqarish shakli bilan birga har bir vazifa uchun alohida loyihalashtirish maqsadga muvofiq). To'rtinchi – algoritmni tanlash (8.14, d-rasm). Beshinchi – bajarishni initsiallash (8.14, e-rasm).

Oltinchi – natijalarni namoyish etadi (rasmda yo'q). Yettinchi – natijalardagi kabi amalga oshiriladi (8.14, f-rasm). Barcha dialoglar ancha o'xshash sxema bo'yicha tuziladi va bu foydalanuvchi ishini yengillashtiradi.




**8.13-rasm.** Yangi topshiriqning yangidan tuzilgan dialogi holati diagrammasi.

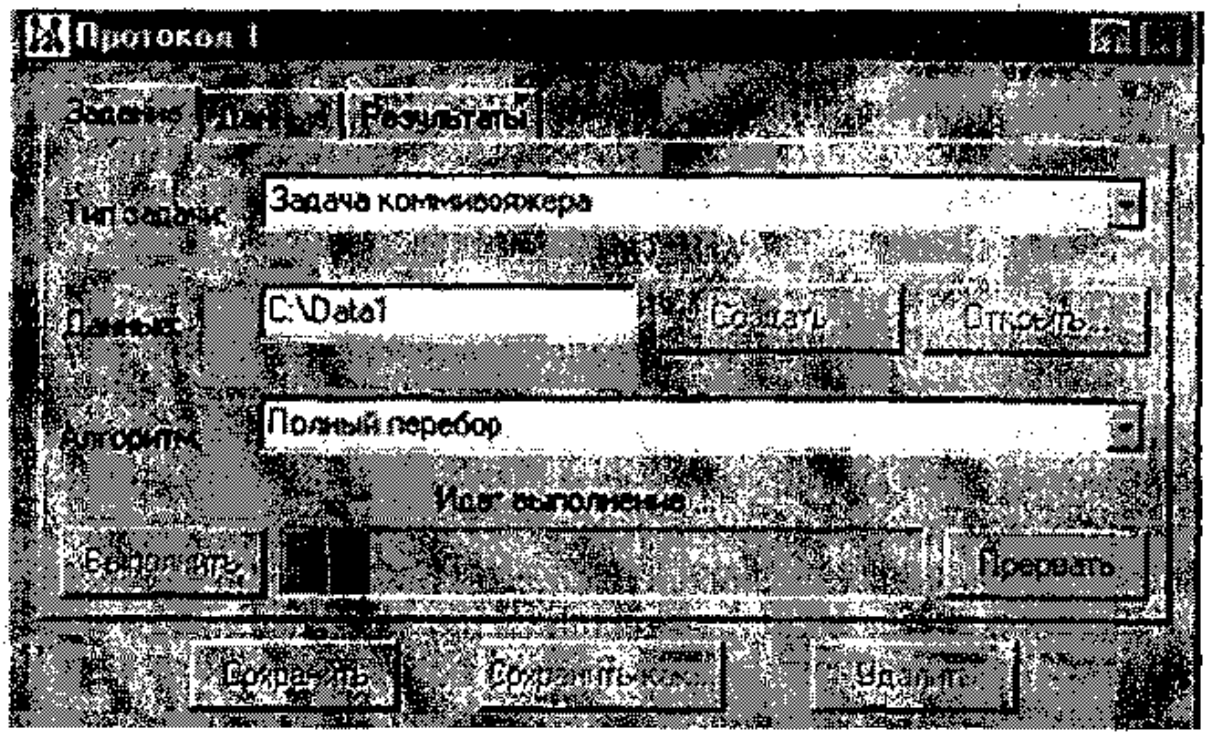



*a* *b*

*d* *e*

*f*

**8.14-rasm.** Yangi topshiriq dialogining asosiy shakllari: a – masala turini aniqlash dialogli oynasi; b – ma'lumotlarni berish usullarini aniqlash dialogli oynasi; d – algoritmni berish dialogli oynasi; e – bajarishni initsiallash dialogli oynasi; f – topshiriqni tugallashning dialogli oynasi.




**8.15-rasm.** Yangi topshiriq dialogini amalga oshirish. Ikkilanish mavjud bo'lganda bir necha variantlarni taklif etish mumkin. Interfeys ma'qullangach, tegishli protseduralar kodlanib amalga oshiriladi.

Ko'rib chiqilgan har ikki variant ma'lum kamchiliklarga ega. Jumladan, shakl ko'rinishida amalga oshirish jarayonlarni boshqaruvchi juda ko'p tugmachalarga ega. Dialoglarni ketmaket ko'rinishda amalga oshirish esa haddan ziyod ko'p qadamli bosqichga ega bo'lgani uchun alohida shakl ko'rinishida taqdim etiladigan ma'lumotar hamda natijalarga kirishni qiyinlashtiradi.

Bunda navigatsiyani amalga oshirishni yaxshilaydigan variantni ko'rib chiqamiz. Bunda quyidagi qator bayonnomaga tegishli ekanligini aniq belgilagan holda tugmachalardan vizual foydalanish imkonini beradi. Bunday interfeysda ma'lumotlar va natijalarni ko'rib chiqishning o'zi yetarli. Buning uchun navbatdagi sahifaga o'tiladi (8.15-rasm).

Dialoglarni amalga oshirish natijasida olingan interfeys to'liqligi tekshiriladi, so'ngra qo'llashning qulayligini testdan o'tkazish uchun foydalanuvchiga taklif etiladi.

## 8.7. Bevosita manipulyatsiyalashni foydalanuvchi interfeyslari va ularni loyihalashtirish

WIMP interfeyslarida ko'rib chiqilgan bevosita manipulyatsiya qilish imkoniyati obyektli-mo'ljallangan interfeyslar ishlab chiqish imkonini beradi.

Mazkur turdagi interfeyslar tashqi darajada dialogning direktiv shaklidan foydalanadi: buyruqlarni kiritish «sichqon» obyekti piktogrammasi bilan ma'lum amallarni bajarishda amalga oshiriladi. Ushbu interfeyslarning asosiy elementlari: metaforalar, obyektlar hamda obyektlar va Drag and Drop («tashidi va tashladi») texnologiyasini taqdim etishdan iborat.

**Metaforalar.** *Metaforalar* – bir obyektning o'xshash xususiyatlari va belgilarini fikran boshqa obyektga ko'chirish. Interfeyslarda metaforalarni qo'llash foydalanuvchida mavjud tajribani faollashtirishni nazarda tutadi.

Bevosita manipulyatsiya interfeysi foydalanuvchiga ilgari u ish faoliyatida yoki turmushda bir necha bor duch kelgan tanish elementlardan iborat muhitni ta'minlashi, alohida obyektlar bilan manipulyatsiya qilish imkonini berishi lozim. Metaforalar mavjudligi foydalanuvchi uchun interfeysni o'zlashtirish jarayonini soddalashtiradi. Masalan, fayllarni olib tashlash uchun Windows foydalanadigan «Chiqindini tashlab yuborish» («Выбрасывание мусора») metaforasi foydalanuvchiga ushbu operatsiyani o'zlashtirishga ko'maklashadi.

Metaforadan juda ehtiyotkorona foydalanish lozim, chunki bunda mazmun interfeysining barcha elementlariga beriladi. Masalan, o'xshash elementlar o'zini o'xshash holatda tutishi, bitta rangda berilgan elementlar esa bir-biri bilan ma'lum aloqada bo'lishi kerak. Interfeys elementlari o'rtasidagi semantik nomuvofiqlik foydalanuvchining g'ashiga tegadi va noto'g'ri mo'ljal hosil qiladi.

Shuni ham hisobga olish lozimki, to'liq muvofiqlik ham foydalanuvchi kutgan narsani bermasligi mumkin. Chunki, u mavjud narsalar emas, ularning modellari bilan ishlaydi. Demak,

uning imkoniyatlari chegaralangan. Shuning uchun tasvirni haddan ziyod real olish maqsadga muvofiq emas.

**Metaforalar va animatsiya.** Metaforalarni amalga oshirishda multimedia vositalari, asosan animatsiya muhim rol o'ynaydi. 8.2-mavzuda qayd etiladiki, harakat e'tiborni tortadi, kadrlarning keskin almashishi, oldingi va keyingi kadrlar aloqasini anglash, yangisini o'rganish uchun ma'lum vaqt talab qiladi. Demak, multiplikatsiyadan foydalanib, foydalanuvchiga qaysidir ma'noda dam berish emas, shuningdek, o'zgargan holatga o'rganish vaqtini qisqartirgan holda uni kadrlar almashuviga tayyorlaydi ham. Masalan, uzundan-uzun ro'yxatni cheki yo'q devor shaklida tasvirlash mumkin (istiqbol qonuni bo'yicha). Shunda ushbu devor bo'ylab «tabiiy» yirik nom bilan tasvirlangan «harakat» manzarani keskin o'zgartirmasdan kerakli axborotni izlash imkonini beradi. Bunda *inson ongida obyektlar aynanligi saqlanib qoladi,* chunki u obyektlar bilan doimiy o'zaro aloqada bo'lishga tayyor.

Biroq, insonning qabul qilish xususiyatlari bilan bog'liq muammolarni hal eta turib, interfeyslar dasturchilar va ishlanma mualliflari uchun qo'shimcha muammo tug'diradi. Dasturning odatdagi funksiyaviy va interfeys darajasiga shuningdek, vizual daraja ham qo'shiladi. Bunday interfeysni amalga oshiruvchi dastur hech mahal to'xtab turmaydi. Chunki foydalanuvchi buyruqlarni kiritishni kutish paytida u tegishli kadrlarni aks ettirib turadi. Bunday dastur asosida vaqtinchalik dasturlash yotadi. Voqeaviy dasturlashdan farqli ravishda *vaqtinchalik dasturlar* modellashtiriladigan jarayon holati va foydalanuvchi harakatiga bog'liq holda kadrlar uzviyligini o'zgartirish imkonini beradi.

**Bevosita manipulyatsiyalash interfeys obyektlari va ularni taqdim etish.** To'g'ridan to'g'ri manipulyatsiyalash interfeyslarining uchta asosiy turi mavjud: ma'lumotlar-obyektlar, konteynerlar-obyektlar va qurilmalar-obyektlar.

*Ma'lumotlar-obyektlar* foydalanuvchini axborot bilan ta'minlaydi. Bular matn, tasvir, elektron jadvallar, musiqa, video, shuningdek, ularning har xil kombinatsiyalari bo'lishi mumkin. Ope-

ratsion tizimi doirasida bunday obyektlarga obyektlarni ochishda hosil bo'ladigan ilovalar muvofiq keladi. Ilovalar o'lchamida obyektga bitta yoki bir nechta shakllar mos keladi. Ularda mavjud obyektlar turli ko'rinishlarda namoyon bo'ladi.

*Konteynerlar-obyektlar* o'zining ichki obyektlari, jumladan, boshqa konteynerlar bilan manipulyatsiya qilish mumkin. Masalan, ularni nusxalaydi yoki har qanday tartibda turlarga ajratadi. Tipik konteynerlarga sumkalar, savatlar va h.z.lar kiradi. Konteynerni ochishda ular saqlaydigan komponentlar namoyon bo'ladi va ular bilan manipulyatsiya qilish imkoni yuzaga keladi. Bunda komponentlar piktogrammalar bilan belgilanishi yoki jadval ko'rinishida taqdim etilishi mumkin.

*Qurilmalar-obyektlar* real dunyoda mavjud qurilmalarni taqdim etadi. Telefon, faks, printerlar shular jumlasidan. Bunday obyektlar ochilganda, odatda uni sozlash nuqtasini ko'rish mumkin.

Shunday qilib, har bir obyektga kamida bitta oyna mos keladi. Boshlang'ich holatda bu oyna piktogramma bilan taqdim etiladi, ammo zarur hollarda uni ochish va talab etiladigan operatsiyani bajarish mumkin. Obyekt oynasi ochiq holatda menyu va instrumentlar paneliga ega bo'lishi mumkin. Piktogramma esa obyekt ustidan o'tkaziladigan operatsiyalar ro'yxatiga ega matn menyusiga muvofiq kelishi kerak.

Piktogramma nomi obyektlarning har bir turi uchun shakllantiriladi. Masalan, ma'lumotlar-obyektlar piktogrammasi saqlanadigan ma'lumotlar nomiga mos keluvchi nom beradi, ma'lumotlar turi esa piktogrammaning o'zi bilan kodlanadi. Piktogramma-konteyner yoki qurilma piktogrammasi nomi obyektning o'zini bildiradi, shu bois mundarijaga bog'liq emas.

Shuni nazarda tutish kerakki, obyektlar turlari o'rtasidagi farq shartli sanaladi. Chunki, bitta va o'sha obyekt turli sharoitda goh ma'lumot-obyekt, goh qurilma-obyekt, goh konteyner-obyekt kabi namoyon bo'lishi mumkin. Masalan, printer odatda qurilma-obyekt sifatida ko'rinishi mumkin, ammo, konteyner-obyekt xususiyatiga ham ega bo'lishi mumkin.

**Drag and Drap texnologiyasi.** Drag and Drap texnologiyasi (tashidi va tashladi) bevosita manipulyatsiya qilishning asosiy tamoyillarini belgilaydi. U IBM firmasining foydalanuvchi interfeysini ishlab chiqish bo'yicha qo'llanmada qo'shimcha bayon etilgan:

- obyektning ko'chish natijasi foydalanuvchi kutgan natijaga mos kelishi lozim;
- foydalanuvchilar kutilmaganda axborotni yo'qotib qo'ymasligi kerak;
- foydalanuvchi noto'g'ri amallarni bekor qilish imkoniga ega bo'lishi zarur.

Bu texnologiya shuningdek, bevosita manipulyatsiya operatsiyasini vizualizatsiya qilishning asosiy tamoyillarini belgilaydi:

- boshlang'ich ajratish – foydalanuvchiga «obyekt tutilgan»ligini xabar qilish uchun qayta aloqa sifatida foydalaniladi. Windows da shu maqsadda ranglar bilan ajratish qo'llaniladi;
- ko'chish vizualizatsiyasi – bajariladigan amallarni identifikatsiyalash uchun foydalaniladi;
- maqsadli ajratish – belgilangan punktni identifikatsiyalash uchun foydalaniladi;
- harakatni vizualizatsiyalash – operatsiyani tugallashni kutish vaqtini belgilash uchun foydalaniladi.

Odatda shu maqsadda animatsiya yoki «qum soat»dagi kursor shaklini o'zgartirish qo'llaniladi.

Shuni nazarda tutish kerakki, belgilangan punktning ikkita turi mavjud. Ularning biri obyektni, ikkinchisi uning nusxasini qabul qiladi. Masalan, agar foydalanuvchi hujjatni «savat»ga «tashlasa», ekrandan olib tashlangan hujjat yo'q qilinadi. Agar u o'sha hujjatni printerga «tashlasa», hujjat butunlay yo'qolmaydi. Printerga hujjat nusxasi uzatiladi.

Windows da hujjat piktogrammasi oldida nusxalash nusxasi «+» paydo bo'lishi bilan tegishli harakat identifikatsiyalanadi. Agar ayrim qurilmalar uchun harakatning har ikki turini amalga oshirish imkoni bo'lsa, uning tegishlisini aniqlash imkoniyatini nazarda tutish lozim. Jumladan, Windows bunday holatda

CTRL klavishini bosish orqali nusxalashni va aksincha holatda ko'chishni amalga oshiriladi.

**Bevosita manipulyatsiyalash interfeysini loyihalashtirish.** Bunday loyihalash ma'lum dasturiy ta'minot uchun ishlab chiqilgan dialog grafasi asosida amalga oshiriladi va quyidagi protseduralarni o'z ichiga oladi:

- ekranda ko'rsatilishi kerak bo'lgan *predmet sohasining ko'plab obyektlarini* shakllantirish. Bunday holatda asos sifatida foydalanish variantlari emas, predmet sohasining konseptual modeli qo'llaniladi;
- *obyektlar* tahlili, ularning *turlari va ko'rinishlarini.* Shuningdek, ushbu obyektlar bilan o'tkaziladigan operatsiyalar ro'yxatini belgilash;
- *obyektlarning o'zaro ta'sirini* aniqlash va bevosita manipulyatsiya matritsasini tuzish;
- obyektlarning *vizual ko'rinishini* aniqlash;
- obyektlar oynasi menyusi va matn menyusini ishlab chiqish;
- interfeys prototipini yaratish;
- foydalanish qulayligini testdan o'tkazish.

**8.6-misol.** Kombinatorli-optimallashgan topshiriqlarni bajarish tizimi uchun bevosita manipulyatsiyalashning foydalanuvchi interfeysini ishlab chiqish.

Chunki, yagona so'raladigan analogiya bu operatsiyalarni qo'lda bajarish va «ishchi stol» metaforasidan foydalangan holda interfeysni quramiz.

Predmet sohasining konseptual modelini tahlil etgan holda ko'plab nomzod-obyektlarni shakllantiramiz.

Har bir obyekt uchun ularni predmet sohasining boshqa obyektlarini bog'lovchi operatsiyalar turi va to'plamini belgilaymiz (8.3-jadval).

Loyihalashtirilayotgan tizimning asosiy obyekti — Bayonnoma. U vazifa, Ma'lumotlar va natijalarni birlashtiradi. Ish jarayonida foydalanuvchiga yangi bayonnomalarni tuzishga to'g'ri keladi. Bunda yangi bayonnomaning vazifa blankiga ega bo'lishi

maqsadga muvofiq. Ma'lumotlarni alohida yaratishga qaror qilamiz. Ma'lumotlar blankini to'ldirgach, Bayonnomaga kiritish, Ma'lumotlar ro'yxatida saqlash, chop etish yoki tashlab yuborish mumkin. Topshiriq bajarilgach, ya'ni vazifa hal etilgach natijalar Bayonnomaga kiritiladi. Bayonnomani to'ldirish, chop etish va uni «ishchi stol»dan olib tashlashning har qanday bosqichda uni Bayonnomalar ro'yxatida saqlash imkoniyatini ko'rib chiqamiz.

Yuqorida qayd etilganlardan tashqari tizimning ko'plab obyektlariga quyidagi qurilma-obyektlar kiritilgan: Kompyuter, Printer, «Savat». Bu vositalar «Bajarish», «Chop etish», «Olib tashlash» operatsiyalarini aks ettirish uchun kerak.

Shundan so'ng obyektlarning o'zaro ta'sir matritsasini (jadvalini) tuzamiz. Mazkur jadvalda vertikal bo'ylab metaforaga ko'ra ko'chirish mumkin bo'lgan (boshlang'ich) obyektlarni, gorizontal ravishda esa ko'chishning oxirgi punkti bo'lishi mumkin bo'lgan obyektlarni joylashtiramiz. Jadvalning o'zida boshlang'ich obyektdan so'nggisiga ko'chirish operatsiyasiga mos keladigan harakatlarni qayd etamiz.

Ko'rib chiqilayotgan misol uchun Bayonnoma, Bayonnoma ro'yxati, Vazifa, Ma'lumotlar, Bayonnomalar ro'yxati, Vazifa va ma'lumotlar blanklari dastlabki obyektlar sanaladi (8.4-jadval). Ularni konteyner-obyektlar: Bayonnoma, Bayonnomalar ro'yxati, Ma'lumotlar ro'yxati, shuningdek, qurilmalar obyektlari: Kompyuter, Printer, «Savat» va Ishchi stolga ko'chirish mumkin. So'ngra qabul qilingan metaforaga muvofiq obyektlarning o'zaro ta'sirini tahlil etamiz.

Agar obyekt Bayonnoma Bayonnomalar ro'yxatiga o'tkazilsa, mohiyatan buni foydalanuvchining Bayonnomani ro'yxatga qo'shish istagi sifatida talqin etish mumkin. Butun jadvalni o'xshash tartibda to'ldiramiz, so'ngra tegishli voqeani dasturlashda foydalaniladi.

*8.3-jadval*

| Obyekt | Tip | Boshqa obyektlar bilan bog'liq harakatlar | Izoh |
|---|---|---|---|
| Bayonnoma | Konteyner | Saqlash, olib tashlash, chop etish | Vazifalar, Ma'lumotlarni, topshiriq bajarilgach – Natijalarni kiritadi |
| Bayonnomalar ro'yxati | Konteyner | Tozalash | – |
| Vazifa | Ma'lumotlar | Bajarish | Vazifa blanki bayonnoma blankiga kiradi |
| Ma'lumotlar | Ma'lumotlar | Bayonnoma bilan bog'lash, saqlash, olib tashlash, chop etish | – |
| Ma'lumotlar ro'yxati | Konteyner | Tozalash | – |
| Natijalar | Ma'lumotlar | – | Agar olingan bo'lsa, Vazifa va Ma'lumotlar bilan bog'liq, ya'ni bayonnomada bo'ladi |
| Bayonnoma blanki | Ma'lumotlar | – | Bayonnoma blanki vazifa blankiga ega |
| Ma'lumotlar blanki | Ma'lumotlar | – | – |
| Kompyuter | Qurilma | – | – |
| Printer | Qurilma | – | – |
| Savat | Qurilma | – | – |

*8.4-jadval*

| Boshlang'ich obyektlar | Obyektlarni ko'chirishda bajariladigan operatsiyalar — Oxirgi obyektlar | | | | | | |
|---|---|---|---|---|---|---|---|
| | Bayonnoma | Bayonnomalar ro'yxati | Ma'lumotlar ro'yxati | Kompyuter | Printer | Savat | Ishchi stol |
| Bayonnoma | – | Ro'yxatga bayonnomani qo'shish | – | Vazipredmeti bajarish | Bayonnomani chop etish | Bayonnomani olib tashlash | – |
| Bayonnomalar ro'yxati | – | – | – | – | – | Barcha bayonnomani olib tashlash | – |
| Vazifa | Vazipredmetni bayonnomaga kiritish | – | – | – | – | Vazipredmetni olib tashlash | – |
| Ma'lumotlar | Ma'lumotlarni ro'yxatga kiritish | – | Ma'lumotlarni ro'yxatga qo'shish | – | – | Ma'lumotlarni olib tashlash | – |
| Ma'lumotlar ro'yxati | – | – | – | – | – | Barcha ma'lumotlarni olib tashlash | – |
| Bayonnoma blanki | – | – | – | – | – | – | Bayonnoma tuzish |
| Ma'lumotlar blanki | – | – | – | – | – | – | Ma'lumotlar tuzish |

* Operatsiya obyekt nusxasi bilan amalga oshiriladi, obyektning o'zi joyida qoladi.

**8.7-misol.** *Bayonnoma obyekti oynasi ko'rinishini ishlab chiqish.*

Bayonnoma obyekti konteyner bo'lib, o'zida vazifa, Ma'lumotlar va Natijalarni saqlash mumkin. Bayonnomani tuzish vaqtida Vazipredmeting toza blanki avtomatik ravishda yaratiladi.

Konteyner – obyektning standart ko'rinishi berilgan. Bu oyna turli vaqtlarda qanday ko'rinishga ega bo'lishi alohida tasvirlangan. Oyna o'z menyusidan quyidagi bandlarni o'z ichiga oladi: Ma'lumotlar, Tuzatish, Ko'rinish va Yordam. Birinchi band «Tuzatish» va «Ochish» – ma'lumotlarni aniqlash imkonini beradi. Ikkinchisi umumiy bufer ishiga javob beradi va «Qirqish», «Nusxalash» va «Qo'yish» bandlarini o'z ichiga oladi. Uchinchi band an'anaviy ravishda «Yirik belgilar», «Mayda belgilar», Ro'yxat va Jadval oyna turlarini boshqaradi. To'rtinchi band – *ma'lumotnoma so'rash uchun foydalaniladi.*

Ishchi stolning foydalaniladigan metaforasi Bayonnoma obyektiga ega standart klaviatura buyruqlarini bajarish imkonini nazarda tutish imkonini beradi. Bunga qirqish (Ctrl-X), Nusxalash (Ctrl-C), Qo'yish (Ctrl-V) operatsiyalari kiradi.

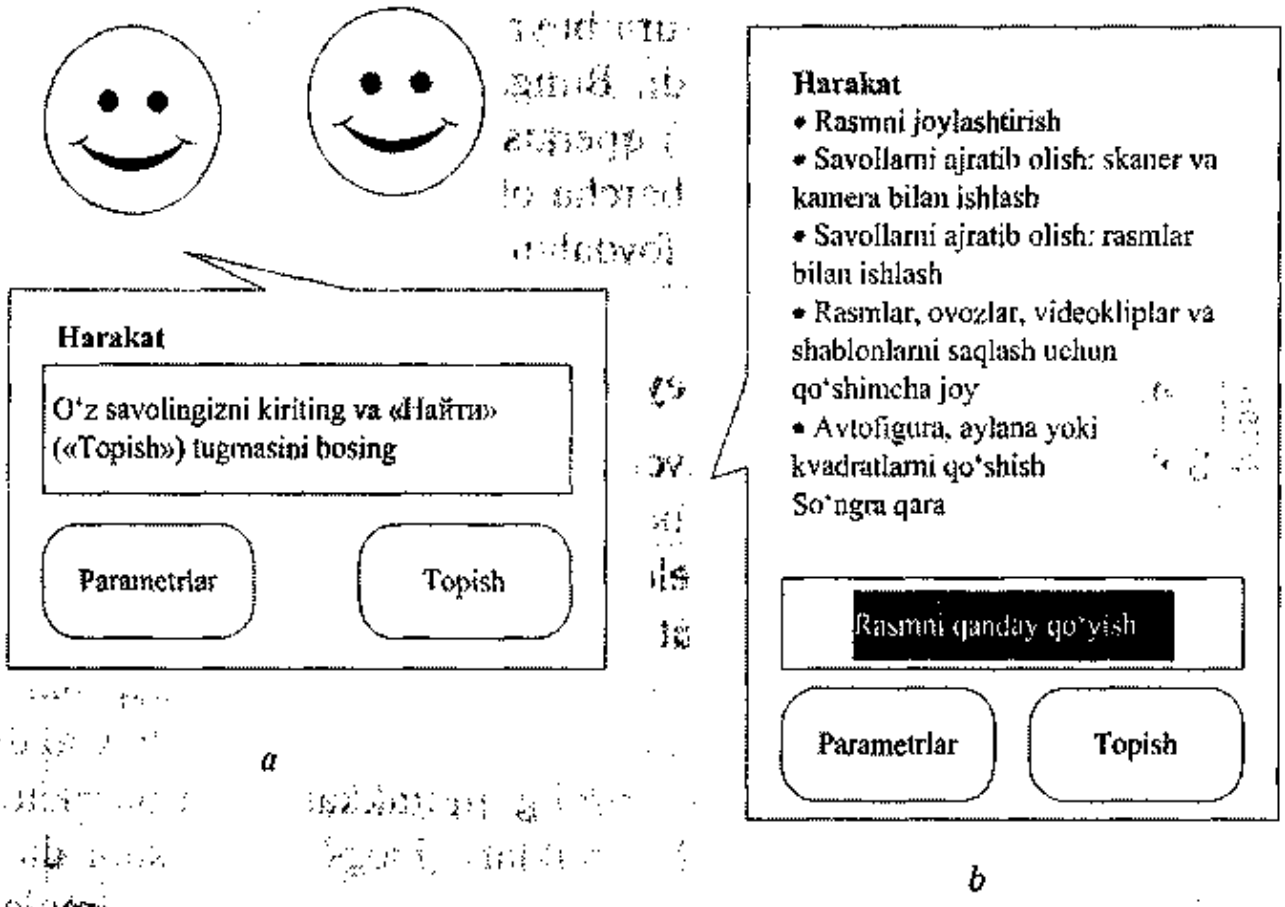
Loyihalashtirish tugagach barcha obyektlar ko'rinishini interfeys prototipi yaratadi va uni foydalanish qulayligi bo'yicha testdan o'tkazishga topshiradi.

## 8.8. Foydalanuvchi interfeysning intellektual elementlari

So'nggi yillarda foydalanuvchi interfeyslarining ko'plab yangi istiqbolli elementlari paydo bo'ladi. Bu, asosan, interfeyslarga kiritiluvchi sun'iy intellekt elementlaridir. Usta, Maslahatchi, Agent kabi nomlardan elementlar shular jumlasidan. Ko'p marta ijtimoiylashtirilgan foydalanuvchi interfeysini yaratishga harakat qilingan, ya'ni «shaxsga ega» interfeyslarni yaratish yotadi. Turli sharoitlarda uy hayvonlarining murakkab xatti-harakatini amalga oshiruvchi Cats (Mushuklar), Dogs (Itlar) kabi dam olish dasturlari shuni ko'rsatadiki, texnik jihatdan bu amalga oshirilishi mumkin bo'lgan vazifa. Biroq, bu sohada psixomantiqiy muammolar mavjud. Misol sifatida, «yuvvosh» Maslahat-

chi – Microsoft Office ni eslab ko'raylik. U ko'plab foydalanuvchilarda salbiy taassurot uyg'otadi. Hozircha bunday «shaxs»ni yaratish yaxshi natija bergani yo'q.

**Maslahatchilar.** Maslahatchilar odatda ko'rsatma shaklida namoyon bo'ladi. Ularni ma'lumotnoma menyusi yoki «suzib chiqadigan» menyu yordamida chaqirish mumkin. Maslahatchilar foydalanuvchilarga aniq topshiriqlarni bajarishda yordam beradi. Faqat buning uchun foydalanuvchi unga nima qilishni ko'rsatishi lozim. Masalan, Microsoft Word da ishlaydigan foydalanuvchi hujjatga birorta rasmni joylashtirmoqchi, ammo buni qanday amalga oshirishni bilmaydi. U «Yordamchi oynasini faollashtiradi va tegishli savolni maxsus joyga kiritadi (8.16, a-rasm). Ma'lumotnoma tizimi savolni tahlil qiladi va foydalanuvchini qiziqtiruvchi ro'yxatni shakllantiradi. Foydalanuvchi esa kerakli ma'lumotni o'zi oladi.




**8.16-rasm.** Microsoft Word 2000 da Yordamchi oynasi: a – so'rov; b – o'zaro bog'liq mavzular ro'yxati.

**Ustalar.** Dastur-usta keng tarqalgan, ammo dastur yoki jihozni o'rnatish kabi ayrim foydalanuvchilar tomonidan bajariladigan vazifalarni amalga oshirish uchun foydalaniladi. Bunday amallarni bajarish foydalanuvchidan o'zaro bog'langan murakkab qarorlar qabul qilishni talab qiladi. Uning ketma-ketligini dastur usta ko'rsatib turadi.

Intellektual Ustalar har bir bosqichda namoyish oynasida foydalanuvchining o'tgan savollarga javoblarining natijalarini namoyish etishi mumkin.

Ustalar dialoglarning uzviy yoki daraxtsimon ssenariysini amalga oshiradi. Shuning uchun yaxshi tuzilmaga ega, ketma-ket masalalarni hal etish uchun foydalanish maqsadga muvofiq. Buning uchun quyidagilarni amalga oshirish lozim:

- foydalanuvchiga o'tgan bosqichga qaytish imkonini berish;
- Usta ishini bekor qilish imkoniyatini nazarda tutish;
- bosqichlarni raqamlash va agar bunday bosqichlar uchtadan oshib ketsa, foydalanuvchiga Ustaning ish bosqichlari soni haqida xabar qilish;
- foydalanuvchiga har bir qadam (bosqichni) tushuntirish;
- imkoniyatga ko'ra har bir bosqichda amalga oshirilgan operatsiya natijalarini imkon qadar namoyish etish.

**Nazorat savollari:**

1. Interfeysning asosiy turlarini aytib bering. Ularning har biri qanday xususiyatlarga ega. Ular qanday vositalar bilan amalga oshiriladi? Hozirda interfeyslarning qanday turlari asosiy sanaladi?
2. Interfeyslarni loyihalashtirishda hisobga olinishi kerak bo'lgan insonning ruhiy-jismoniy xususiyatlarini sanab bering.
3. «Dialog» atamasi deganda nima tushuniladi? Dasturiy ta'minot nechta dialogni amalga oshirishi mumkin?
4. Dialogning asosiy turlari va shakllarini aytib bering. Dialoglarni loyihalashtirish uchun dastlabki ma'lumotlar sifatida qanday vositalar xizmat qiladi? Dialoglarni bayon etish uchun qanday modellardan foydalaniladi?

5. Sodda grafik muharrir uchun dialog grafasini tuzing. Nima uchun u shunday ko'rinishga ega? Qanday sharoitlarda dialog grafasi zanjir yoki daraxtsimon ko'rinishga ega bo'ladi?

6. Grafik muharrir menyusini taklif eting. Ushbu menyuni sizga tanish grafik muharrir menyusi bilan qiyoslang. Farqlarni qiyoslang.

7. Grafik foydalanuvchi interfeysining asosiy komponentlarini sanab bering. Ularning har biri qanday holatlarda qo'llaniladi?

8. Grafik muharrir uchun bevosita manipulyatsiya interfeysini taklif eting. Bunday interfeyslarni loyihalashtirishning asosiy murakkabligi nimada? Ulardan qanday holatlarda foydalanish maqsadga muvofiq?

9. Hozirda foydalanuvchi interfeysining qanday intellektual komponentlari mavjud? Ularning asosiy ahamiyati nimadan iborat? Qanday holatlarda ulardan foydalanish maqsadga muvofiq?

# 9. DASTURIY MAHSULOTLARNI TESTDAN O‘TKAZISH

Testdan o‘tkazish – dasturiy ta’minotni ishlab chiqish jarayonining juda muhim va ko‘p mehnat talab qiladigan bosqichi. Chunki, testdan to‘g‘ri o‘tkazish dasturni tuzishda yo‘l qo‘yilgan ko‘plab xatolarni aniqlash imkonini beradi.

Dasturiy ta’minotni ishlab chiqish jarayonida testdan o‘tkazishning quyidagi uchta bosqichi nazarda tutiladi: avtonom, kompleks va tizimli. Ularning har biri tizimning tegishli qismini tugallashiga muvofiq keladi.

Testni tuzishga yondashuvning ikki turi farqlanadi: tuzilmaviy va funksiyaviy. Mazkur yondashuvlarning har biri o‘z xususiyatlari va qo‘llash sohalariga ega.

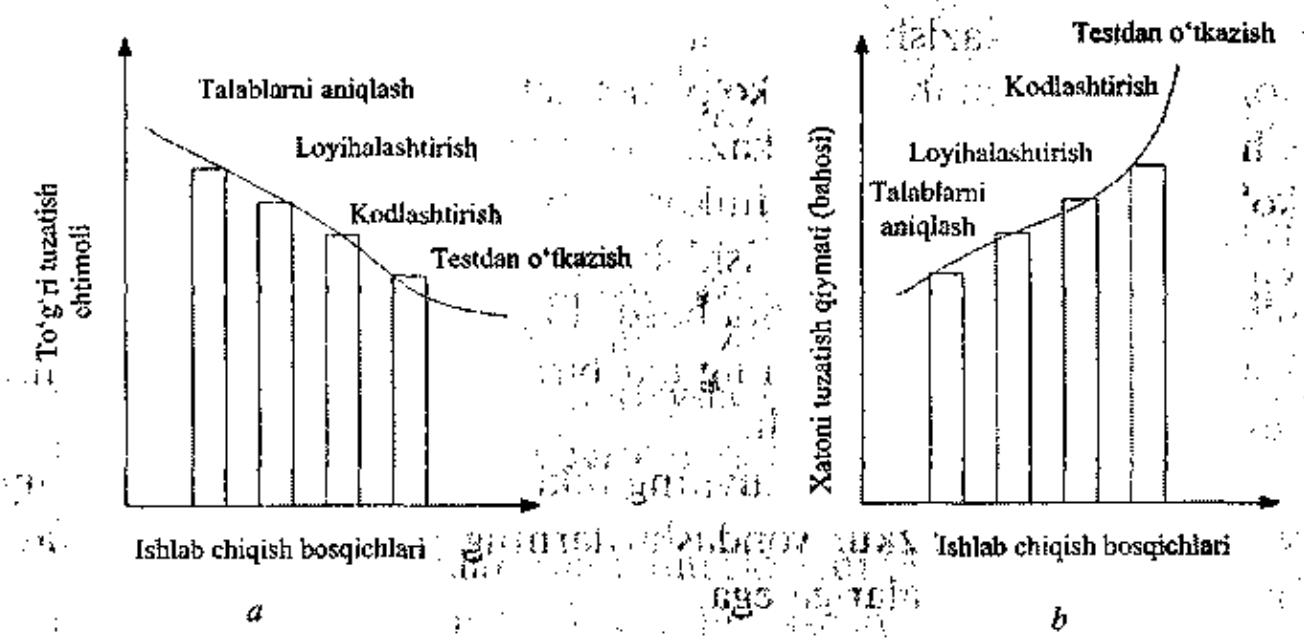
## 9.1. Ishlab chiqilayotgan dasturiy ta’minot sifatini nazorat qilish turlari

Dasturiy ta’minotni loyihalashtirish va kodlashtirishning o‘zi yetarli emas, shuningdek, talablar va tasniflarga mos kelishini ham ta’minlash lozim. Ko‘p martalik tadqiqot shuni ko‘rsatadiki, u yoki bu nomuvofiqlikda xato qanchalik erta aniqlansa, ularni shunchalik to‘g‘ri tuzatish mumkin (9.1, a-rasm).

Dasturiy ta’minotni ishlab chiqish bo‘yicha zamonaviy texnologiya ishlanmaning barcha bosqichlarida natijalarni nazorat qilish evaziga xatolarni ertaroq topishni nazarda tutadi. Boshlang‘ich bosqichlarda bunday nazorat, asosan, qo‘lda yoki CASE vositasidan (test) foydalangan holda amalga oshiriladi.

**Testdan o‘tkazish** bu dasturni bajarish jarayoni bo‘lib, asosiy maqsad xatolarni aniqlashdan iborat. Hech qanday testdan o‘tkazish jarayoni biror murakkab dasturiy ta’minotda xato yo‘qligini isbotlab bera olmaydi. Bunday dasturiy ta’minot uchun to‘liq testdan o‘tkazishni amalga oshirish imkoni bo‘lmaydi. Demak, har doim dasturiy ta’minotda aniqlanmagan xatolar mavjudligi ehtimoli saqlanib qoladi. Biroq, testdan o‘tkazishning

asosiy qoidalariga rioya etish va testlarning ilmiy asoslangan tanlovi xatolar sonini kamaytirishi mumkin.




**9.1-rasm.** To'g'ri tuzatish ehtimoli (a) va uning qiymatining ishlab chiqish bosqichlariga bog'liqligi.

**Eslatma.** Odatda testdan o'tkazish to'g'risidagi savolga boshlovchi dasturchilar «bundan maqsad dasturning to'g'riligini isbotlash» deb javob beradi. Bu butunlay noto'g'ri fikr. G.Mayers ushbu qoidani aniqlash uchun juda qiziqarli fikrni taklif etadi. Tasavvur qilingki, Siz shifokorga biqiningiz og'riyotganidan shikoyat qildingiz. Shifokor sizni eshitib, tekshiruvga yo'lladi. Ma'lum vaqtdan so'ng tekshiruv xulosasi va analiz natijalari bilan yana shifokor qabuliga keldingiz. Bu xulosalarda barcha tekshiruv natijalari me'yorida ekanligi ta'kidlangan. Ammo baribir biqiningiz og'riyapti, demak, nimadir joyida emas, analiz xulosalarida bu ko'rsatilmagan. Xuddi shunday barcha test tanlovlarida xatosiz ishlaydigan murakkab dasturiy ta'minot ham ayrim xatoliklarga yo'l qo'yishi mumkin.

Dasturiy ta'minotni ishlab chiqish jarayoni testdan o'tkazishning uch bosqichini nazarda tutadi:

- dasturiy ta'minot komponentlarini avtonom testdan o'tkazish;

• ishlab chiqilayotgan dasturiy ta'minotni kompleks testdan o'tkazish;

• asosiy sifat mezonlariga muvofiq tizimli yoki baholi testdan o'tkazish.

Testdan o'tkazish sifatini oshirish uchun quyidagi asosiy tamoyillarga rioya etish talab etiladi:

• taxminiy natijalar testgacha ma'lum bo'lishi kerak;

• muallif bilan dasturni testdan o'tkazishdan qochish lozim;

• har bir test natijalarini batafsil o'rganish zarur;

• noto'g'ri ma'lumotlar tufayli, kutilmagan salbiy ta'sir keltirib chiqarishiga qarshi dasturni tekshirish lozim.

Shuni nazarda tutish kerakki, dasturda topilmagan xatolar soni topilgan xatolar soniga teng bo'lishi ehtimolini nazarda tutish lozim.

**Test to'plamlarini tuzish.** Testdan o'tkazishni belgilashga muvofiq mazkur mavzu boshida hech bo'lmasa, bitta xatoni topgan testni muvaffaqiyatli deb hisoblash mumkin. Shu nuqtayi nazardan, har biri imkon qadar yuqori ehtimolda xato topishi mumkin bo'lgan test to'plamlaridan foydalanish maqsadga muvofiq.

## 9.2. Dasturiy ta'minotni qo'lda nazorat qilish

Qo'lda nazorat qilish odatda ishlanmaning boshlang'ich bosqichlarida qo'llaniladi. U yoki bu bosqichda qabul qilingan barcha loyiha qarorlari ularning to'g'riligi va maqsadga muvofiqligi nuqtayi nazaridan imkon qadar oldinroq tahlil etiladi. Chunki, ishlanmaning dastlabki bosqichlarida bunday qarorlarni amalda tekshirib bo'lmaydi.

Qo'lda nazorat qilishning statistik va dinamik yondashuvlari farqlanadi.

*Statistik* yondashuvda dasturning boshqaruvi va axborot aloqalari, uning keladigan hamda chiqadigan ma'lumotlari tuzilmasi tahlil etiladi. *Dinamik* yondashuvda qo'lda testdan o'tkazish bajariladi, ya'ni berilgan dastlabki ma'lumotlar dasturini amalga oshirish jarayoni qo'lda modellashtiriladi.

Bunday tekshiruv uchun texnik topshiriq, tasnif, dasturiy mahsulotning tuzilmaviy va funksional sxemalari, ayrim komponentlar sxemasi dastlabki ma'lumotlar sanaladi. Dastur algoritmi va matnlari, shuningdek matn to'plami keyingi bosqichlarda ana shunday vazipredmeti o'taydi.

Shu narsa isbotlanganki, qo'lda nazorat qilish dastur samaradorligi va ishonchliligini oshirishga yordam beradi. Uning yordamida mantiqiy loyihalashtirish va kodlashtirishdagi xatolarning 30–70 foizini aniqlash mumkin. Demak, qo'lda nazorat qilishning bir yoki bir nechta usullari har bir dastur loyihasida qo'llanilishi shart.

Qo'lda nazorat qilishning asosiy usullari:

- boshlang'ich matn inspeksiyasi;
- oraliq ko'zdan kechirish;
- stol ortida tekshirish;
- dasturni baholash.

**Boshlang'ich matn inspeksiyasi.** Mazkur inspeksiya mutaxassislar guruhi tomonidan matnni o'rganishda xatolarni topish protsedurasi va usullarining to'plamidan iborat. Mutaxassislar guruhiga dastur muallifi, loyihachi, test bo'yicha mutaxassis va nufuzli dasturchi (dastur muallifi emas) kiradi. Inspeksiyaning umumiy protsedurasi quyidagi operatsiyalarni nazarda tutadi:

- guruh ishtirokchilariga oldindan dastur varaqasi va uning tavsifi beriladi;
- dasturchi dastur ishi mantig'i to'g'risida aytib beradi va inspektor savollariga javob beradi;
- dastur uzoq vaqt mobaynida dasturlashning yig'ilib qolgan xatolarini aniqlash uchun savollar ro'yxatini tahlil etadi.

**1. Ma'lumotlar nazorati**

- Hamma o'zgaruvchanlar initsiallashtirilganmi?
- Massiv va qatorlarning eng yuqori (yoki mavjud) o'lchami oshib ketmaganmi?
- Matritsalar bilan ishlashda qatorlar bilan ustunlar chalkashgan emasmi?

• O‘xshash nomga ega o‘zgaruvchanlar mavjudmi?

• Fayllardan foydalanadimi? Agar foydalanilsa, fayldan kiritishda uning tugallanganligi tekshiriladimi?

• Yoziladigan va o‘qiladigan simvollar (belgilar) turi mos keladimi?

• Tiplashtirilmagan o‘zgaruvchan omillar, ochiq massivlar, dinamik xotiradan foydalaniladimi? Agar foydalanilsa, format olinayotganda almashuvchilar turlari o‘zaro mos keladimi? Indekslar massivlar chegarasidan chiqib ketmaganmi?

**2. Hisoblash nazorati**

• Ifodalar to‘g‘ri yozilganmi (operatorlarni kuzatish tartibi)?

• Arifmetik bo‘lmagan o‘zgaruvchanlar ustidan hisoblash to‘g‘ri bajarilganmi?

• Turli tipdagi o‘zgaruvchanlar bilan hisoblash to‘g‘ri bajarilganmi?

• Razryad setkasi to‘lib ketishi mumkinmi?

• Aniqlik uchun qo‘yilgan talablar bilan hisoblash mos keladimi?

• Turli tipdagi o‘zgaruvchanlarni taqqoslash mavjudmi?

**3. Boshqaruvni uzatish nazorati**

• Sikllar to‘g‘ri tugallanadimi?

• Dastur tugallanadimi?

• Kirish shartlari buzilganligi uchun bajarilmaydigan sikllar mavjudmi? Hisoblash to‘g‘ri davom etadimi?

• Qidiruv sikllari mavjudmi? «Element topildi» yoki «Element topilmadi» holatlari sifatli ishlaydimi?

**4. Modullararo interfeyslar nazorati**

• Parametrlar va argumentlar ro‘yxati tartib, tip, o‘lchov birligi bo‘yicha mos keladimi?

• o‘zgarishi kerak bo‘lmagan argumentlarning qism dastur (dastur qismi)i o‘zgarmaydimi?

• Bir xil nomdagi global va lokal o‘zgaruvchilar harakati borasida buzilishlar ro‘y bermaydimi?

**Oraliq ko'rib chiqishlar.** Oraliq ko'rib chiqishlar xuddi inspeksiya kabi dastur matniga qaraydigan bir guruh shaxslar tomonidan xatolarni topish usuli to'plamidir. Bunday ko'rib chiqishning inspeksiyalash jarayoni bilan umumiy o'xshashligi bor, ammo xatolarni aniqlash protsedurasi va usullaridan farqlanadi. Oraliq nazoratni amalga oshiruvchi guruh 3—5 nafar mutaxassisdan iborat bo'ladi. Jumladan, rais yoki koordinator, barcha xatolarni qayd etuvchi kotib, testdan o'tkazish bo'yicha mutaxassis, dasturchi va mustaqil ekspert. Oraliq nazorat quyidagi vazifalarni amalga oshirishni nazarda tutadi:

- guruh ishtirokchilariga oldindan dastur varaqasi va uning tasnifi beriladi;
- majlis ishtirokchilariga bir necha testlar taklif etiladi;
- majlis ishtirokchilari dastur mantig'iga muvofiq xayolan har bir testni bajaradi. Bunda dastur holati (o'zgaruvchanlar belgilari) qog'oz yoki doskaga qayd etib boriladi;
- zarur hollarda dasturchiga loyihalashtirish mantig'i va yo'l qo'yilgan xatolar to'g'risida savollar beriladi.

Ko'pgina oraliq ko'rib chiqishda dasturchidan so'rashdan ko'ra kamroq xato topiladi.

**Stolda tekshirish.** Qo'lda test o'tkazishning ushbu qadimiy usuli eng avval paydo bo'lgan. Chunki u mutaxassislar guruhini talab etmaydi. Bu boshlang'ich matnni tekshirish yoki bir inson tomonidan bajariladigan oraliq ko'rib chiqish jarayoni. U dastur matnini o'qiydi, unda tez-tez uchraydigan xatolarning maxsus ro'yxati bo'yicha xatolar mavjudligini tekshiradi va dastur orqali test ma'lumotlarini o'tkazadi. Testdan o'tkazish tamoyillaridan kelib chiqqan holda stol ortida o'tirib tekshirishni dastur muallifi bo'lmagan inson olib borishi lozim. Bu usul nisbatan samarasiz. Chunki tekshirish tartibga solinmagan, unda fikr almashish va sog'lom raqobat yo'q.

**Dasturni baholash.** Bu usul test o'tkazish bilan bevosita bog'liq emas, lekin undan foydalanish dasturlash sifatini yaxshilaydi. Undan dasturni anonim baholash uchun uning umumiy sifati,

oddiy va tushunarligi atamalarida qo'llashda foydalaniladi. Usul maqsadi – nisbatan obyektiv baholash va dasturchilarning o'z-o'zini baholashlarini ta'minlash.

Shundan so'ng baho natijalari qiyoslanadi, tekshiruvchi umumiy sharhlaydi va dasturni yaxshilash yuzasidan tavsiyalar beradi.

## 9.3. Tuzilmaviy testlash

Tuzilmaviy testdan shuningdek, «marshrut» bo'yicha testdan o'tkazish deb ataladi. Chunki bunday holatda test to'plamlari algoritmida ko'rib chiqilgan marshrutlarni tahlil etish yo'li bilan tuziladi.

Marshrut deganda, bu holda boshlang'ich ma'lumotlarning aniq variantida bajariladigan dasturlar operatorlarining uzviyligi tushuniladi.

Tuzilmaviy testdan o'tkazish asosida dasturning barcha marshrutlarini maksimal to'liq testdan o'tkazish konsepsiyasi yotadi. Masalan, dastur algoritmi «shaxoblash»ni kiritsa, boshlang'ich ma'lumotlarning bitta to'plamida operatorlarning uzviyligi amalga oshirilishi mumkin. Shunga muvofiq dastur uchun variantlar bilan tarmoqlanishda farqlanuvchi marshrutlar mavjud bo'ladi.

Testdan o'tkazishning tuzilmaviy yondashuvi bir qator kamchiliklarga ega. Mazkur strategiya bo'yicha tuzilgan test to'plamlari:

- qoldirib ketilgan marshrutlarni topmaydi;
- qayta ishlanayotgan ma'lumotlarga bog'liq xatolarni topmaydi. Masalan, if operatorida (a–b)<eps – abs absolyut qiymat funksiyasining tushib qolganligi agar, a<b bo'lsagina namoyon bo'ladi;
- dastur to'g'riligiga kafolat berolmaydi (masalan, quyilashtirish bo'yicha tanlash o'rniga, ko'tarilish bo'yicha tanlash amalga oshirilgan bo'lsa).

Ma'lumotlarni testdan o'tkazish uchun test to'plamlarini tuzish quyidagi bir necha mezonlarga ko'ra amalga oshirilishi mumkin:

- operatorlarni qoplash;

- qarorlarni (o‘tishlarni) qoplash;
- shartlarni qoplash;
- qarorlar/shartlarni qoplash;
- shartlarni kombinatorli qoplash.

**Operatorlarni qoplash.** Mazkur mezon matnlarni shunday tanlashni nazarda tutadiki, bunda har bir dastur operatori kamida bir marta bajarilishi lozim. Bu zarur, ammo maqbul test uchun yetarli bo‘lmagan sharoit. Aytilganlarni misol bilan aniqlashtiramiz.

## 9.4. Funksional testlash

Dasturni tekshirish usullaridan biri ma’lumotlar yoki «qora quti» tamoyili bo‘yicha boshqaruv bilan testdan o‘tkazish sanaladi. Uning asosiy maqsadi, dastur harakati o‘z xususiyatlariga mos kelmaydigan sharoitlarni aniqlashdan iborat.

Ma’lumotlar bo‘yicha boshqaruvdan foydalangan holda dasturdagi barcha xatolarni topish uchun ma’lumotlarning mumkin bo‘lgan barcha to‘plamlarida testdan o‘tkazish lozim. Buyruqlar bajarilishi o‘tgan holatlarga bog‘liq bo‘ladigan dasturlar uchun barcha ketma-ketlik variantlarini tekshirish lozim. Shunisi qiziqki, tugallovchi testni ko‘p hollarda o‘tkazish imkoni yo‘q. Shuning uchun «oqilona» yoki «maqbul» test usuli amalga oshiriladi. Mazkur variant dastur xususiyatidan chetlashish bo‘lmasligiga kafolat bermaydi.

To‘g‘ri tanlangan test boshqa testlar sonini (kamida ikkita) kamaytirishi lozim. Funksional testdan o‘tkazishda test to‘plamlarini tuzishning quyidagi usullari farqlanadi:

- ekvivalent bo‘lish;
- chegara ahamiyatini tahlil etish;
- sabab-oqibat aloqalarini tahlil etish;
- xato to‘g‘risidagi farazlar.

**Ekvivalentli bo‘lish.** Mazkur usul tavsifi quyidagicha, dasturning kiruvchi ma’lumotlariga tegishli barcha to‘plamlar har bir parametr bo‘yicha guruhlarning bo‘linishi mumkin bo‘lgan eng oxirgi soniga

– ekvivalentlar sinfiga ajratiladi. Bunday sinflarning ma'lumotlar to'plami topilishiga qarab bir xil xatolarni birlashtiradi. Agar biror sinf to'plami ayrim xatolarni topsa, ushbu sinfning boshqa barcha testlari ushbu xatoni topadi deb faraz qilinadi yoki aksincha.

Ekvivalent asosida bo'lish usuli bilan testlarni ishlab chiqish ikkita bosqichda amalga oshiriladi. Birinchi bosqichda ekvivalentli sinflar ajratiladi, ikkinchisida testlar shakllantiriladi.

Ekvivalentlik sinflarini ajratish evristik jarayon sanaladi. Biroq, ayrim parametrlarning mumkin bo'lgan yoki bo'lmagan ahamiyatini o'z ichiga oluvchi to'plamlar ekvivalentligining alohida sinflariga ajratish maqsadga muvofiq sanaladi. Bunda quyidagicha qoidalar mavjud:

- ayrim x parametrlar [1, 999] intervalda qiymatlarni qabul qila olsa, u holda bitta to'g'ri sinf – $1 \le x \le 999$ va ikkita noto'g'ri sinf $x<1$ va $x>999$ ajratiladi;
- agar, kiruvchi shartlar tartib tusidagi qoida ko'lamini (masalan, «avtomobilda bir kishidan olti kishigacha yurishi mumkin» kabi) belgilasa, u holda ekvivalentlikning bitta to'g'ri va ikkita noto'g'ri sinfi va olti nafardan ko'p kishi belgilanadi;
- agar, kirish shartlari ko'plab kiruvchi ahamiyatlarni bayon etsa, demak dasturchi har bir ahamiyatni alohida ta'riflaydi (masalan, «grafik fayllar turlari: bmp, jpeg, vsd» kabi) deyishga asos bo'lsa, har bir ahamiyat uchun ekvivalentlikning to'g'ri sinfi va bitta noto'g'ri sinfi (masalan, txt) belgilanadi;
- agar, kirish shartlari «bo'lishi kerak» vaziyatini (masalan, «identifikatorning birinchi simvoli (ramzi) harf bo'lishi kerak») bayon etsa, u holda bitta to'g'ri ekvivalent sinfi (birinchi simvoli (ramzi) – harf) va bitta noto'g'ri ekvivalent sinfi (birinchi simvoli (ramzi) – harf emas) belgilanadi;
- agar ekvivalentlik sinfining turli elementlari dastur bilan bir xil ta'riflanmaydi, deyishga asos bo'lsa, u holda mazkur sinf ekvivalentlikning yanada mayda sinflariga bo'linadi.

Shunday qilib, ekvivalentlik sinflari texnik vazifada yoki spetsifikatsiyani aniqlashda har bir kiradigan ta'rif uchun belgilan-

gan chegaralarga yetib ajraladi. Har bir cheklashlar ikki yoki undan ortiq guruhlarga bo'linadi. Buning uchun maxsus blankalar, ekvivalentlik sinflari jadvallaridan foydalaniladi.

| Parametrlar qiymati uchun cheklashlash | To'g'ri ekvivalentlik sinflari | Noto'g'ri ekvivalentlik sinflari |
|---|---|---|
| | | |

**Chegara ahamiyatlari tahlili.** *Chegara qiymati* bu kirish ahamiyati yoki uning atrofidagi ekvivalentlik sinflari chegaralaridagi qiymatlardir. Tahlil shuni ko'rsatadiki, bu joylarda xato topilishi keskin ortadi. Masalan, uchburchak turlarining tahlili dasturida A+B>C o'rniga A+B≥S yozilgan bo'lsa, u holda chegara qiymati topshirig'i xatolikka olib keladi: chegara chizig'i uchburchak turlaridan biriga tegishli bo'ladi.

Chegara qiymatini tahlil etish usulini qo'llash ma'lum darajada ijodga va ko'rib chiqilayotgan muammoga ixtisoslashgan bo'ladi. Ayni paytda ushbu usulni qo'llash uchun bir necha umumiy qoida mavjud:

- agar, kirish shartlari qiymat chegarasini bayon etsa, u holda chegaradan – biroz chiqish holatlari uchun noto'g'ri kirish ma'lumotlariga testlar va chegara sohasi uchun testlar tuzilishi lozim. Masalan, agar soha [-1.0, +1.0] deb yozilgan bo'lsa, testlar quyidagicha generatsiyalangan bo'lishi kerak: -1.0; +1.0; -1.001 va +1.001;
- agar, kirish shartlari ahamiyatning diskret qatorini qanoatlantirsa, u holda minimal va maksimal qiymat uchun testlar hamda ushbu ikkita ahamiyatdan ko'p va kam ahamiyatga ega testlarni tuzish kerak. Masalan, kirish fayli 1 dan 255 tagacha yozuvni o'z ichiga olsa, 0,1,255 va 256 ta yozuvlarni tekshirish lozim;
- agar, chiqish qiymatini cheklash mavjud bo'lsa, ularni analogik ravishda testdan o'tkazish maqsadga muvofiqdir. Albatta, har doim ham tegishli natijalarni olib bo'lmaydi, ammo ushbu imkoniyat ko'rib chiqishga arziydi.

Qayd etilgan chegara qiymatidan tashqari boshqalarini ham izlash maqsadga muvofiq.

Chegara qiymati tahlili, agar u to‘g‘ri qo‘llangan bo‘lsa, testlarni loyihalashtirishning nisbatan foydali usullaridan biri sanaladi. Biroq, shuni nazarda tutish kerakki, chegaralovchi ahamiyatni aniqlash murakkab. Bu mazkur usulning kamchiligidir.

Qayd etilgan har ikki kirish ma’lumotlarini tadqiq etishga asoslangan. Ular ma’lumotlarni turlicha uyg‘unlashtirishda olinadigan natijalarni tekshirish imkonini beradi.

**Sabab-oqibat aloqalari tahlili.** Sabab-oqibat aloqalari tahlili yuqori natijali testlarni tizimli tanlash imkonini beradi. Usul mantiq algebrasidan foydalanadi hamda «Sabab», «oqibat» tushunchalari bilan operatsiyalaydi. Ushbu holatda alohida kirish shartlari yoki ekvivalentlik sinfi *sabab* hisoblanadi. *Oqibat* – chiqish shartlari yoki tizimning qayta tuzilishi. Barcha oqibatlarni sababga olib borib taqash, ya’ni sabab-oqibat aloqalarini aniqlash usul g‘oyasi sanaladi.

Testlarni tuzish bir necha bosqichda amalga oshiriladi. Avval, sabab-oqibat aloqalari jadvallari usulni yirik spetsifikatsiyalarga qo‘llashda kattalashib ketadi. Spetsifikatsiya «ishchi» uchastkalarga bo‘ladi, sabab-oqibat aloqalarining mustaqil guruhlariga oid ayrim jadvallar ajratib ko‘rsatiladi. So‘ngra spetsifikatsiyaga ko‘plab sabab va oqibatlar belgilanadi.

Keyin tasnifning semantik (mazmuniy) mundarijasini tahlil etish asosida «haqiqat» jadvali tuziladi. Unda mumkin bo‘lgan har bir sabab kombinatsiyasiga muvofiq ravishda oqibat belgilanadi. Bunda haqiqatni «1», yolg‘onni esa «0» shaklida belgilash maqsadga muvofiq. Shartning befarq holatini belgilash uchun «x» belgisini qo‘llash lozim.

Jadvalning har bir qatoriga test qayta tuziladi. Bunda imkon qadar testlarni mustaqil jadvallar testlari bilan birga qo‘llash lozim.

Mazkur usul yuqori natijalarga ega testlarni tuzish va dastlabki tavsiflarning to‘liq emasligini aniqlash imkonini beradi. Chegara qiymatini noadekvat tadqiq etish uning kamchiligi sanaladi.

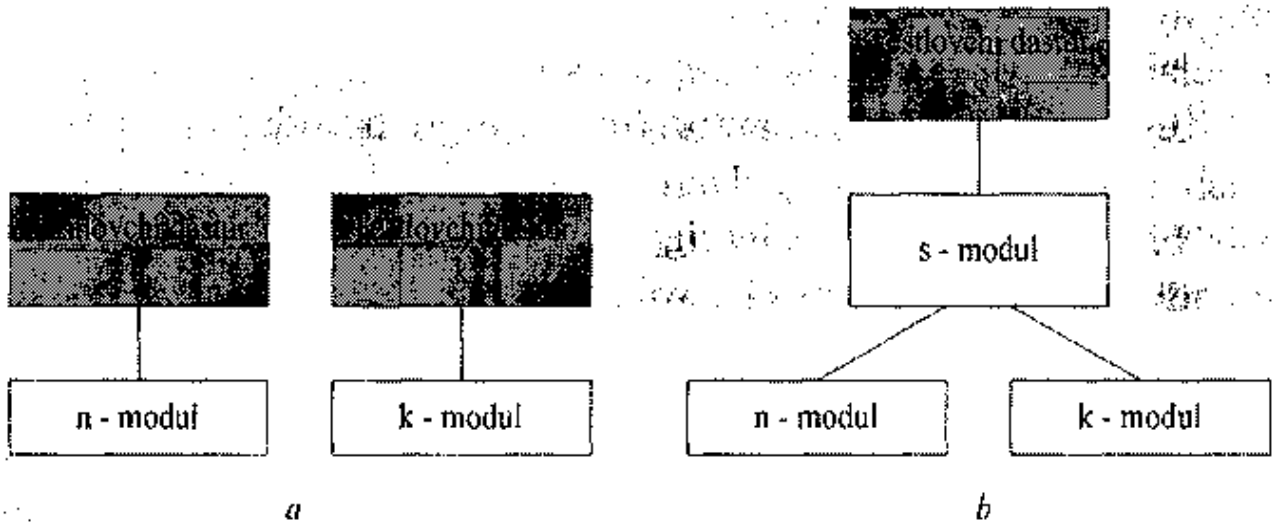
**Xato to‘g‘risida farazlar.** Yuqori malakaga ega bo‘lgan dasturchi ko‘pincha «Hech qanday usulni qo‘llamasdan» xatolarni topadi. Aslida esa bilib turib «xato to‘g‘risidagi faraz» usulini qo‘llaydi.

Xato to‘g‘risidagi faraz usuli protsedurasi ko‘p jihatdan intuitsiyaga asoslangan. Uning asosiy g‘oyasi ayrim ro‘yxatlarda bo‘lishi mumkin xatolar yoki vaziyatlarni sanab o‘tishi, so‘ngra ushbu ro‘yxat asosida test tuzishdan iborat. Boshqacha aytganda, loyihalashtirishda hisobga olinmagan alohida holatlarni sanab o‘tish joiz.

## 9.5. Modullarni testlash va kompleks testlash

Dasturiy ta’minot modullarini testlashda shuningdek, loyihalashtirish va kodlashtirishda ham yuqorilovchi ham pastlashuvchi yondashuvni qo‘llash mumkin.

**Yuqorilovchi testlash.** Yuqorilovchi testlash shuni nazarda tutadiki, har bir modul mavjud tasnifga muvofiq alohida testlanadi. So‘ng testlangan modul nisbatan yuqori modulga yig‘iladi va ularni testlaydi. Bunda iyerarxiyaning nisbatan quyi darajasidagi modullarini qo‘shish uchun foydalaniladigan modullararo interfeyslar tekshiriladi. Butun dasturiy mahsulot yig‘ilgunga qadar shunday davom etadi (9.2-rasm).




**9.2-rasm.** Yuqorilovchi yondashuvda dasturiy ta’minotni testlash: a – quyi darajadagi avtonom testlash; b – navbatdagi darajadagi testlash.

Bunday yondashuv to‘liq avtonom testlashni ta’minlaydi. Uning uchun modulga to‘g‘ridan-to‘g‘ri uzatiladigan test ketma-ketligini generatsiyalash oson. Biroq, uning jiddiy kamchiliklari mavjud. Birinchidan, yuqorilovchi testlashda, shuningdek, yuqorilovchi loyihalashda ham xatolar faqat loyihalashning yakunlovchi bosqichidagina aniqlanishi mumkin. Ikkinchidan, quyi darajadagi modullarni testlash uchun maxsus testlovchi dasturni ishlab chiqish lozim. U kerakli parametrlarga ega bo‘lgan va bizni qiziqtiruvchi modullarni chaqiruvchini ta’minlaydi. Ayni paytda ushbu interfeys ham xatoliklarga ega bo‘lishi mumkin.

**Pastlashuvchi testlash.** Pastlashuvchi testlash organik jihatdan pastlashuvchi loyihalash va ishlab chiqish bilan bog‘liq. Bu biror-bir modulni loyihalash tugallanishi va u kodlanib testlashga uzatilishi holatiga tegishli.

Bu holatda faqat asosiy modul testlanadi. Uni testlashda chaqiriladigan modullar harakatini u yoki bu darajada imitatsiyalaydigan modullar bilan almashtiriladi.

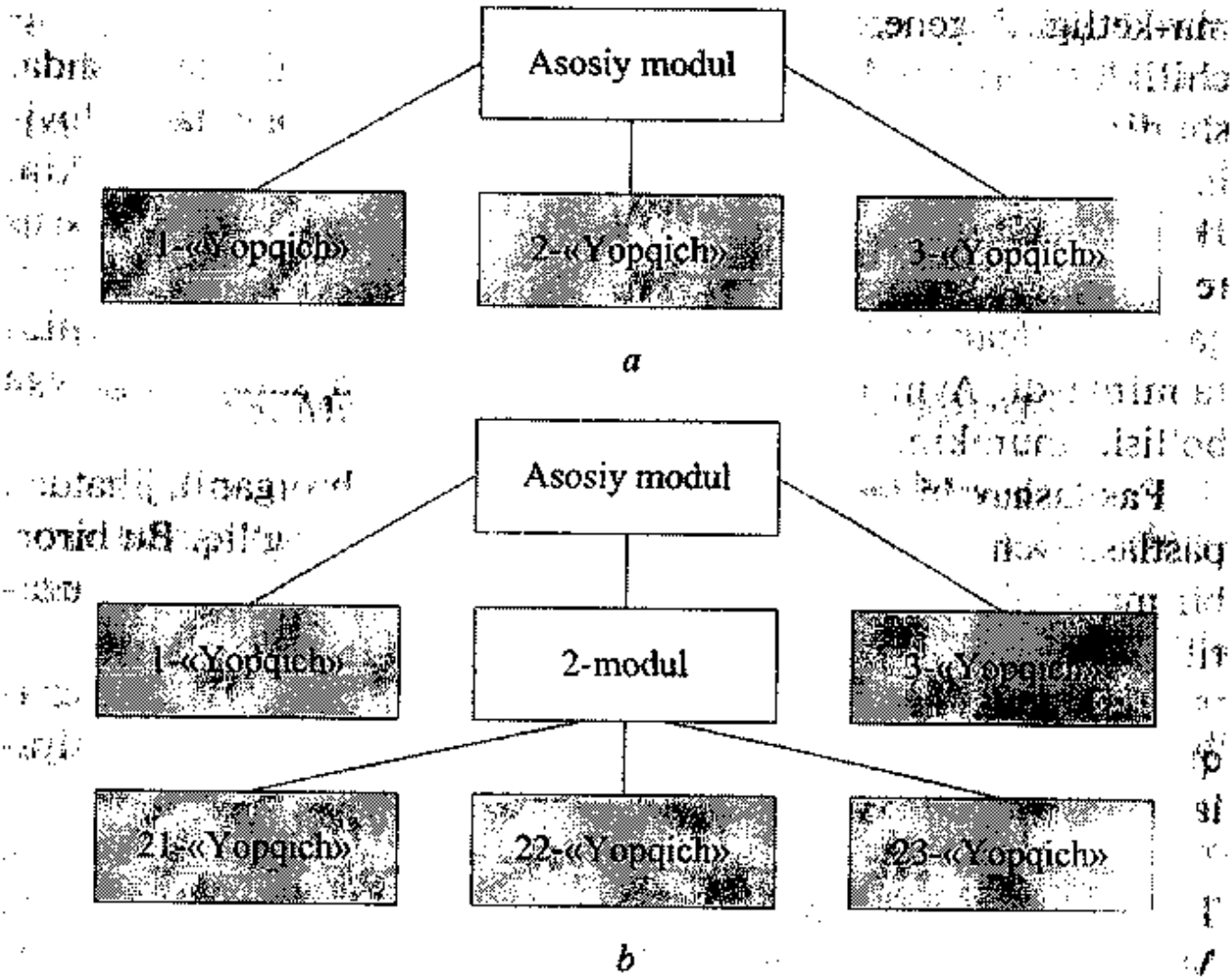
Bunday modullarni «yopqichlar» deb atash qabul qilingan. Testlanuvchi dasturlardan farqli ravishda «yopqichlar» juda sodda. Masalan, boshqaruv ularga berilganligini qayd etish mumkin, xolos. «Yopqichlar» ko‘pincha biror-bir qayd etilgan ma’lumotlarni qaytaradi.

Asosiy modulni testlash tugashi bilan, uning o‘zi chaqirgan modullar va kerakli «yopqichlar» qo‘shiladi, so‘ng ularning qo‘shma testlash jarayonini o‘tkazadi. Keyin butun tizim yig‘ilmaguncha navbatdagi modullar qo‘shilaveradi.

Pastlashuvchi testlashning asosiy kamchiligi unda modullarni avtonom testlash mavjud emasligidir. Modul ma’lumotlarni bevosita emas, chaqiruvchi modul orqali oladi. Shu bois uni yetarli darajada testlash ancha murakkab.

Mazkur usulning asosiy yutug‘i shundaki, u asosiy masalalarni ertaroq tekshiradi va dasturiy ta’minot mazmunida modullar kuchlanishini ko‘p marta sifatli testlaydi. Pastlashuvchi testlashda

buyurtmachi bilan tashqi dasturiy ta'minot (interfeys) ko'rinishini kelishish imkoni mavjud.




**9.3-rasm.** Testlashning boshlang'ich bosqichlari: a – asosiy modul bosqichi; b – ikki modul bosqichi.

**Kombinatsiyalashgan yondashuv.** Kombinatsiyalashgan yondashuv eng ko'p qo'llaniladi. Yuqori darajadagi modullar pastlashuvchi usul bilan, quyi darajadagi modullar esa yuqorilashuvchi usul bilan testlanadi. Mazkur usul bir tomondan interfeysni testlash bilan boshlanadi, ikkinchi tomondan quyi darajadagi modullarni sifatli avtonom testlashni ta'minlaydi.

**Mutaxassislar tomonidan dasturiy ta'minotni testlash.** Asosiy tamoyillarga ko'ra, dasturiy ta'minot muallif tomonidan testlanmasligi lozim. Shu bois, ushbu ishni qoidaga ko'ra boshqa dasturchi (imkon qadar ushbu soha mutaxassisi) amalga oshiradi.

## 9.6. Baholovchi testlash

Kompleks testlash yakunlangandan so‘ng baholovchi testga kirishiladiki, uning maqsadi dasturni asosiy talablarga muvofiqligini testlashdir. Testlashning ushbu bosqichi, ayniqsa bozorda sotish uchun mo‘ljallangan dasturiy mahsulotlar uchun g‘oyatda muhimdir.

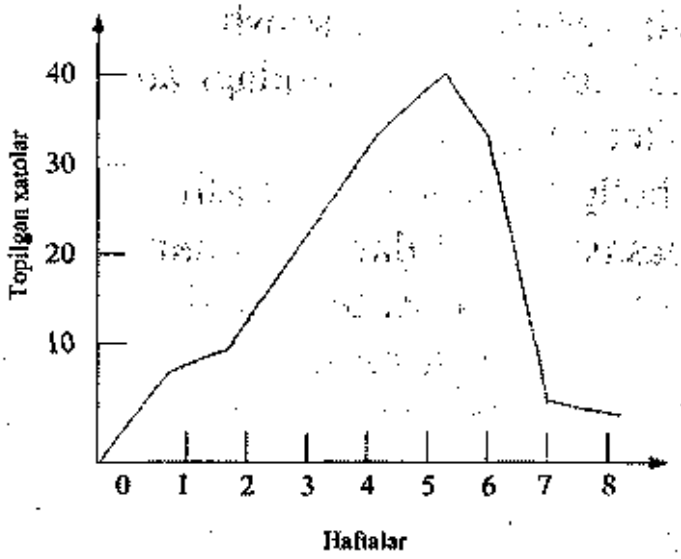
«Tizimni yaxlit testlash» deb ham nomlanadigan baholovchi testlash quyidagi turlarni o‘z ichiga oladi:

- foydalanish qulayligini testlash – dasturiy mahsulot va uning hujjatlarini texnik topshiriqdagi asosiy qoidalariga muvofiqligini ketma-ket tekshirish;
- mumkin bo‘lgan hajmda testlash – dasturning eng ko‘p ma’lumotlar, masalan, matnlar, jadvallar, katta miqdordagi fayllar hajmlarida ish qobiliyatini tekshirish;
- eng yuqori bosimda testlash – qisqa muddatda kelib tushgan ma’lumotlarga ishlov berish bo‘yicha dastur imkoniyatini tekshirish;
- tasarruf qulayligini testlash – dasturiy ta’minlash bilan yuzaga keladigan psixomantiqiy omillar tahlili. Ushbu testlash interfeys qulay emasligini, rangli yoki tovushli yo‘ldoshlik salbiy ta’sir ko‘rsatmasligi va hokazolarni aniqlashga imkon beradi;
- himoyani testlash – masalan, axborotga sanksiyalanmagan ruxsatli kirishdan muhofazani tekshirish;
- mahsuldorlikni testlash – topshirilgan bosimdagi konfiguratsiyada uning imkoniyatini belgilash;
- xotiraga talablarni testlash – tezkor va tashqi xotirada mavjud ehtiyojlarni aniqlash;
- jihozlar konfiguratsiyasini testlash – turli jihozlarda dasturiy ta’minot ish qobiliyatini tekshirish;
- muvofiqlikni testlash – tizimning navbatdagi versiyasi tizimdagi oldingi versiyada yaratilgan fayllar bilan ishlash imkoniyatini ta’minlovchi maxsus konvekterlarni ko‘zda tutishi shart bo‘lgan holatlarda ma’lumotlar formatlari o‘zgarishi bo‘yicha versiyalar vorisiyligini tekshirish;

• qurilma qulayligini tekshirish – qurilma qulayliklarini tekshirish;

• ishonchlilikni testlash – tegishli matematik modellardan foydalangan holda ishonchlilikni tekshirish;

• qayta tiklanishni testlash – dasturiy ta'minlanish qayta tiklanishini, masalan uskuna va dastur izdan chiqishlaridan so'ng ma'lumotlar bazasini o'z ichiga oluvchi qayta tiklanishni tekshirish;




**9.3-rasm.** Xatolarni topish grafigiga misol.

• xizmat ko'rsatish qulayligini testlash – dasturiy ta'minotga kiritilgan xizmat ko'rsatish vositalarini tekshirish;

• hujjatlashtirishni testlash – hujjatlashtirishni batafsil tekshirish. Masalan, agar hujjatlashtirish misollarga ega bo'lsa, ularning barchasini sinab ko'rish lozim;

• protseduralarni testlash – tizimda taklif etilgan qo'l jarayonlarini tekshirish.

Tabiiyki, ushbu tekshirishlarning barchasi texnik vazifaga nomutanosiblikni izlash sanaladi. Testlashning barcha turlari bajarilgandan so'nggina dasturiy mahsulot foydalanuvchiga yoki amalga oshirilishga topshirilishi mumkin. Biroq, amaliyotda baholi testlashning ham turlari bajarilavermaydi. Chunki bu juda qimmat va og'ir ish.

**Nazorat savollari:**

1. Dasturni testlashdan maqsad nima? Nima uchun?
2. Sizga tanish bo'lgan dasturiy ta'minot sifatini nazorat qilish turlarini sanab bering.
3. Testlashning qanday yondashuvlarini bilasiz?
4. Nima uchun funksional testlash «Qora quti uslubi bo'yicha testlash» deb nomlanadi? Funksional testlash uslublarini sanab bering va ularning har biri qanday holatda qo'llanilishini belgilab bering.
5. Nima uchun tuzilmaviy testlash «oq yoki shaffof quti uslubi bo'yicha testlash» deb nomlanadi? Tuzilmaviy testlash uslublarini sanab o'ting va har birining imkoniyatlarini aniqlang. Tuzilmaviy testlashning qanday uslubi xatolarni topishda ko'proq imkoniyat yaratadi?

# 10. DASTURIY TA'MINLASHNI SOZLASH

Dasturni sozlash dasturiy ta'minotni ishlab chiqishning eng murakkab bosqichlaridan biri bo'lib, quyidagilar bo'yicha teran bilimlarni talab qiladi:

- foydalaniluvchi texnik vositalarni boshqarish spetsifikasi;
- operatsion tizimlar;
- dasturlash muhitlari va tillari;
- amalga oshiriluvchi jarayonlar;
- turlicha xatolarning tabiatlari va spetsifikasi;
- tegishli dasturiy vositalar va sozlash metodikalari.

Mazkur bob keyingi ikki masalaning muhokamasiga bag'ishlanadi.

## 10.1. Xatolar tasnifi

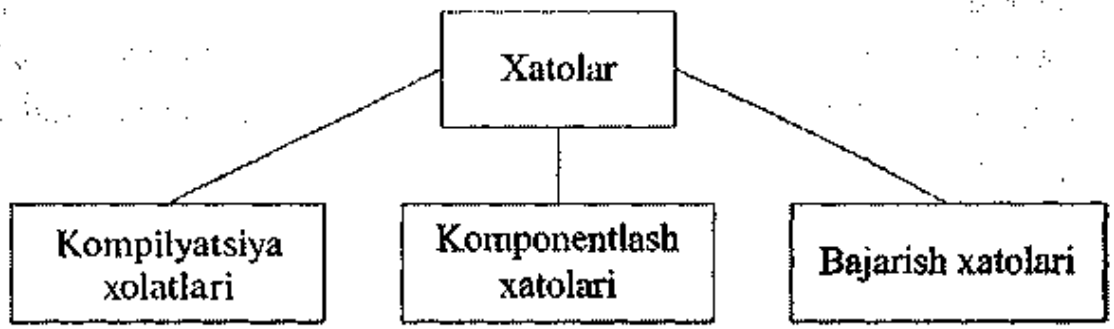
Sozlash — dasturiy ta'minotni testlashda aniqlangan xatolarni lokallashtirish hamda tuzatish jarayonidir. Boshlanishi me'yoriy hisoblash jarayonining buzilishini yuzaga keltiruvchi dastur operatorini belgilash jarayoni lokallashtirish deyiladi. Xatoni tuzatish uchun uning sababini, ya'ni xato mavjud operatorni yoki fragmentini belgilash zarur. Xatolarning sabablari aniq-ravshan bo'lishi ham, juda chuqur yashiringan bo'lishi ham mumkin.

Yaxlit e'londa sozlash murakkabligi quyidagi sabablar bilan shartlaniladi:

- dasturchidan foydalaniluvchi texnik vositalarni boshqarish spetsifikasi, operatsion tizim, dasturlash muhitlari va tili, amalga oshiriluvchi jarayonlar, turlicha xatolarning tabiati hamda spetsifikasi, tegishli dasturiy vositalarning va sozlashning metodikasi bo'yicha teran bilimlarni talab etadi;
- psixologik jihatdan diskomfort sanaladi, chunki qoida bo'yicha cheklangan vaqt sharoitlarida o'z xatolarini izlash zarur;
- dasturning turli qismlarida xatolarning o'zaro ta'siri ehtimoli bor, masalan bir modul xotirasi sohasini o'chirish hisobidan manzillashtirish xatolari tufayli boshqalari yuzaga kelishi mumkin;

• aniq shakllantirilgan sozlash metodikalari mavjud emas.

Xatolar namoyon bo'ladigan ishlov berish jarayoniga muvofiq quyidagilar farqlaniladi (10.1-rasm).




**10.1-rasm.** Dasturga ishlov berish bosqichi bo'yicha xatolar tasnifi.

Sintaktik xatolar – dasturning sintaktik va qisman semantik tahlilini bajarishda kompilyator (translyator, interpretator) tomonidan qayd etiluvchi xatolar.

Komponentlash xatolari – dastur modellarini birlashtirishda komponentlovchi (aloqalar muharriri) aniqlagan xatolar.

Bajarish xatolari – dastur bajarilishida operatsion tizim, apparat vositalari yoki foydalaniluvchi aniqlagan xatolar.

**Sintaktik xatolar.** Sintaktik xatolar eng sodda guruhga mansubdir, chunki til sintaksisi, qoidaga ko'ra qat'iy rasmiylashtirilgan va xatolar keng sharh bilan birgalikda uning joylashuvi ko'rsatilgan holda kuzatiladi. Bunday xatolarning sababini aniqlash, qoida bo'yicha, qiyin emas va hatto til qoidalarini aniq bilganda ham bir necha «progon», ya'ni o'tkazish davomida mazkur tildagi barcha xatolarning yo'qolishiga erishiladi.

**Komponentlash xatolari.** Nomidan ayonki, komponentlash xatolari tashqi murojaatlarni hal etishda aniqlangan muammolar bilan bog'liq. Masalan, boshqa modul tagdasturiga murojaat ko'zda tutilgan, modullarni birlashtirishda esa bu topilmaydi yoki parametrlar ro'yxatlari ulashmaydi. Aksariyat hollarda bunday turdagi xatolarni tez lokallashtirishga va bartaraf etishga muvaffaq bo'ladi.

**Bajarish xatolari.** Oldindan ko'ra olish eng qiyin guruhga bajarish xatolari mansubdir. Eng avvalo, ular turli tabiatga ega bo'lishi,

muvofiq ravishda turlicha namoyon bo'lishi mumkin. Xatolarning bir qismi operatsion tizim bilan aniqlanadi va hujjatlashtiriladi. Bunday xatolar namoyon bo'lishining turlicha usuli ajratiladi:

- mashinaviy buyruqlar bajarilishining nazorat sxemalari qayd etgan xato haqida xabar paydo bo'lishi, masalan, razryad to'ri ortiqcha to'ldirilganda, «nolga bo'lish» vaziyatida, manzillashtirish buzilganligida va hokazo;
- operatsion tizim aniqlagan xato haqida xabar paydo bo'lishi, masalan xotira himoyasi buzilganda, yozuvdan himoyalangan qurilmaga yozuv yozishga urinishda, yuklatilgan nomli fayl mavjud bo'lmaganda va hokazo;
- kompyuterning «osilib qolishi», jumladan operatsion tizimni qayta yuklatishsiz dasturni yakunlashga erishiladigan sodda «osilib qolishi» ham, ishni davom ettirish uchun qayta yuklatish zarur bo'ladigan «og'ir» «osilib qolishi» ham;
- olingan natijalarning kutilayotganlari bilan to'g'ri kelmasligi.

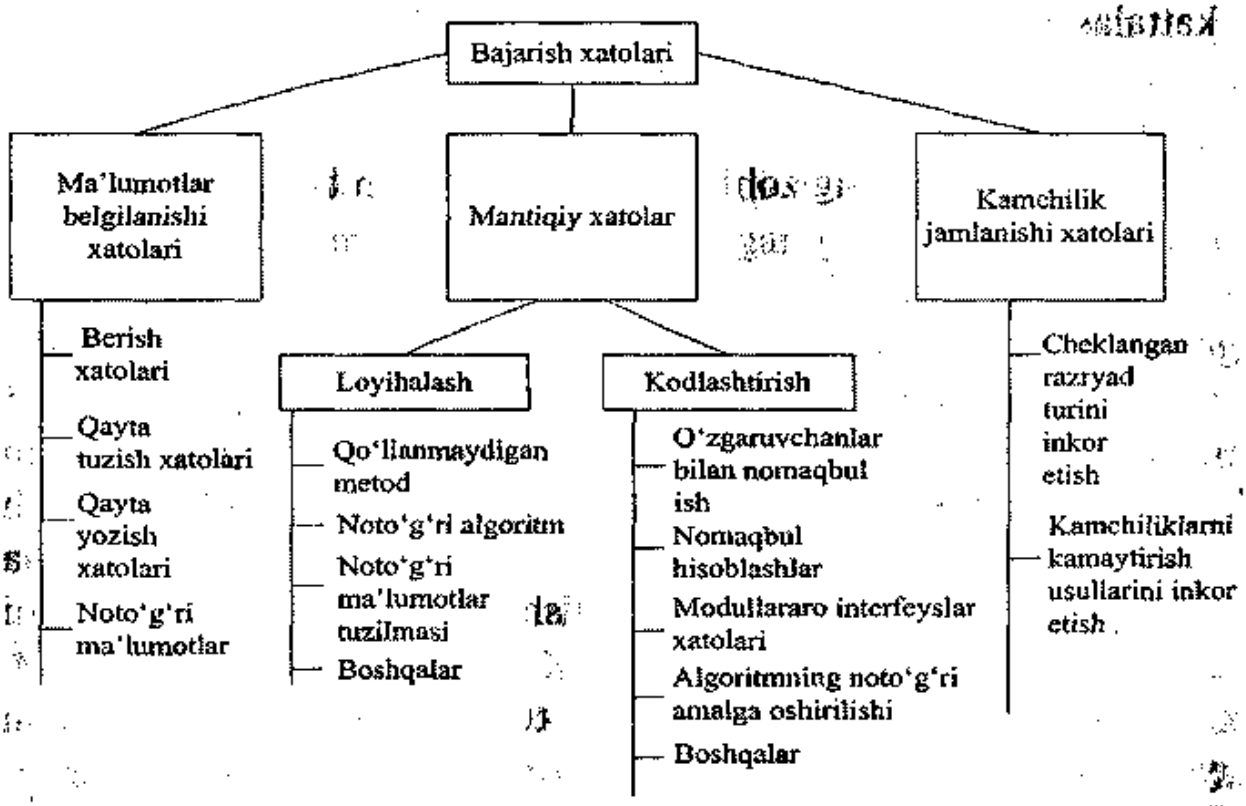
Bajarish xatolari sabablari g'oyat turli-tuman, shu boisdan lokallashtirish o'ta murakkab bo'lishi mumkin. Xatolarning barcha ehtimolli sabablarini quyidagi guruhlarga ajratish mumkin.

- boshlang'ich ma'lumotlarning noto'g'ri belgilanishi;
- mantiqiy (mantiqiy) xatolar;
- hisoblash natijalari xatolarining jamlanishi (10.2-rasm).

Boshlang'ich ma'lumotlarning noto'g'ri belgilanishi agar kiritish-chiqarish operatsiyalarini bajarishda har qanday xatolar: berish xatolari, qayta tuzish xatolari, qayta yozish xatolari va ma'lumotlar xatolari yuzaga kelganda sodir bo'ladi. Bunda maxsus texnik vositalardan va xatolardan himoyali dasturlashdan foydalanish mazkur xatolarning faqat bir qismini aniqlash hamda bartaraf etishga imkon beradiki, buni unutmaslik lozimligi shubhasiz.

Mantiqiy xatolar turli tabiatga ega bo'ladi. Xususan, ular loyihalashda yo'l qo'yilgan, masalan usullarni tanlashdagi, algoritmlarni ishlab chiqishdagi yoki sinflar tuzilmalarinini belgilashdagi xatolardan kelib chiqishi ham, modulni kodlashtirishda bevosita kiritilishi ham mumkin. So'nggi guruhga quyidagilar mansub:

• o‘zgaruvchanlardan nomaqbul foydalanish xatolari, masalan ma’lumotlar tiplarining muvaffaqiyatsiz tanlovi, initsiallashtirilgunga qadar o‘zgaruvchanlardan foydalanish, massivlar belgilanishi chegaralaridan chetga chiquvchi indekslardan foydalanish, ma’lumotlar tipini ravshan yoki noravshan qayta belgilashdan foydalanishda ma’lumotlar tiplari muvofiqliklari buzilishlari, xotirada joylashgan, tiplashtirilmagan o‘zgaruvchanlar, ochiq massivlar, birlashmalar, dinamik xotira manzilli arifmetika buzilishlari va hokazo;




**10.2-rasm.** Ehtimolli sabablar bo‘yicha bajarish bosqichi xatolarining tasnifi.

• hisoblashlar xatolari, masalan noarifmetik o‘zgaruvchanlar ustidan nomaqbul hisoblashlar, butun sonli arifmetikadan nomaqbul foydalanish, hisoblashlar jarayonida ma’lumotlar tiplarini nomaqbul qayta tuzish, arifmetik va mantiqiy ifodalar uchun operatsiyalarni bajarilishida ustuvorliklarni bilmaslik bilan bog‘liq xatolar va h.k;

• modullararo interfeys xatolari, masalan tizimli kelishuvlarni inkor etish, parametrlarni berishda tiplar va bosqichlilik buzilishi,

rasmiy va fakt parametrlar o'lchov birliklari birligiga rioya qilmaslik, lokal hamda global o'zgaruvchanlar harakati sohasining buzilishi;

• boshqa kodlashtirish xatolari, masalan kodlashtirishda dastur mantiqiysini noto'g'ri amalga oshirish, muayyan dasturlash tili xususiyatlarini yoki cheklovlarini inkor etish.

Yuqorida sanalgan xatolar yuzaga kelishining barcha sabablarini sozlash jarayonini nazarda tutish lozim. Bundan tashqari, sozlash murakkabligi quyidagi omillarning ta'siri oqibatida ham kattalashadi:

• xatolarning bilvosita namoyon bo'lishi;

• xatolarning o'zaro ta'sir imkoniyatlari;

• turli xatolarning zohiran bir xil namoyon bo'lishining olinishi imkoniyatlari;

• ayrim xatolarning ishga tushirilishdan keyingi ishga tushirilishigacha namoyon bo'lishi takroriyligining mavjud emasligi — stoxastik xatolar deb nomlanuvchi xatolar;

• dasturga ayrim o'zgarishlarni kiritishda tadqiq qilinayotgan vaziyatda xatolarning tashqi namoyon bo'lishini bartaraf etish imkoniyati, masalan dasturga diagnostik fragmentlar qo'yilganda xatolarnining tashqi namoyon bo'lishi yo'qotilishi yoki o'zgarishi mumkin;

• dastur alohida qismlarining turli-dasturchilar tomonidan yozilishi.

## 10.2. Dasturiy ta'minlashni sozlash usullari

Dastur sozlanishi har qanday holatda xato haqidagi barcha mavjud axborot o'ylanishi hamda mantiqan mushohada qilinishini nazarda tutadi. Aksariyat xatolarni bilvosita davomatlar bo'yicha dasturlar matnlarining va qo'shimcha axborot olinmasidan testlanish natijalarini batafsil tahlil qilish vositasida aniqlash mumkin. Bu borada quyidagicha turli usullardan foydalaniladi:

• qo'lda testlash;

• induksiya;

• deduksiya;

• teskari kuzatilish.

**Qo'lda testlash.** Bu mazkur guruhda eng sodda va tabiiy usul. Xatoni aniqlaganda birgalikdagi ishda xato aniqlangan testli majmuadan foydalangan holda testlanuvchi dastur qo'lda bajaradi.

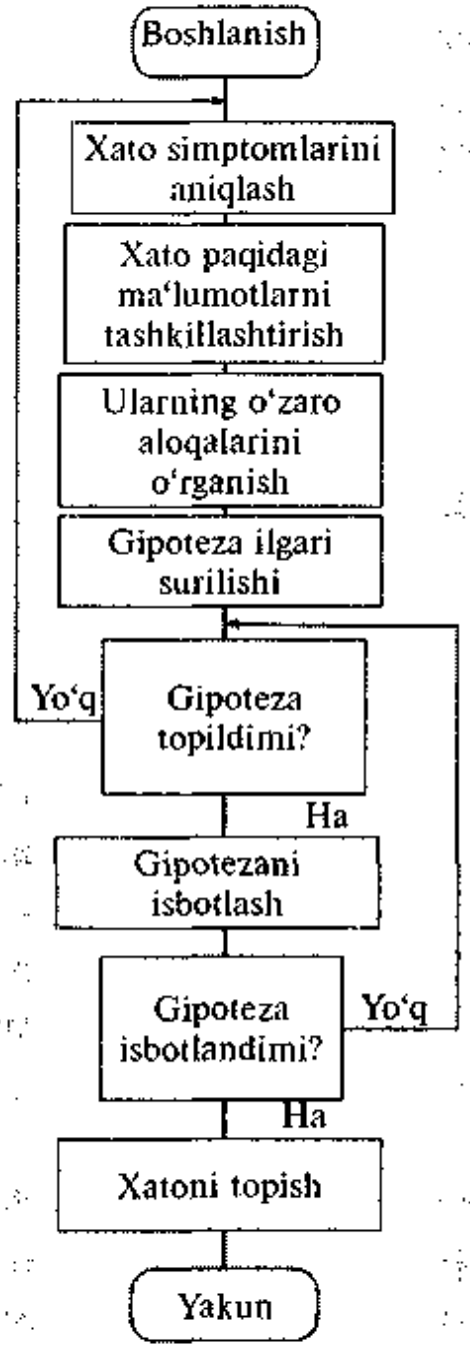
**Induksiya usuli.** Usul hisoblashlarning noto'g'ri natijalari yoki xato haqidagi xabar sifatida namoyon bo'lishi mumkin xato simptomlarining bajarish tahliliga asoslangan. Agar kompyuter shunchaki «osilib qolsa», eng so'nggi olingan natijalar va foydalanuvchi harakatlaridan kelib chiqqan holda xatoning namoyon bo'lish fragmenti hisoblanadi. Shu tariqa olingan axborot dasturning tegishli fragmenti ko'rilgan holda tashkillashtiriladi va batafsil o'rganadi. Mazkur harakatlar natijasida xatolar haqidagi gipotezalar ilgari surilib, ularning har biri tekshiriladi. Agar gipoteza to'g'ri bo'lsa, xato haqidagi axborot detallashtiriladi, boshqacha bo'lsa boshqa gipotéza ilgari suriladi. Induksiya usuli bilan sozlash bajarilishining ketma-ketligi 10.3-rasmda algoritm sxemasi tarzida ko'rsatilgan.

Eng mas'uliyatli bosqich xato simptomlarini aniqlash. Xato haqidagi ma'lumotlarni tashkillashtirgan holda uning namoyon bo'lishi haqida ma'lum barcha narsalarni yozib qo'yish maqsadga muvofiq, bunda xatoli fragment me'yorida bajariluvchi vaziyatlar ham, xato namoyon bo'ladigan vaziyatlar ham qayd etiladi. Agar ma'lumotlarni o'rganish natijasida hech qanday gipoteza paydo bo'lmasa, demak xato haqida qo'shimcha axborot zarur. Qo'shimcha axborotni, masalan o'xshash testlar bajarilishi natijasida olish mumkin.

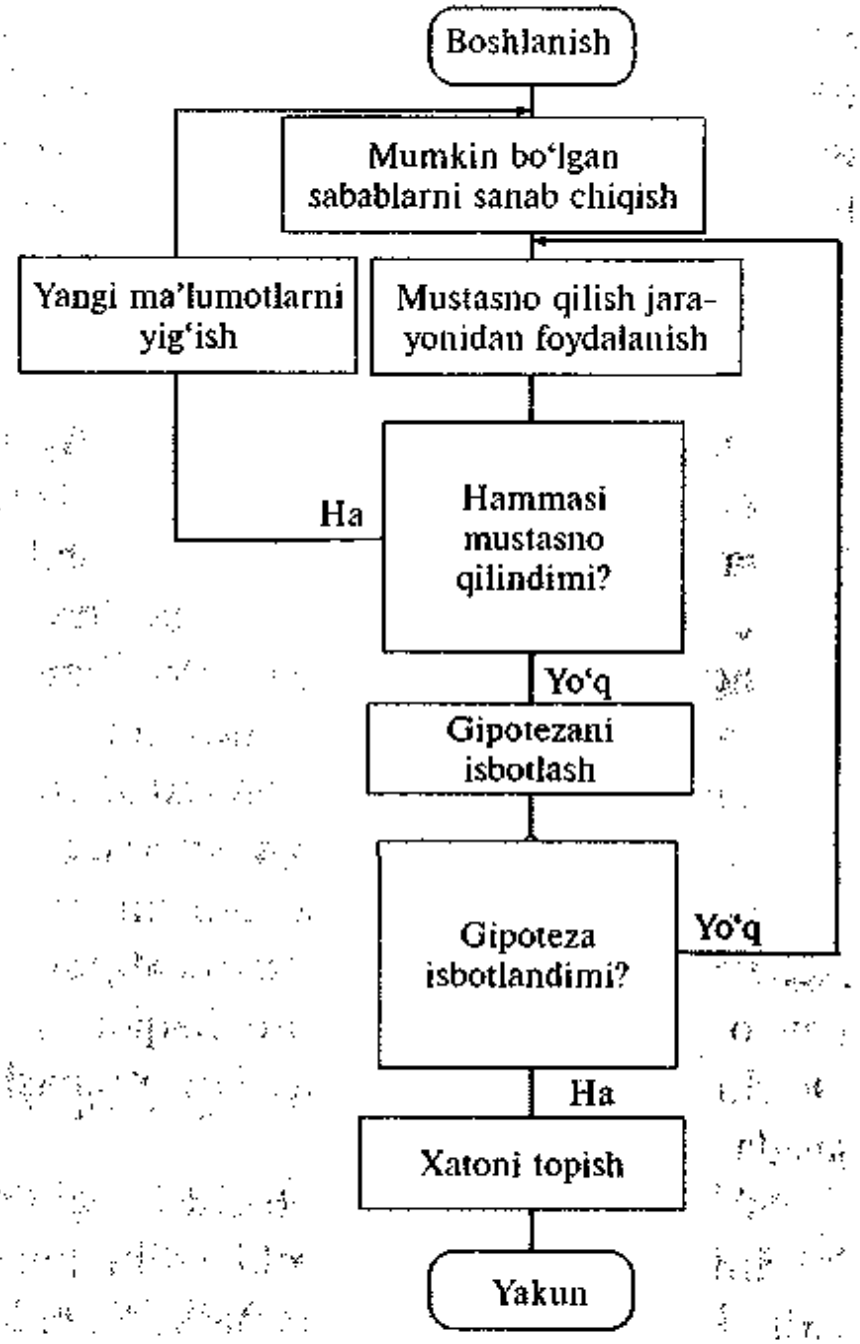
Isbotlash jarayonida tegishli gipoteza xatoning barcha ko'rinishlarini izohlashi yoki izohlamasligini aniqlashga uriniladi, agar hammasini izohlamasa, u holda gipoteza noto'g'ri, yoki xatolar bir nechta bo'ladi.

**Deduksiya usuli.** Deduksiya usuli bo'yicha dastlab xatoning tegishli ko'rinishini yuzaga keltirishi (mumkin) ko'plab sabablarni shakllantiradi. So'ngra sabablarni tahlil qilgan holda mavjud ma'lumotlarga zid bo'lganlari mustasno qilinadi. Agar barcha sabablar mustasno qilinsa, u holda tadqiq qilinayotgan fragment-

ning qo'shimcha testlanishini bajarish lozim. Aksincha holatda esa eng ko'p ehtimolli gipotezani isbotlashga harakat qilinadi. Agar gipoteza xatoning olingan alomatlarini izohlasa, demak xato topildi, boshqacha bo'lsa, keyingi sabab tekshiriladi (10.4-rasm).




**10.3-rasm.** Induksiya usuli bilan sozlash jarayoni sxemasi.




**10.4-rasm.** Deduksiya usuli bilan sozlash jarayonining sxemasi.

**Teskari kuzatilish usuli.** Katta bo'lmagan dasturlar uchun teskari kuzatilish usulidan foydalanish samaralidir. Noto'g'ri natijani, chiqarish nuqtasidan boshlaniladi. Ushbu nuqta uchun mavjud natijaning

olinishini keltirishi mumkin bo'lgan asosiy o'zgaruvchanlar birliklari haqida gipoteza quriladi. So'ngra shu gipotezadan kelib chiqqan holda o'zgaruvchanlarning avvalgi nuqtadagi ifodalari haqida taxminlar qilinadi. To xato aniqlangunga qadar jarayon davom ettiriladi.

### 10.3. Qo'shimcha axborot olish usullari va vositalari

Xato haqida qo'shimcha ma'lumotni olish uchun qo'shimcha testlarni bajarish yoki maxsus usullar va vositalardan foydalanish mumkin:

- sozlovchi chiqarish;
- integralli sozlash vositalari;
- mustaqil sozlovchilar.

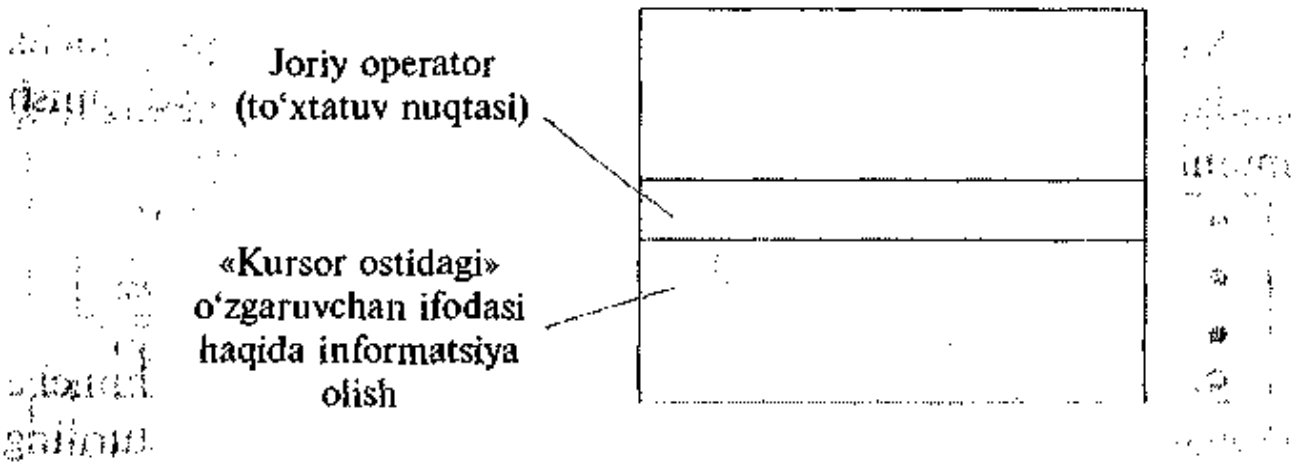
**Sozlovchi chiqarish.** Usul biriktiruv nuqtalarida qo'shimcha sozlovchi chiqarish dasturga qo'shilishini talab etadi. Dasturning asosiy o'zgaruvchanlari o'z ifodalarini o'zgartiradigan algoritm nuqtalari biriktiruv nuqtalari o'z aksini topadi. Masalan, ayrim ifodalar massivining o'zgarish sikli boshlangunga qadar va shu sikl yakunlangandan so'ng sozlovchi chiqarishni nazarda tutishi lozim bo'ladi. (Agar sozlovchi chiqarish siklda nazarda tutilsa, odatda taftishi murakkab bo'luvchi ortiqcha ko'p ifodalar chiqariladi). Bunda chiqarilgan ifodalarning tahlilini bajargach, dasturchi noto'g'ri ifodalar olingan paytni aniqlashtirishi va xatoning sababi haqida xulosa chiqarishi lozimligi nazarda tutiladi.

**Integralli sozlash vositalari.** Aksariyat zamonaviy dasturlash muhitlari (Delphi, Builder C++, Visual studio va h.k.) maksimal samarador sozlashni ta'minlovchi sozlash vositalarini o'z ichiga oladi. Ular quyidagilar uchun imkon beradi:

- dasturni qadamlar bo'yicha bajarish, bunga tagdasturga kirish birgalikdagisi ham, ularni yaxlit bajarish ham kiradi;
- to'xtatuv nuqtalarini ko'zda tutish;
- kursor bilan ko'rsatilgan operatorgacha bo'lgan dasturni bajarish;
- qadamli bajarishda har qanday o'zgaruvchanlarning borligini aks ettirish;

• xabarlar oqimini kuzatib borish va h.k.

10.5-rasmda dasturning Delphi to'xtatuv nuqtasiga yetgunga qadar qadamli bajarish tartibiga o'tishi paytidagi ko'rinishi tasvirlangan. Dasturchi xuddi shu paytda o'zini qiziqtiruvchi o'zgaruvchanlar ifodasiga qarab chiqish imkoniyatiga ega bo'ladi.




**10.5-rasm.** Delphi to'xtatuv nuqtasiga yetgunga qadar qadamli bajarish tartibiga o'tish (ramkada «kursor ostidagi» o'zgaruvchining ifodasi ko'rsatilgan).

Xato tabiatini aniqlash uchun mashinaviy kodlarni, bayroqlarni tahlil qilish hamda dasturni va xotira ifodalarini 16 lik tarzda taqdim etish mumkin (10.6-rasm).

Xato sababini ko'rib chiqilgan usullardan biridan foydalangan holda belgilanadi. Bunda gipotezalarni tekshirish uchun integralli sozlash vositalaridan ham foydalanish mumkin.

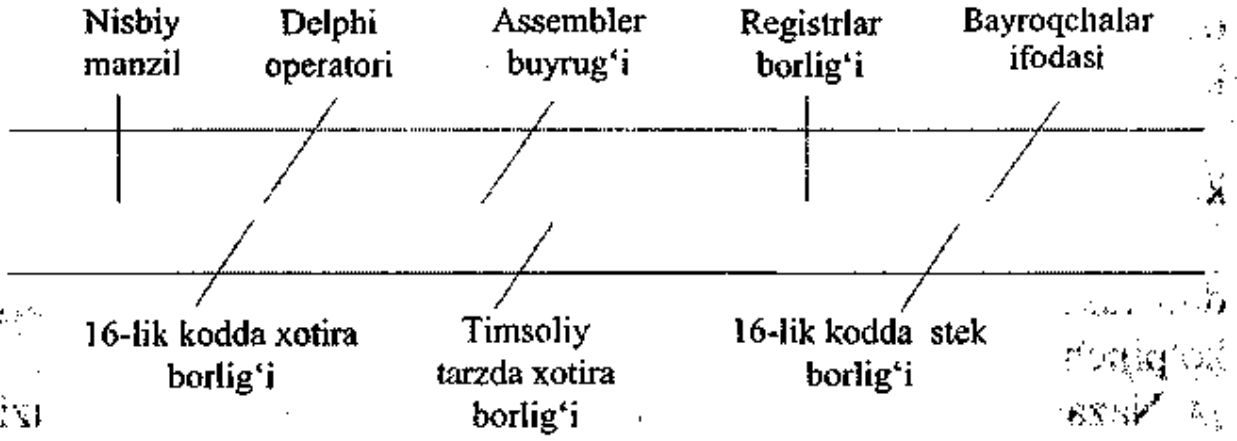
## 10.4. Dasturiy ta'minlashni sozlashning umumiy metodikasi

Barcha yuqorida aytilganlarni umumlashtirib, Ms DOS va Win 32 opearatsion tizimlarini bajarish uchun universal dasturlash tillarida yozilgan.

Dasturiy ta'minotni sozlashning quyidagi metodikasini taklif etish mumkin:

*1-bosqich,* xatoning namoyish bo'lishini o'rganish – agar qandaydir xabar berilgan yoki noto'g'ri yoxud notugal natijalar berilgan bo'lsa, ularni o'rganish hamda qanday xato shunday namoyon bo'lishi mumkinligini tushunishga harakat qilish zarur. Bunda sozlashning induktiv va deduktiv usullaridan foydalaniladi. Nati-

jada xato xarakteri to‘g‘risida versiyalar ilgari suriladi va ularni tekshirish zarur. Buning uchun xato haqida qo‘shimcha axborot olishning usullari va vositalarini qo‘llash mumkin.




**10.6-rasm.** 16 lik kodda dasturni sozlashda ekran ko‘rinishi (CPV tartibi).

Agar xato topilmasa yoki tizim «osilib qolsa» ikkinchi bosqichga o‘tiladi.

*2-bosqich,* xatoni lokallashtirish – bajarilishida nazarda tutilgan hisoblash jarayonidan og‘ish yuz bergan muayyan fragmentni belgilash. Lokallashtirish quyidagicha bajarilishi mumkin:

- dastur qismlarini kesish yo‘li bilan, bunda dasturning ayrim qismini kesishda xato yo‘qolib qolsa, bu xatoning shu qism bilan bog‘liqligini ham, kiritilgan o‘zgartirish xato namoyon bo‘lishini o‘zgartirganini ham anglatadi;
- bizni qiziqtiruvchi dasturning qadamli tartibdagi fragmentini bajarishga hamda xatoning namoyon bo‘lish joyi va xarakteri haqida qo‘shimcha axborotni olishga, masalan ko‘rsatilgan o‘zgaruvchanlar borlig‘ini aniqlashga imkon beruvchi sozlash vositalaridan foydalanish bilan birgalikda.

Bu borada agar noto‘g‘ri natijalar olingan bo‘lsa, u holda qadamli tartibda tegishli, natijaning shakllanishi jarayoniga xos kalit nuqtalar tekshiriladi.

Aynan yuqorida ta’kidlanganidek, xatoga u paydo bo‘lgan joyda yo‘l qo‘yilmagan bo‘lishi ham mumkin. Muayyan holatda shunday bo‘lsa, keyingi bosqichga o‘tiladi.

*3-bosqich,* xato sababini aniqlash – ikkinchi bosqich natijalarini o'rganish va xatoning ehtimolli sababi versiyalarini shakllantirish. Mazkur versiyalarni tekshirish zarur, ehtimol bunda operatorlar ketma-ketligini yoki o'zgaruvchanlarning ifodalarini ko'rib chiqish uchun sozlash vositalaridan foydalanilishi mumkin.

*4-bosqich,* xatoni tuzatish, birgalikda bajarilishi, xatoga olib kelgan barcha operatorlarga tegishli o'zgarishlarni kiritish.

*5-bosqich,* takroriy testlash – barcha testlarni avval boshidan takrorlash, chunki aniqlangan xatolarni tuzatishda dasturga ko'pincha yangilari kiritiladi.

Nazarda tutish lozimki, sozlash jarayonini dasturlashga tuzilmaviy yondashuvning asosiy tavsiyalariga ergashgan hamda jiddiy ravishda soddalashtirish zarur:

- dasturni «yuqoridan-quyiga», interfeysdan ishlab beruvchi tagdasturlarga uni tagdasturlar qo'shilishi davomida testlagan holda kuchaytirish lozim;
- foydalanuvchiga u kiritgan ma'lumotlarni tekshirish uchun chiqarish va ularni kiritilgandan so'ng darhol yo'l qo'yilishi bo'yicha tekshirish kerak;
- algoritmning barcha biriktiruv nuqtalarida (tarmoqlarda, tagdasturlar chaqiruvida) asosiy ma'lumotlarning chiqarilishini ko'zda tutish zarur.

**Nazorat savollari:**

1. Qanday jarayon sozlash deyiladi? Uning murakkabligi nimada?
2. Xatolarning asosiy tiplarini ayting. Dasturni bajarishda ular qanday namoyon bo'ladi?
3. Sozlashning asosiy usullarini sanab ko'rsating. Ular o'rtasidagi tafovut nimadan iborat? Xatolar mavjud har qanday dasturni olib, yuqorida sanalgan usullarning har biridan foydalangan holda xatoni topishga intiling. Qanday usul siz uchun sodda va tabiiyroq hamda buning sababi nimada?
4. Xatolar haqida qo'shimcha axborot olishning qanday vositalarini bilasiz?

# ADABIYOTLAR RO'YXATI

1. Агабеков Л.К., Иванова Г.С. Программирование на С++. Ч. 1. Средства процедурного программирования. Учебное пособие. – М.: Изд-во МГТУ им. Баумана, 1999.

2. Агабеков Л.Е., Иванова Г.С. Программирование на С++. Ч.2. Средства объектно-ориентированного программирования: Учеб. пособие. – М.: Изд-во МГТУ им. Баумана, 1996.

3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++, 2-е изд.: Пер. с англ. – М.: Бином, СПб.: Невский диалект, 1998.

4. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998.

5. Вирт И. Алгоритмы и структуры данных. Пер. с англ. – М.: Мир, 1989.

6. Иванова Г.С. Основы программирования. Учеб. для вузов. – М.: Изд-во МГТУ им. Баумана, 2001.

7. Иванова Г.С. Технология программирования. Учебник для вузов. 2-е изд., стереотип. – М.: Изд-во МТТУ им. Н.Э. Баумана, 2003. 320 с.

8. Кабулов В.К, Файзуллаев А.Ф., Назиров Ш.А. Ал-Хоразмий, алгоритм, алгоритмизация. – Ташкент: Фан, 2006. 664 с.

9. Sh.A. Nazirov, M.M. Musayev, A. Nematov, R.V. Qobulov. Delphi tilida dasturlash asoslari. – Toshkent: G'ofur G'ulom nomidagi nashriyot-matbaa ijodiy uyi. 2007, 280 b.

10. Sh.A. Nazirov, R.V. Qobulov. Obyektga mo'ljallangan dasturlash. – Toshkent: G'ofur G'ulom nomidagi nashriyot-matbaa ijodiy uyi. 2007, 184 b.

11. Sh.A. Nazirov, F.M. Nuraliyev, B.Sh. Aytmuratov. Rastr va vector grafika. – Toshkent: G'ofur G'ulom nomidagi nashriyot-matbaa ijodiy uyi. 2007, 192 b.

12. Sh.A. Naziov, R.V. Qobulov, A.H. Haydarov, A. Nematov, N.A. Aripova, Q.S. Raxmonov. Kompyuter va ofis jihozlaridan foydalanish. – Toshkent: G'ofur G'ulom nomidagi nashriyot-matbaa ijodiy uyi. 2007, 160 b.

13. Назиров Ш.А., Неъматов А., Кабулов Р.В. Основы программирования в Delphi. Учебное пособие. – Ташкент, 2007

Издательско-полиграфический творческий дом им. Чулпана. 320 с.

14. Назиров Ш.А., Кабулов Р.В. Объектно-ориентированное программирование. Учебное пособие. – Ташкент, 2007. Издательско-полиграфический творческий дом им. Чулпана. 160 с.

15. Назиров Ш.А., Кабулов Р.В., Арипова Н.А., Урынбаев С.К., Неъматов А., Рахманов К.С. Компьютерные офисные приложение. Учебное пособие. – Ташкент, 2007. Издательско-полиграфический творческий дом им. Чулпана. – 270 с.

16. Назиров Ш.А. Технология программирования. Учебное пособие. – Ташкент, 2007. Издательско-полиграфический творческий дом им. Чулпана. 128 с.

17. Ш.А. Назиров. Основы программирования. Учебное пособие. – Ташкент, 2007. Издательско-полиграфический творческий дом им. Гафура Гуляма. 300 с.

18. Ш.А. Назиров. Основы алгоритмизации. Учебное пособие. – Ташкент, 2007. Издательско-полиграфический творческий дом им. Гафура Гуляма. 96 с.

19. Sh.A. Nazirov. Guyunter Divald. Dasturlash asoslari. O‘quv qo‘llanma. – Toshkent, Sharq NMAK. 2007.

20. Sh.A. Nazirov, N.G. Mardonovalar. Dasturlash maxsus kursi. O‘quv qo‘llanma. – Toshkent, Sharq NMAK. 2007.

21. Sh.A. Nazirov, R.V. Qobulov, F.M. Nuraliyev. Obyektga mo‘ljallangan dasturlash Web–sahifalarni dasturlash. O‘quv qo‘llanma. – Toshkent, Sharq NMAK. 2007.

22. Непейвода Н.Н. Стили и усулы программирования. Лекции 2004 г. – М.: Ижевск: Институт компьютерных исследований. 2004 г. 328 с.

23. Н.Н. Непейвода, И.Н. Скоплин. Основания программирования. – Москва, Ижевск:. Институт компьютерных исследований, 2003 г. 864 с.

(Footnotes)

* Interfeys parametrlari miqdoriga bog‘liq

# MUNDARIJA

**KIRISH** . . . . . **3**

1. DASTURLASH TEXNOLOGIYASI. ASOSIY TUSHUNCHALAR VA YONDASHUVLAR . . . . . 5

2. DASTURIY MAHSULOTLAR TEXNOLOGIKLIGINI TA'MINLASH USULLARI. . . . . 51

3. DASTURIY TA'MINOTGA VA UNI LOYIHALASH UCHUN BOSHLANG'ICH MA'LUMOTLARGA TALABLARNI BELGILASH . . . . . 85

4. TUZILMAVIY YONDASHUVDA DASTURIY TA'MINOT TALABLARINI TAHLIL ETISH VA IXTISOSLIGINI ANIQLASH . . . . . **112**

5. TUZILMAVIY YONDASHUVDA DASTURIY TA'MINOTNI LOYIHALASHTIRISH . . . . . **143**

6. OBYEKTLI YONDASHUVDA TALABLAR TAHLILI VA DASTURIY TA'MINOT SPETSIFIKATSIYALARINI BAHOLASH. . . . . 168

7. OBYEKTLI YONDASHUVDA DASTURIY TA'MINOTNI LOYIHALASHTIRISH. . . . . 183

8. FOYDALANUVCHI INTERFEYSINI ISHLAB CHIQISH. . . . . 201

9. DASTURIY MAHSULOTLARNI TESTDAN O'TKAZISH . . . . . 249

10. DASTURIY TA'MINLASHNI SOZLASH . . . . . 266

**ADABIYOTLAR RO'YXATI** . . . . . **277**

Nazirov Shodmankul Abdirozikovich,
Ivanova Galina Sergeyevna,
Gaynazarov Sultan Mamanazarovich

# DASTURLASH TEXNOLOGIYALARI

*Darslik*

Muharrir: *M. Tursunova*
Musahhih: *M. Turdiyeva*
Dizayner sahifalovchi: *D. Ermatova*

O‘zbekiston faylasuflari milliy jamiyati nashriyoti.
100029, Toshkent shahri, Matbuotchilar ko‘chasi, 32-uy.
Tel.: 236-55-79; Faks: 239-88-61.

Nashriyot litsenziyasi: AI №216, 03.08.2012.
Bosishga ruxsat etildi 26.09.2014. «Uz-Times» garniturasi. Ofset usulida chop etildi. Qog‘oz bichimi 60x84 $^1/_{16}$. Bosma tabog‘i 17,5. Nashr hisob tabog‘i 18,0. Adadi 500 nusxa. Buyurtma № 43.

«START-TRACK-PRINT» XK bosmaxonasida chop etildi.
Manzil: Toshkent shahri, 8-mart ko‘chasi, 57-uy.

18500-00
22.9-7
H 196

ISBN 978-9943-391-81-9

O‘ZBEKISTON FAYLASUFLARI
MILLIY JAMIYATI