N. XOJIYEVA, B. SHARIPOV, SH. MUXSINOV,
O. RO'ZIBAYEV, S. MUMINOV

DASTURIY TA'MINOT QURILMASI VA EVOLYUTSIYASI

O'QUV QO'LLANMA

TOSHKENT - 2023

O‘ZBEKISTON RESPUBLIKASI RAQAMLI TEXNOLOGIYALAR VAZIRLIGI

MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

N. XOJIYEVA, B. SHARIPOV, SH. MUXSINOV, O. RO‘ZIBAYEV, S. MUMINOV

DASTURIY TA’MINOT QURILMASI VA EVOLYUSIYASI

O‘quv qo‘llanma

TOSHKENT - 2023

UO‘K 519.6:811.512.1
KBK: 22.18: 81.2

N.Xojiyeva, B.Sharipov, Sh.Muxsinov, O.Ro‘zibayev, S.Muminov. Dasturiy ta’minot qurilmasi va evolyusiyasi. O‘quv qo‘llanma. – T.: OOO “Muxr-Press”, 2023. – 255 bet.

ISBN 978-9943-9193-1-0

Mazkur o‘quv qo‘llanmada dasturiy ta’minotni konstruksiyalash asoslari yoritib berilgan. Qo‘llanmada dasturiy ta’minotni konstruksiyalashning tamoyillari taqdim etilgan, konstruksiyalashning asosiy elementlari tasnifi ochib berilgan, dasturiy ta’minotni ishlab chiqish metodologiyalari ko‘rib chiqilgan.

O‘quv qo‘llanma Dasturiy injiniring bakalavriat ta’lim yo‘nalishida tahsil olayotgan talabalariga va dasturiy ta’minotni konstruksiyalash yo‘nalishida faoliyat olib borayotgan mutaxassislarga tavsiya etiladi.

UO‘K 519.6:811.512.1
KBK: 22.18: 81.2

Taqrizchilar:

Sh. M. Gulyamov - Islom Karimov nomidagi Toshkent davlat texnika unversiteti “Ishlab chiqarish jarayonlarini avtomatlashtirish” kafedrasi professori, t.f.d.

A.T. Raxmonov. - Muhammad al-Xorazmiy nomidagi TATU, “Tizimli va amaliy dasturlashtirish” kafedrasi dotsenti, texnika fanlari nomzodi

O‘quv qo‘llanma Muhammad al-Xorazmiy nomidagi Toshkent axborot texnologiyalari universiteti ilmiy-metodik Kengashining qarori bilan chop etildi.

ISBN 978-9943-9193-1-0

KIRISH

“Har qanday ahmoq kompyuter tushunadigan kod yozishi mumkin. Yaxshi dasturchilar odamlar tushunadigan kod yozadilar”.

Martin Fouler (1999)

Biz o‘z oldimizga mamlakatimizda Uchinchi Renessans poydevorini barpo etishdek ulug’ maqsadni qo‘ygan ekanmiz, buning uchun yangi Xorazmiylar, Beruniylar, Ibn Sinolar, Ulug’beklar, Navoiy va Boburlarni tarbiyalab beradigan muhit va sharoitlarni yaratishimiz kerak. Bunda, avvalo, ta’lim va tarbiyani rivojlantirish, sog’lom turmush tarzini qaror toptirish, ilm-fan va innovasiyalarni taraqqiy ettirish milliy g’oyamizning asosiy ustunlari bo‘lib xizmat qilishi lozim.[1]

Hozirgi vaqtda jamiyatni axborot kommunikasiya texnologiyalarsiz tasavvur qilib bo‘lmaydi. Umuman olganda, har bir mamlakatni rivojlanishini zamonaviy axborot-kommunikasiya texnologiyalarini sohalarga joriy qilinishi bilan baholash mumkin. Mamlakatimizda xam buning uchun bir qator ishlar amalga oshirilib kelinmoqda. Xususan, xukumatimiz tomonidan zamonaviy axborot-kommunikasiya texnologiyalarini joriy etish va rivojlantirish (21.03.2012 yildagi PQ-1730), axborot-kommunikasiya texnologiyalari sohasida kadrlar tayyorlash tizimini yanada takomillashtirish (26.03.2013 yildagi PQ-1942), dasturiy ta’minot vositalari ishlab chiquvchilarni rag’batlantirishni yanada kuchaytirish (20.09.2013 yildagi PQ-2042) bo‘yicha chora-tadbirlarni keltirib o‘tish mumkin.

Bundan tashqari, “O‘zbekiston Respublikasi Axborot Texnologiyalari va Kommunikasiyalarini Rivojlantirish Vazirligini tashkil etish to‘g’risida”gi (04.02.2015 yil PF-4702) O‘zbekiston Respublikasining Prezidentining Farmonida yangi tashkil etilgan vazirlikning asosiy vazifalaridan biri qilib “raqobatdosh dasturiy mahsulotlarning mamlakatimizda ishlab chiqarilishini va ichki bozorini hamda ularga ko‘rsatiladigan xizmatlarni rivojlantirishga ko‘maklashish va uning muvofiqlashtirilishini ta’minlash, iqtisodiyotning real sektori tarmoqlarida va iste’molchilarda zamonaviy dasturiy mahsulotlar, axborot tizimlari va axborot resurslarini joriy etish” etib belgilangan. Ma’lumki, xozirgi kunda dasturiy ta’minot yoki dasturiy mahsulotni

[1] O‘zbekiston Respublikasi Prezidenti Shavkat Mirziyoyevning Oliy Majlisga Murojaatnomasi

yaratish dasturiy ta’minot ishlab chiquvchi mutaxassislar jamoasi orqali amalga oshiriladi. Umuman olganda, xozirda, har bir dasturiy ta’minot yoki dasturiy mahsulotni ishlab chiqishga amaliy loyiha sifatida qarash mumkin. Chunki, har bir ishlab chiqilgan dasturiy ta’minot biror bir sohadagi (ijtimoiy, iqtisodiy, sanoat, xalq xo‘jaligi va boshqalar) ma’lum bir muammolarni hal qilishga yo‘naltirilgan bo‘ladi.

Dasturiy ta’minotni ishlab chiqishga tizimli va kompleks yondashuv dasturiy injiniring (Software Engineering) tushunchasini keltirib chiqardi. Kelinglar, dasturiy injiniring tushunchasini vujudga kelishi sabablariga to‘xtalib o‘taylik. Dasturiy ta’minotni ishlab chiqish faoliyatini rivojlanishini dasturiy ta’minot bilan bog’liq bo‘lgan bir qator muammolarni vujudga kelishi va ularni hal qilish uchun amalga oshirilgan ishlar orqali izohlash mumkin. Dasturiy ta’minotni ishlab chiqish bo‘yicha vujudga kelgan asosiy muammolar quyidagilardan iborat bo‘lgan:

- dasturiy ta’minot tan-narxining juda yuqoriligi;
- talab etilayotgan dasturiy ta’minotni yaratishning murakkabligi;
- dasturiy ta’minotni ishlab chiqish jarayonlarini boshqarish va bashoratlash bo‘yicha zarurat paydo bo‘lishi va h.k.

Dasturiy ta’minotni konstruktsiyalash – bu dasturiy ta’minot ishlab chiqishni samaradorligini oshirish va optimallashtirish bilan shug’ullanuvchi amaliy fan bo‘lib, u dasturiy ta’minotni yaratishni loyihalashtirish(tahlil qilish), ishlab chiqish, joriy qilish va kuzatishni ilmiy asoslangan usullarini o‘z ichida mujassamlashtirgan.

Ushbu o‘quv qo‘llanmada dasturiy ta’minotni konstruktsiyalsh tushunchalari, dasturiy ta’minot yashash sikli, dasturiy ta’minot ishlab chiqishning modellari va texnologiyalari, dasturiy ta’minot arxitekturasi, dasturiy ta’minotni testlash va tekshirish yo‘llari, dasturiy ta’minotni ishlab chiqishdagi extimolli holatlar va ularni bartaraf etish yo‘llari va usullari, dasturiy ta’minot sifati, dasturiy ta’minotni ishlab chiqish va testlash bo‘yicha standartlar masalalari bayon etilgan.

1-BOB. DASTURIY TA’MINOTNI KONSTRUKSIYALASHGA KIRISH

1.1. Dasturiy ta’minotni konstruksiyalash tushunchasi

“Dasturiy ta’minot qurilmasi va evolyusiyasi” fanini o‘qitishdan maqsad – talabalarga dasturiy ta’minotni konstruksiyalash bo‘yicha bilimlarning nazariy asoslarini, dasturiy ta’minotni konstruksiyalashning tushunchalarini, dasturiy ta’minotni konstruksiyalash usullarini, dasturiy ta’minotni konstruksiyalash va rivojlantirish tamoyillarini o‘rgatish hamda ularni amaliyotda tadbiq etish ko‘nikmasini hosil qilishdan iborat.

“Dasturiy ta’minot qurilmasi va evolyusiyasi” fanining vazifasi – nazariy bilimlar, amaliy ko‘nikmalar, dasturiy ta’minotni konstruksiyalash va rivojlantirish jarayonlariga uslubiy yondoshuv hamda ilmiy dunyoqarashni shakllantirish, dasturiy ta’minotni konstruksiyalashdagi metodlar va ularning mazmun-mohiyatini, dasturiy ta’minotni konstruksiyalashning o‘rni va ahamiyatini ochib berish.

Dasturiy ta’minot (rus. Программное обеспачение, ingl. Software) — bu Kompyuterda ma’lum bir turdagi vazifani bajarish uchun ishlab chiqilgan vositadir. Aynan shu dasturiy ta’minotgina kompyuter — „quruq temir“ degan atamani yo‘qqa chiqargan. Dasturiy vositalar kompyuter tomonidan qo‘llaniladigan barcha dasturlar to‘plamidir. Dasturiy ta’minot 3 guruhga bo‘linadi:

1. Tizimli dasturiy ta’minot (unga turli yordamchi vazifalarni bajaruvchi dasturlar kiradi);
2. Amaliy (unga foydalanuvchiga aniq bir foydalanish sohasida ma’lumotlarga ishlov berish va qayta ishlashni amalga oshiruvchi dasturlar kiradi);
3. Uskunaviy dasturlar.

Dasturiy ta’minotni konstruktsiyalash atamasi kodlashtirish, tekshirish, modulli sinov, integratsiya tekshiruvi va nosozliklarni tuzatish kabi jarayonlardan iborat dasturiy tizimni batafsil ishlab chiqishni tavsiflaydi. Ushbu bilim sohasi boshqa sohalar bilan bog’liq. Eng kuchli bog’liqlik dasturiy ta’minotni konstruksiyalash (Software Design) va dasturiy ta’minotni sinash (Software Testing) sohalari bilan mavjud. Buning sababi, dasturiy ta’minotni konstruktsiyalash jarayonining o‘zi konstruksiyalash va sinov faoliyatining muhim jihatlariga tegishlidir. Bundan tashqari, konstruktsiyalash konstruksiyalash va sinov natijalariga asoslanadi. Loyihalash, konstruksiyalash va sinov o‘rtasidagi

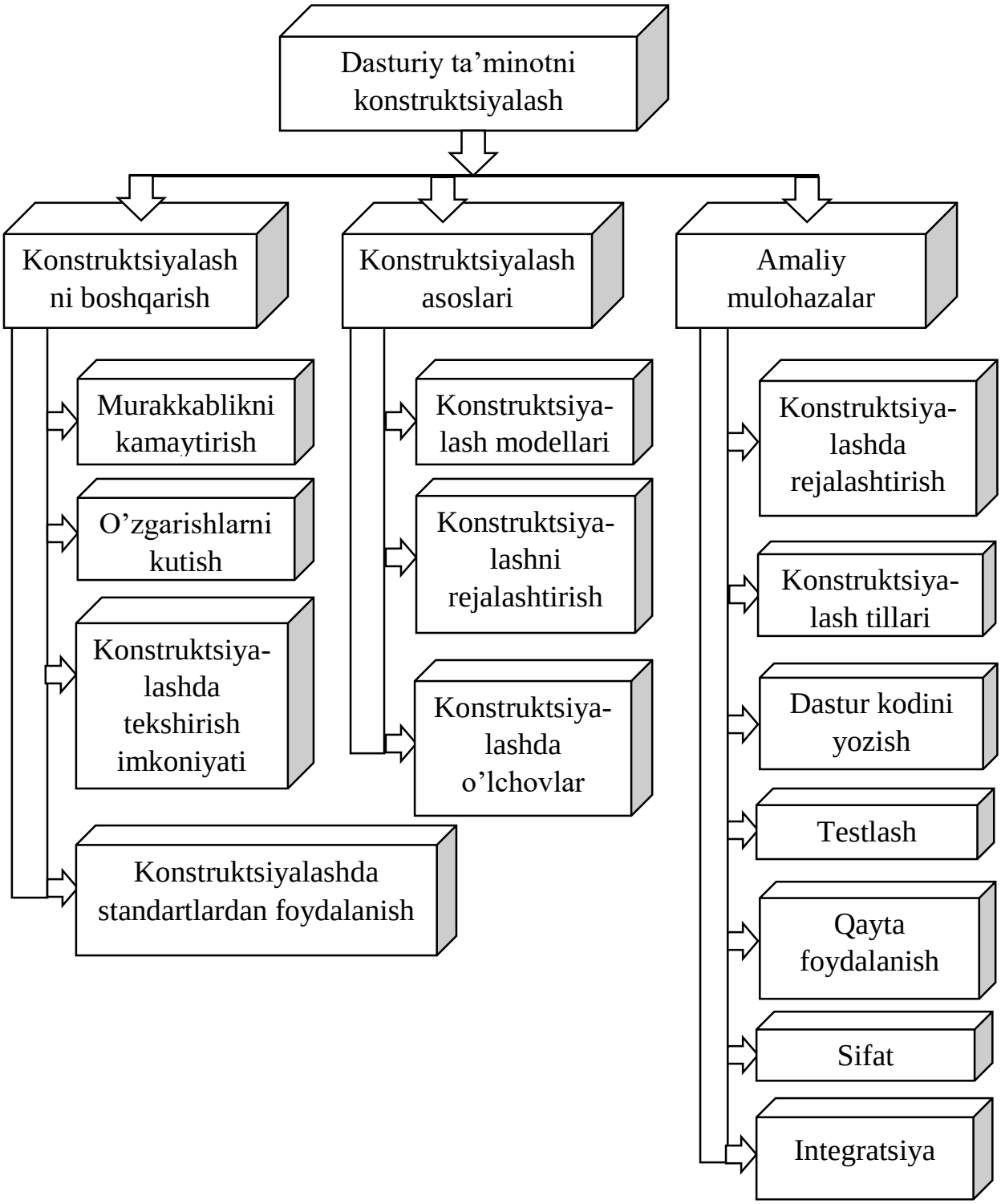
chegaralarni aniqlash juda qiyin, chunki ularning barchasi hayot tsikli jarayonlarining yagona kompleksiga bog’langan. Loyihalash jarayoni bu quyi darajadagi konstruksiyalash va kodlashni o‘z ichiga olgan dasturiy ta’minotni ishlab chiqish jarayonidir. Quyi darajadagi konstruktsiyalash - bu dasturiy ta’minot arxitekturasini yanada batafsil ishlab chiqish: ob’ektga yo‘naltirilgan dasturlashda sinflarni konstruksiyalash, ma’lumotlar bazasini boshqarish tizimida ma’lumotlar bazasi tuzilishini yaratish (ma’lumotlar bazasini boshqarish tizimi), Web dasturlari va tarkibiy qismlarini tashkil qilish va boshqalar.

Kodlash - bu dastur kodini yozish tartibi. Bu yuqori va quyi darajadagi konstruksiya arxitekturasi asosida dasturni ishlab chiqishdir. Ba’zi konstruksiyalarda, agar maqsadga muvofiq bo‘lsa, konstruktsiyalash bosqichi konstruksiyalash jarayoni bilan birlashtiriladi. Konstruktsiyalash jarayoni va ishlab chiqish jarayonlari ishlab chiqilayotgan dasturlarning turli toifalari uchun farq qiladi, eng keng tarqalganlari orasida quyidagi ishlab chiqish turlari mavjud.

Ma’lumotlar bazasini yaratish. Ma’lumotlar bazalari dasturlarning alohida toifasi hisoblanadi. Ma’lumotlar bazasini ishlab chiqish ko‘p hollarda ma’lumotlar bazasida saqlanadigan ma’lumotni boshqaradigan dastur turlaridan birini ishlab chiqish bilan bevosita bog’liqdir. Ma’lumotlar bazasini dasturlash bilan ko‘pincha alohida dasturchilar guruhi shug’ullanadilar. Strukturali dasturlash asosida dasturlarni ishlab chiqish. Strukturali dasturlash ma’lum bir dastur sinfi uchun bir qator dasturlash tillarida qo‘llaniladi: qurilma drayverlari, operatsion tizimlar va boshqalar.

Ob’ektga yo‘naltirilgan dasturlashga asoslangan dasturlarni ishlab chiqish. Ob’ektga yo‘naltirilgan tillar ko‘p sonli dasturlarda qo‘llaniladi. Ushbu dasturlarni ishlab chiqishda asosiy vazifalardan biri bu sinf ierarxiyasini konstruksiyalashdir. Sinflarni konstruksiyalashda xatolar dasturni takomillashtirishga imkon bermaydi, bu ishlab chiqish vaqtining kechikishiga, narxning oshishiga va boshqa salbiy oqibatlarga olib kelishi mumkin. Web-ilovalarni ishlab chiqish. Web-ilovalar o‘zlarining rivojlanish xususiyatlariga ega bo‘lgan dasturiy mahsulotlarning yana bir katta toifasiga tegishli, masalan, web-brauzerlar (appletlar) uchun dasturlarni ishlab chiqish ushbu toifadagi dasturlar orasida juda keng tarqalgan. Tegishli dasturiy fanlardan ushbu bilim sohasining eng yaqin va tabiiy aloqasi informatika fani bilan bog’liq. Odatda, ularda algoritmlarni qurish va ulardan foydalanish masalalari ko‘rib chiqiladi. Va nihoyat, konstruktsiyalash konstruksiyani boshqarish faoliyatiga ham

tegishlidir. Dasturiy ta’minotni konstruksiyalash vositalariga dasturlash va konstruktsiyalash tillari, shuningdek dasturlash usullari va instrumental tizimlar (kompilyatorlar, MBBT, hisobot generatorlari, versiyalarni boshqarish tizimlari, konfiguratsiya, test va boshqalar) kiradi.



1.1-rasm. Dasturiy ta’minotni konstruksiyalash muhiti[2].

[2] Романов А.А. Конструирование программного обеспечения: учебное пособие. – Ульяновск: УлГТУ, 2016.

Dasturiy ta’minotni konstruktsiyalash (“Software Construction”) bilim sohasi quyidagi bo‘limlarni o‘z ichiga oladi:

- murakkablikni kamaytirish (*Reduction* in Complexity),
- uslubdan og’ishning oldini olish (Anticipation of *Diversity*),
- sinovlarni tuzilmalashtirish (Structuring for Validation),
- tashqi standartlardan foydalanish (Use of External Standards).

Murakkablikni kamaytirish - bu konstruktsiyalashdagi murakkablikni minimallashtirish va alohida qismlarga bo‘lish. Murakkablikni minimallashtirish, ijrochilarning murakkab tuzilmalarni va uzoq vaqt davomida katta hajmdagi ma’lumotlarni qayta ishlash qobiliyatining cheklanganligi bilan belgilanadi.

Murakkablikni minimallashtirishga, xususan, konstruktsiyalash jarayonida modullar va boshqa sodda elementlardan foydalanish, shuningdek, standartlarning tavsiyalari yordamida erishiladi. Dasturiy ta’minot konstruktsiyalashning murakkabligini kamaytirish uchun oddiy va oson o‘qiladigan kodni yaratish orqali erishiladi.

Bunda kodning samaradorligini oshirish, kodning sinov qulayligini ta’minlash, kodning ishlashi va belgilangan mezonlarga javob berishiga e’tibor beriladi. Bu konstruksiyaning funktsionalligi, xususiyatlari va cheklovlariga ta’sir qiladi. Murakkablikni kamaytirish zarurati konstruktsiyalashning barcha jihatlariga ta’sir qiladi va ayniqsa dasturiy komponentlarning konstruktsiyalash natijalarini tekshirish va sinovdan o‘tkazish uchun juda muhimdir.

Murakkablikni lokalizatsiya qilish - bu ob’ektga yo‘naltirilgan yondashuvdan foydalangan holda konstruktsiyalashning usuli bo‘lib, bu ob’ektlarning interfeysini cheklaydi, ularning o‘zaro ta’sirini soddalashtiradi, shuningdek ob’ektlarning to‘g’riligini va ular o‘rtasidagi munosabatlarni tekshirishni soddalashtiradi. Lokallashtirish kodda aniqlangan xatolarga o‘zgartirishlarni osonlashtiradi.

1.2. Dasturlash konstruktorlik ishi sifatida

Bugungi kunda axborot-kommunikatsiya sohasini rivojlantirishning asosi hisoblangan dasturiy mahsulotlarni ishlab chiqarish milliy iqtisodiyot rivojlanishining muhim sharti sifatida alohida ahamiyat kasb etmoqda. Hayotimizning har jabhasida faol qo‘llanilayotgan axborot almashinuv texnologiyalaridan samarali foydalanish ulardagi dasturiy ta’minotlarning o‘ziga xosligi, ommabopligi va innovatsion yangiligiga bog‘liq.

Dasturlash bu kompyuterga buyruqlar berishdir. Masalan, biror web-sayt tayyorlanmoqda va bu holatda uni tayyorlash uchun kompyuterga buyruqlarni yozish kerak. Faqat kompyuter o‘zbek tilini yoki boshqa bir horijiy tilni tushunmaydi. Shuning uchun unga o‘zi tushunadigan tilda ya’ni dasturlash tilida yozish kerak. Mana shu holat esa dasturlash deyiladi. Dasturchi bunday kodlarni yozib har-xil qurilamalar uchun ilovalar, web-saytlar, tizimlar va boshqa dasturlar tayyorlashi mumkin. Dasturchi ishlab chiqarayotgan mahsulotlardan sezish mumkinki, dasturchilarga hozirgi kunda ehtiyoj juda yuqori. Chunki dasturchilar tomonidan tayyorlanadigan mahsulotlar hozirgi kunning asosiy talablaridan biridir.

Dasturiy ta’minotni loyihalash va konstruksiyalash – dasturiy mahsulotni yaratishning eng ma’suliyatli bosqichlaridan biri. Bu bosqichda ishlab chiqilayotgan dasturiy ta’minotga asosiy talablar bayon etiladi. Funksiyalar va foydalanishning talablari qanchalik to‘liq aniqlanishidan konstruksiyalash jarayonini aniqlovchi asosiy qarorlar qanchalik to‘g’ri qabul qilinishida ishlanma bahosi va uning sifatiga bog’liq bo‘ladi.

Funksional belgiga ko‘ra dasturiy mahsulotlarni sinflash. Har bir dasturiy mahsulot ma’lum funksiyalarni bajarish uchun mo‘ljallangan vazifasiga ko‘ra barcha dasturiy mahsulotlarni uchta guruhga ajratish mumkin: tizimli, amaliy va gibrid. Dasturlash — kompyuterlar va boshqa mikroprotsessorli kompyuterlar uchun dasturlar tuzish, sinash va o‘zgartirish jarayonidan iborat. Odatda dasturlash yuqori saviyali dasturlash tillari (Delphi, Java, C++, Python) vositasida amalga oshiriladi. Bu dasturlash tillarining semantikasi odam tiliga yaqinligi tufayli dastur tuzish jarayoni ancha oson kechadi. Dasturlash – bu:

1) kompyuterlarda masalalarni yechish hamda ularda har xil aqliy mehnat turlarini bajarish nazariyasi va usullarini ishlab chiqish bilan shug‘ullanadigan fan;

2) algoritmlar nazariyasining amaliy bo‘limi;

3) insonning kompyuter bilan aloqa qilish vositasi. Asosiy vazifalaridan biri kompyuterlar uchun programma (dastur) tuzish usullari, ularni tekshirish va takomillashtirishdan iborat. Yechilishi lozim bo‘lgan masala algoritmi dasturlashda „kompyuter tili“ga o‘tkaziladi.

Dasturlash bevosita dasturlash va avtomatik dasturlashga bo‘linadi. Bevosita Dasturlashda programmaning umumiy sxemasini ishlab chiqishdan kodlash va mashi-naga kiritishgacha bo‘lgan barcha ishni programmachi bajaradi. Avtomatik dasturlashda esa programmachi faqat

programma sxemasini tuzib, uni qisqartirilgan simvolik kurinishda yozadi, profamma tuzish va uni kodlash kabi texnikaviy ishlarni esa kompyuterning o‘zi maxsus dasturlash programmasi yordamida bajaradi. Dasturlash jarayoni, odatda, quyidagi bosqichlarga bo‘linadi: masalaning qo‘yilishi; masalaning algoritmik tavsifini tuzish; masalani yuqori darajadagi programma tilida dasturlash; masalani kompyuter tilida dasturlash.

Dasturlash tili programmalar tuzishning asosiy vositasidir. Bu tillar konkret kompyuter komandalari tizimsiga boglits bo‘lmasligi va iboralar struktu-rasi jihatidan umumiy xususiyatga ega bo‘lishi bilan boshqa tabiiy tillarga o‘xshab ketadi. Iboralar ikki turga — operatorlar hamda tavsiflarga bo‘linadi, ularning bir-biri bilan bog‘liqligi qavslar bilan, alohidaligi nukali vergul bilan ajratiladi. Operator tilning amal birligi bo‘lib, o‘z navbatida, o‘zgaruvchan kattalikka qiymat beruvchi operatorlar, shartga muvofiq tegishli hisoblash tarmog‘ini tanlovchi (shartli) operator va takroriy hisobni amalga oshiruvchi sikl ope-ratorlariga bo‘linadi. Tavsifda o‘zgaruvchan kattalik va boshqa belgilar xususiyatlari yoziladi. Biror xususiy masalani yechish uchun tuzilgan programmani simvolik ravishda funksional belgilash mumkin. Bunday belgilash va tavsif birgalikda kichik programma deb yuritiladi. Yangi programmalar tuzishda kichik programmalardan tayyor holda foydalanish mumkin.

Juda ko‘p dasturlash tillari (algol-60, q. Algol), muhandislik va ilmiy masalalarni yechish uchun fortran, iqtisodiy hisoblashlar uchun kobol, matematik modellar uchun si mula, tako-millashgan algol-68, PL/I yaratildi. Ularning har biri uchun shu tillarda ifodalangan masalalarga qarab kompyuter programmasini avtomatik tarzda qaytatuzuvchi translyatorlar mavjud. Taxminiy kompyuter tili ikkilik tizimdan ko‘ra yanada qulayroq simvollarda ifodalangan kompyuter komandalari terminlaridagi programmalar bo‘lib, bunda ko‘pincha, yuqori darajadagi til sifatida blok sxemalardan foydalaniladi.

Dasturlashning programma tuzilgandan keyingi yana bir asosiy bosqichi “tekshirish” (otladka) bo‘lib, bunda yo‘l qo‘yilgan xatolar topiladi va tuzatiladi. Programmalar kodlanadi va kompyuterga maxsus qurilma yordamida kiritiladi. Amaliyotda Dasturlashning yangi va tezkor usullari bor (2004); 2) matematik dasturlash — amaliy matematikaning bir bo‘limi; umumiy ma’noda — biron-bir funksiya fo(x) ning ekstremumini topish masalasi tushuniladi.

1.3. Dasturiy ta’minotni konstruktsiyalash bosqichlari

Boshlang’ich bosqichlarda bu jarayonni ko‘p jihatdan hamda ishlanma sifati va mehnat sarfini aniqlovchi prinsipial yechimlar qabul qilinishi mumkin:

- dasturiy ta’minot arxitekturasini tanlash;
- foydalanish interfeysi tipi va hujjatalr bilan ishlash texnologiyasini tanlash;
- ishlanmaga yondashuvni (tuzilmani yoki obyektni) tanlash;
- dasturlash tili va muhitini tanlash

Bu yechimlar nima loyihalashini qanday xarakteristikalarga ega, qanday vositalar bilan bajarilishini aniqlaydi. Dasturiy ta’minot arxitekturasini tanlash. Dasturiy ta’minot arxitekturasi deb uni tuzishning bazaviy konsepsiyalari jamlanmasiga aytiladi. Dasturiy ta’minot arxitekturasi yechilayotgan masalalar murakkabligi, ishlanayotgan dasturiy ta’minot universallik darajasi va uning birorta nusxasi bilan bir vaqtda ishlanayotgan foydalanuvchilar soni bilan aniqlanadi.

– bir foydalanuvchili arxitektura dasturiy ta’minot personal kompyuterda ishlayotgan bitta foydalanuvchiga mo‘ljallangan;

– ko‘p foydalanuvchili arxitektura lokal yoki global tarmoqda ishlashga mo‘ljallangan.

Bundan tashqari bir foydalanuvchi arxitektura doirasida:

- dasturlar;
- dastur paketlari;
- dasturiy komplekslar;
- dasturiy sistemalar.

Ko‘p foydalanuvchili arxitektura «mijoz-server» prinsipi bo‘yicha tuzilgan sistemalarni amalga oshiradi. Dastur deb kompyuterga jo‘natilgan konkret masalani yechish uchun bajarish zarur bo‘lgan amallar ketma-ketligini aniq tavsiflovchi ko‘rsatmalar majmuasiga aytiladi.

Tuzilishli yondashuvda qo‘yilgan masalani yechish jarayonida bir-birini chaqiruvchi ichki dasturlar iyerarxiyasidan obyektli yondashuvda – bajarish uchun maxsus sinflar ishlab chiqilgan bir-biri bilan xabar almashinuvchi obyektlar jamlanmasi. Dastur bu holda qism dasturlar standart kutubxonalari foydalanadigan alohida kompilisiyalanuvchi dasturiy birlikdan iborat bo‘lib, odatda o‘zining kutubxonalarini tashkil etmaydi. Bu arxitekturaning unchalik katta bo‘lmagan masalalarni yechishda foydalaniladigan eng sodda turi.

Dasturlar paketlari biror amaliy soha masalalarni yechadigan dasturlar jamlanmasi. Masalan, grafik dasturlar paketi, matematik dasturlar paketi. Bunday paket dasturlar o‘zaro ma’lum amaliy sohaga tegishliligi bilan bog’langan. Dasturlar paketlari har biri o‘zi zarur ma’lumotlar va natijalarni chiqaradigan alohida dasturlar jamlanmasini amalga oshiradi. Ular – dasturlar kutubxonasi.

Dasturiy komplekslar bitta amaliy soha murakkab masalalarni biror sinfini yechishni birgalikda ta’minlovchi dasturlar jamlanmasidan iborat. Bunday masalalarni yechish uchun kompleksning dasturlarini chaqirib bir nechta qism-masalalarni yechish zarurati paydo bo‘ladi. Dasturlar va dasturiy kompleksni tanlash maxsus dastur – murakkab bo‘lmagan interfeysini ta’minlovchi va biror ma’lumotli axborotni berishi mumkin bo‘lgan dispetcher bajaradi.

Dasturlar paketidan dasturiy komplekslar bilan farq qiladigan bir nechta dastur ketma-ket yoki siklik ravishda bitta masalani yechish uchun chaqirilishi mumkin, demak, bitta foydalanuvchi loyihasi doirasida berilgan ma’lumotlar va chaqiruvlar natijalarini saqlash maqsadga muvofiq. Bu holda dasturlar alohida yoki birgalikda kompilinerlanuvchi dasturiy birliklar kabi amalga oshiriladi, berilgan ma’lumotlar esa operativ xotirada yoki fayllarda saqlanadi.

Dasturiy komplekslardan farqli dasturiy sistemaga kiruvchi dasturlar umumiy ma’lumotlar orqali o‘zaro ta’sirda bo‘ladi. Dasturiy sistemalar odatda rivojlangan va ichki interfeyslarga ega, bu esa ularni puxta loyihalashni talab etadi. Ko‘p foydalanuvchili dasturiy sistemalar odatdagi dasturiy sistemalardan farqli dasturiy ta’minot alohida komponentlar o‘zaro ta’sirini tashkil etishi lozim, bu esa uni ishlab chiqishni yanada qiyinlashtiradi. Bunday dasturiy ta’minotni ishlab chiqish uchun maxsus texnologiyalar yoki platformalar, masalan, CORBA, COM, Java va h.k. texnologiyalar ishlatiladi.

Foydalanuvchi interfeys tipini tanlash. To‘rtta foydalanish interfeyslar tiplari farqlanadi:

– primitiv – ishning yagona ssenariysini, masalan, ma’lumotlarni kiritish – qayta ishlash – natijalarni chiqarishni amalga oshiradi;

– menyu – amallari iyerarxik tuzilmalarga shakllangan ishning ssenariylari to‘plamini, masalan, «qo‘yish», «faylni qo‘yish», «simvolni qo‘yish» va h.k.larni bajaradi;

– erkin novigasiyali – iyerarxiya darajalariga bog’lanmagan va ishning konkret qadamida mumkin bo‘lgan amallar to‘plamini aniqlashni

ko‘zda tutadigan ssenariylar to‘plamini amalga oshiradi, bu shaklning interfeyslari asosan Windows ilovalardan foydalanadi;

– to‘g’ridan-to‘g’ri manipulasiyalash – obyektlar ustidagi amallarda taqdim etilgan ssenariylar to‘plamini bajaradi, asosiy amallar sichqoncha bilan obyektlar piktogrammalarini siljitish bilan amalga oshiriladi, bu shaklli erkin novigasiyali
interfeysga muqobil bo‘lib Windows operasion tizimi o‘zining
interfeysida amalga oshirilgan.

Foydalanuvchi interfeys tipi ishlanmaning murakkabligi va mehnat sarfini belgilaydi. Dasturiy ta’minotni ishlab chiqishning obyektli – yo‘naltirilgan vizual muhitlari dasturlashga hodisaviy yondashuvdan foydalanadi va erkin novigasiyali interfeyslarni yaratishga mo‘ljallangan bo‘lib, bunday interfeyslarni ishlab chiqish mehnat sarfini jiddiy kamaytirdi va to‘g’ridan-to‘g’ri monipulyasiyalash interfeyslarni amalga oshirishni soddalashtirdi.

Interfeys tipini tanlash hujjatlar bilan ishlash texnologiyalarini tanlashni o‘z ichiga oladi. Ikkita texnologiya mavjud:

– bir hujjatli, u bir hujjat interfeys (SDI – Single Document Interface) ni ko‘zda tutadi;

– ko‘p hujjatli, u ko‘p hujjat interfeys (MDI – Multiple Document Interface) ni ko‘zda tutadi.

Ko‘p hujjatli texnologiya dasturiy ta’minot bir nechta hujjatlar bilan bir vaqtda ishlaganda, masalan, bir nechta matn yoki bir nechta tasvirlar bilan ishlaganda foydalaniladi. Bir hujjatli – agar bir nechta hujjatlar bilan bir vaqtda ishlash zarurati bo‘lmasa, hozirgi kutubxonalardan foydalanib ko‘p hujjatli interfeyslarni ishlatish mehnat sarfi birinchisiga qaraganda 3...5% yuqori.

Ishlanmaga yondashuvni tanlash. Agar erkin novigasiyali yoki to‘g’ridan-to‘g’ri manipulyasiyali interfeys tanlangan bo‘lsa, u holda hodisaviy dasturlash va obyektli yondashuvdan foydalaniladi, chunki zamonaviy vizual dasturlash muhitlari Visual C++, Delphi, Builder C++ va ularga o‘xshashlar kutubxona sinflari obyektlari ko‘rinishida interfeys komponentlarni taqdim etadi. Bunda predmet soha murakkabligiga bog’liq holda dasturiy ta’minot obyektlardan foydalanish orqali yoki sof prosedurali ravishda amalga oshirilishi mumkin, faqat bundan boshqa prinsipda tuzilgan Perl kabi internet-ilovalarni ishlab chiqishni maxsus tillardan foydalanish hollari mustasno. Primitiv interfeys va menyu tipidagi interfeys tuzilmali obyektli yondashuvlar bilan muvofiqlashadi. Shuning uchun yondashuvni tanlash qo‘shimcha axborotdan foydalanish

bilan amalga oshadi. Obyektli yondashuv juda katta dasturiy sistemalar (universal dasturlash tilida 100000 dan ko‘p bo‘lgan operatorlar mavjud) ini va predmet soha obyekt tuzilishi aniq ifodalangan hollarda samarali. Shuningdek u dasturiy ta’minot samaradorligiga qattiq cheklashlarda ham foydalaniladi.

Dasturlash tilini tanlash

Til quyidagilar bilan aniqlanish mumkin:

– ishlanmani olib boruvchi tashkilot, masalan, agar firma C++ Builder ning lisenziyali variantiga ega bo‘lsa, u holda u berilgan muhitda ishlanma olib boradi;

– dasturchi tomonidan, u imkoni boricha yaxshi tanish tildan foydalanishga harakat qiladi;

– turib qolgan .... bilan («barcha ishlanmalar C++ da yoki Java da bajarilishi lozim») va h.k

Dasturlash tillarini quyidagi guruhlarga ajratish mumkin:

– yuqori darajali universal tillar;
– dasturiy ta’minot ishlab chiqaruvchisini maxsus tillari;
– foydalanuvchining maxsus tillari;
– past darajali tillar.

Yuqori darajali dasturlash tillar gurihida hozirgi paytda karvonboshi C (S++ bilan barcha) til hisoblanadi. Haqiqatdan C va C++ lar turli talqinlari qator juda muhim afzalliklarga ega:

– ko‘p platformalilik – hozirgi paytda foydalanilayotgan barcha platformalar uchun;
– C va C++ tilidan kompilyatorlar mavjud;
– asosiy tuzilmali algoritmik konstruksiyalarni amalga oshiruvchi operatorlar mavjudligi (shartli qayta ishlash, sikllarning barcha turlari);
– tezkor xotira manzillaridan foydalanib past (sistema) darajada dasturlash imkoniyati;
– qism dasturlar va sinflarning o‘lkan kutubxonalari.

Bularning barchasi C va C++ ni operasion tizimlarini yaratish uchun ishlatiladigan asosiy tillarga aylantiradi, bu esa o‘z navbatida ular uchun qo‘shimcha reklama bo‘lib xizmat qiladi. Lekin C va C++ jiddiy kamchiliklarga ham ega:

– ma’lumotlarning to‘laqonli ichki to‘qima tuzilmali tiplari mavjud emasligi;
– sintaktik bir qiymatlilikning mavjud emasligi, bu esa kompelyatorga dasturning to‘g’riligini nazorat qilishga imkon bermaydi;

– qism dasturda uzatilayotgan parametrlarning cheklangan nazorati, bu esa faqat dasturni sozlash jarayonida aniqlanadi;

C va C++ ga muqobil bo‘lib, puxta sintaksisi tufayli kompilyatorlari sintaksis xatolardan tashqari semantik xatolarni ham aniqlaydigan Pascal tili hisoblanadi. Delphi muhitida foydalanilgan Object Pascal talqini turli katta ishlanmalar olib borishni soddalashtiruvchi jumladan, ma’lumotlar omborida foydalanishni talab etuvchi sinflarning kasbiy kutubxonalari bilan birgalikda Delphi ni Windows ilovalarini yaratish uchun yetarlicha samarali muhit bo‘lishini ta’minlaydi. Universal tillar guruhiga Basic, Modula, Ada va boshqalar ham tegishli. Ular ham
o‘z xususiyatlariga mos ravishda o‘z qo‘llanish sohasiga ega. Ishlab chiqaruvchi maxsus tillari dasturiy ta’minot konkret tillarini yaratish uchun ishlatiladi. Ularga:

- ma’lumotlar bazasi tillari;
- to‘r ilovalarini yaratish uchun tillar;
- sun’iy intellekt sistemalari yaratish tillari

Bu tillar maxsus kurslarda o‘rganiladi. Foydalanuvchining maxsuslashgan tillari odatda foydalanuvchi kasbiy muhitlari qismi hisoblanadi, tor yo‘nalishga egaligi bilan xarakterlanadi va dasturiy ta’minot ishlab chiqaruvchilari tomonidan ishlatilmaydi. Past darajali tillar mashina buyruqlari darajasida dasturlarni amalga oshirishga imkon beradi. Bunda vaqt nuqtai nazaridan, dasturning zarur xotira hajmi nuqtai nazaridan ham eng optimallari olinadi. Lekin bu tillar katta dasturlar dasturiy tizimlar yaratish uchun umuman yaroqsiz. Asosiy sabab – ishlab chiqaruvchi amal qiladigan abstraksiyalar past saviyasidir, bundan ishlanmaga katta vaqt sarflashga yo‘l qo‘yilmaydi. Bu tillar tuzilmali dasturlash prinsiplarini qo‘llamaydi, bu esa ishlab chiqilayotgan dasturlash texnologiyasini yomonlashtiradi.

Bevosita texnik vositalar bilan, masalan, drayverlar bilan o‘zaro ta’sirda bo‘lgan nisbatan oddiy dasturlarni yozishda foydalaniladi, chunki bu holda mos qurilmani qat’iy sozlashga to‘g’ri keladi, yuqori darajali dasturiy tillar afzalliklari unchalik sezilmaydi. Yuqori darajali tillardagi dasturlarga qo‘yishlar ko‘rinishida masalan, ko‘p sonli takrorlanishlarga ega sikllarda ma’lumotlarni almashtirishni tezlashtirish uchun.

Dashturlash muhiti deb, dasturlarni yozish va saqlash jarayonini soddalashtiradigan kompilyatorga ichki qurilgan maxsus matn muharriri, komponovshik sozlovchi, ma’lumotlar sistemasi va boshqa dasturlarni o‘z ichiga olgan dasturiy kompleksga aytiladi. Vizual dasturlash muhitlarida dasturchi komponentlar maxsus kutubxonalaridan ba’zi

kodlar dasturiga vizual ulanish imkoniyatiga ega bo‘ladi, bu esa obyektli yo‘naltirilgan dasturlash rivoji tufayli sodir bo‘ldi. Eng ko‘p foydalanadigan dasturlash muhitlariga Borland (Inprise Corporation) firmasini Delphi, C++ Builder, Microsoft firmasini Visual C++, Visual Basic, IBM firmasini Visual Ada vizual muhitlari kiradi.

Bu firmalarni asosiy vizual muhitlari Delphi, C++ Builder va Visual C++ lar orasida muhim farqlar mavjud: Microsoft firmasi vizual muhitlari «Windows osti» dasturini pastroq saviyasini ta’minlaydi. Bu ularning ham kamchiligi va afzalligidir. Afzalligi – nostandart vaziyat paydo bo‘lish ehtimoli pasayadi, ya’ni kompyuterlar kutubxonalari ishlab chiqaruvchilari ko‘zda tutmagan vaziyat, kamchiligi – bu dasturchiga ko‘p ish yuklaydi, bundan Delphi yoki C++Builder bilan ishlovchi dasturchi esa ozod. Loyihalash ixtiyoriy texnologiyaning real qo‘llash loyiha barcha qatnashchilari rioya qilish lozim bo‘lgan dastur standartlarni shakllantirish yoki tanlashni talab etadi:

- loyihalash standarti;
- loyiha hujjatni rasmiylashtirish standarti;
- foydalanuvchi interfeysi standarti.

Loyihalash standarti quyidagilarni aniqlashi lozim:

– loyihalash har bir bosqichida zarur modellar majmuasi va ularni detallashtirish darajasi;

– diagrammalarda loyiha yechimlarni qayd etish qoidalarini, shu jumladan atamalar bo‘yicha obyektlarni nomlash va shartnomalar qoidalari, barcha obyektlar uchun atributlar majmuasi va ularni har bir bosqichida to‘ldirish qoidalari, obyektlar shakli va o‘lchamlarga talablar kiritilgan diagrammalarni rasmiylashtirish qoidalari;

– operasion tizim va foydalanilayotgan CASE – vositalarni sozlashni o‘z ichiga olgan ishlab chiqaruvchilar ish joylari konfigurasilariga talablar;

– loyiha ustida birgalikda ishlashni ta’minlash mexanizmi, shu jumladan loyiha qism sistemalarini integrasiyasi va loyiha yechimlarni qarama-qarshilikka tahlil etish qoidalari.

Loyiha hujjatlarini rasmiylashtirish standarti:

- har bir bosqichda hujjatlar komplektligi, tarkibi va tuzilishi;
- uning mazmuni va rasmiylashtirilishiga talablar;
- hujjatni tayyorlash, qarab chiqish, kelishtirish va tasdiqlash qoidalari.

Foydalanuvchi interfeysi standarti

– ekranlarini (shriftlar va ranglar), oynalar tarkibi, joylashishi va boshqaruv elementlarini rasmiylashtirish qoidalari;
– klaviatura va sichqonchadan foydalanish qoidalari;
– yordamchi mtnlarni rasmiylashtirish qoidalari;
– standart xabarlar ro‘yxati;
– foydalanuvchi reaksiyasini qayta ishlash qoidalarini aniqlashi lozim.

Barcha yuqorida keltirilgan loyihaviy yechimlar ishlanmasi mehnat sarfi va murakkabligi jiddiy ta’sir ko‘rsatadi. Ularni qabul qilgandan so‘nggina loyihalanayotgan dasturiy ta’minotning talablar tahliliga va o‘ziga xos xususiyatlarini ishlab chiqishga o‘tish lozim.

1-bob bo‘yicha xulosalar

Dasturiy ta’minotni konstruktsiyalash atamasi kodlashtirish, tekshirish, modulli sinov, integratsiya tekshiruvi va nosozliklarni tuzatish kabi jarayonlardan iborat dasturiy tizimni batafsil ishlab chiqishni tavsiflaydi. Ushbu bilim sohasi boshqa sohalar bilan bog’liq. Eng kuchli bog’liqlik dasturiy ta’minotni konstruksiyalash (Software Design) va dasturiy ta’minotni sinash (Software Testing) sohalari bilan mavjud. Buning sababi, dasturiy ta’minotni konstruktsiyalash jarayonining o‘zi konstruksiyalash va sinov faoliyatining muhim jihatlariga tegishlidir. Bundan tashqari, konstruktsiyalash konstruksiyalash va sinov natijalariga asoslanadi. Loyihalash jarayoni bu quyi darajadagi konstruksiyalash va kodlashni o‘z ichiga olgan dasturiy ta’minotni ishlab chiqish jarayonidir. Quyi darajadagi konstruktsiyalash - bu dasturiy ta’minot arxitekturasini yanada batafsil ishlab chiqishdir.

Dasturiy ta’minotni loyihalash va konstruksiyalash – dasturiy mahsulotni yaratishning eng ma’suliyatli bosqichlaridan biri. Bu bosqichda ishlab chiqilayotgan dasturiy ta’minotga asosiy talablar bayon etiladi. Funksiyalar va foydalanishning talablari qanchalik to‘liq aniqlanishidan konstruksiyalash jarayonini aniqlovchi asosiy qarorlar qanchalik to‘g’ri qabul qilinishida ishlanma bahosi va uning sifatiga bog’liq bo‘ladi.

Mamlakatimizda dasturiy ta’minot sanoatini shakllantirish va rivojlantirish uchun zarur shart-sharoitlar, sohaga daxldor mustahkam qonunchilik bazasini yaratishga e’tibor qaratilmoqda. Hozirgacha 10 dan ziyod qonun, Prezident va Vazirlar Mahkamasining qator farmon hamda qarorlari qabul qilindi.

1-bob bo‘yicha nazorat savollari

1. “Dasturiy ta’minot qurilmasi va evolyusiyasi” fanini o‘qitishdan maqsad va vazifasi nimalardan iborat?
2. Dasturiy ta'minotdan foydalanish qulayligi haqida nimalar bilasiz?
3. Dasturiy ta'minotning barqarorligi nimalardan iborat?
4. Software so‘zining ma’nosini aytib bering.
5. Shareware so‘zining ma’nosini aytib bering.
6. Freeware so‘zining ma’nosini aytib bering.
7. “Free and Open Source Software” iborasi nimani anglatadi ?
8. Kompyuterga dasturiy ta’minotni o‘rnatish jarayoni qanday nomlanadi ?
9. Dasturiy ta’minot qanday guruhlarga bo‘linadi?
10. Dasturiy ta'minotning harakatchanligi deganda nimani tushunasiz?

2-BOB. DASTURIY TA’MINOTNI KONSTRUKSIYALASH ASOSLARI

2.1. Dasturiy ta’minotni konstruksiyalashning ahamiyati

Dasturiy ta’minotni konstruktsiyalash - bu dasturchiga dasturiy ta’minotni kodlash va amalga oshirishda yordam beradigan, foydalanuvchi talablarini kerakli shaklga o‘tkazish jarayoni. Foydalanuvchi talablarini aniqlash uchun, kodlash va kiritish uchun dasturiy ta’minotga nisbatan aniqroq va batafsil talablarni talab qiladigan SRS (Software Requirement Specification) hujjati yaratiladi.

Bu jarayonning natijasi to‘g’ridan - to‘g’ri dasturlash tillarida yozilishi mumkin. Dasturiy ta’minotni konstruktsiyalash - bu SDLC (Dasturiy ta’minotni loyihalashtirish hayotiy tsikli) ning birinchi bosqichi bo‘lib, u kontsentratsiyani muammoli sohadan yechim maydoniga o‘tkazadi. U SRSda ko‘rsatilgan talablarni qanday bajarish kerakligini aniqlashga harakat qiladi.

Dasturiy ta’minotni konstruktsiyalash darajalari

Dasturiy ta’minotni konstruktsiyalash 3 darajali natijalarni beradi:

– **Arxitekturaviy konstruktsiyalash**. Arxitekturaviy konstruktsiyalash - tizimning eng yuqori abstract versiyasi. Bu dasturiy ta’minotni bir -biri bilan o‘zaro ta’sir qiladigan ko‘plab komponentlardan iborat tizim sifatida belgilaydi. Bu darajada konstruktorlar taklif qilinayotgan soha haqida tasavvurga ega bo‘ladilar.

– **Yuqori darajali konstruktsiyalash**. Yuqori darajali konstruktsiyalash arxitektura konstruktsiyaning yagona tizimini ko‘p komponentli kichik tizimlar va modullarning kamroq abstrakt ko‘rinishiga ajratadi va ularning bir-biri bilan o‘zaro ta’sirini ochib beradi. Yuqori darajali konstruktsiyalash tizimni barcha komponentlari bilan birgalikda modullar ko‘rinishida qanday amalga oshirilishiga qaratilgan. U har bir kichik tizimning modulli tuzilishini, ularning o‘zaro munosabatlari va o‘zaro ta’sirini o‘rganadi.

– **Batafsil konstruktsiyalash.** Batafsil konstruktsiyalash oldingi ikkita konstruktsiyalashdagi tizim va uning quyi tizimlari sifatida ko‘riladigan qo‘shimchalar bilan bog’liq. Bu modullar va ularning qo‘shimchalari haqida batafsilroq ma’lumot beradi. U boshqa modullar bilan aloqa o‘rnatish uchun har bir modulning mantiqiy tuzilishini va ularning interfeyslarini belgilaydi.

Modullashtirish

Modullashtirish - bu dasturiy ta’minot tizimini vazifalarni mustaqil bajarishga qodir bo‘lgan bir nechta diskret va mustaqil modullarga bo‘lish usuli. Bu modullar barcha dasturlar uchun asosiy konstruktsiyalash bloklari vazifasini o‘tashi mumkin. Konstruktorlar modullarni shunday konstruktsiyalashga harakat qilishadiki, bunda ularni alohida va mustaqil ravishda ishlab chiqilishi va bajarilishi mumkin bo‘ladi.

Modulli konstruktsiyalash tasodifan bo‘linish qoidalariga bo‘ysunadi va muammoni hal qilish strategiyasini yengadi, chunki dasturiy ta’minotni modulli konstruktsiyalashning ko‘plab afzalliklari bor.

Modullashtirishning afzalliklari:

- Kichikroq qismlarni saqlash osonroq;
- Dasturni funktsional jihatlariga qarab ajratish mumkin;
- Istalgan darajadagi abstraktsiyani dasturga kiritish mumkin;
- Ko‘p aloqali komponentlar qayta ishlatilishi mumkin;
- Bir vaqtning o‘zida bir necha komponentlar bajarilishi mumkin;
- Xavfsizlik jihatidan ta’minlangan.

Dasturiy modulni ishlab chiqishda quyidagi tartibga rioya qilish tavsiya etiladi:

- Modul spetsifikatsiyasini o‘rganish va tekshirish, dasturlash tilini tanlash;
- Algoritm va ma’lumotlar tuzilishini tanlash;
- Modulni dasturlash (kodlash);
- Modul matnini sayqallash;
- Modulni tekshirish;
- Modulni kompilyatsiya qilish.

Dasturiy ta’minot modulini ishlab chiqishdagi birinchi qadam, asosan, dastur tuzilmasini pastdan tutashgan boshqarishdir: modulning spetsifikatsiyasini o‘rganib chiquvchi, uning o‘zi uchun tushunarli ekanligiga va ushbu modulni ishlab chiqish uchun etarli ekanligiga ishonch hosil qilishi kerak. Ushbu qadam oxirida dasturlash tili tanlanadi: garchi dasturlash tili allaqachon butun dasturiy ta’minot tizimi uchun oldindan belgilab qo‘yilgan bo‘lsa-da, shunga qaramay ba’zi hollarda (agar dasturlash tizimi bunga yo‘l qo‘ysa) ushbu modulni amalga oshirish uchun ko‘proq mos keladigan boshqa tilni tanlash mumkin (masalan, yig’ilish tili).

Dasturiy modulni ishlab chiqishning ikkinchi bosqichida, qo‘yilgan muammoni hal qilish uchun yoki unga yaqin bo‘lgan algoritmlar

allaqachon ma’lum yoki yo‘qligini aniqlash kerak. Agar mos algoritm topilsa, undan foydalanish maqsadga muvofiqdir. Modul o‘z funktsiyalarini bajarganda foydalaniladigan ma’lumotlarning mos tuzilmalarini tanlash asosan ishlab chiqilayotgan modulning mantiqiy va sifat ko‘rsatkichlarini oldindan belgilab beradi, shuning uchun uni juda mas’uliyatli qaror deb hisoblash kerak.

Uchinchi bosqichda modul matni tanlangan dasturlash tilida tuziladi. Modul spetsifikatsiyasida ko‘rsatilgan funktsiyalarni amalga oshirishda hisobga olinishi kerak bo‘lgan har xil tafsilotlarning ko‘pligi osongina juda ko‘p xato va noaniqliklarni o‘z ichiga olgan juda chalkash matn yaratilishiga olib kelishi mumkin. Bunday modulda xatolarni topish va unga kerakli o‘zgarishlarni kiritish juda ko‘p vaqt talab qilishi mumkin. Shuning uchun modul matnini yaratish uchun texnologik jihatdan asosli va amalda tasdiqlangan dasturlash intizomidan foydalanish juda muhimdir. Dijkstra birinchi marta bunga e’tiborni qaratdi, tuzilgan dasturlashning asosiy tamoyillarini shakllantirdi va asoslab berdi. Amaliyotda keng qo‘llaniladigan ko‘plab dasturiy fanlar ushbu printsiplarga asoslanadi. Eng keng tarqalgan intizom - bu bosqichma-bosqich takomillashtirish, bu 8.2 va 8.3-bo‘limlarda batafsil muhokama qilinadi.

Modulni rivojlantirishning navbatdagi bosqichi dasturiy ta’minot tizimining sifat spetsifikatsiyasiga muvofiq modul matnini to‘liq shaklga keltirish bilan bog’liq. Modulni dasturlashda ishlab chiquvchi modul funktsiyalarini to‘g’ri bajarilishiga e’tibor qaratadi, to‘liq bo‘lmagan izohlarni qoldiradi va dastur uslubiga qo‘yiladigan talablarning ayrim buzilishlariga yo‘l qo‘yadi. Modul matnini sayqallashda u talab qilinadigan sifatli ibtidoiylikni ta’minlash uchun matndagi mavjud sharhlarni tahrir qilishi va qo‘shimcha izohlarni kiritishi kerak. Xuddi shu maqsadda dastur matni uslubiy talablarga javob beradigan tarzda tahrirlangan.

Modulni tekshirish bosqichi - bu modulning ichki mantig’ining xatolarini tekshirishdan oldin (kompyuterda bajarilishidan foydalangan holda) qo‘lda tekshirish, muhokama qilingan dasturlash texnologiyasi uchun ishlab chiqilgan umumiy printsipni amalga oshirish, dasturiy ta’minotni ishlab chiqishning har bir bosqichida qabul qilingan qarorlarni nazorat qilish zarurligi to‘g’risida. Va nihoyat, modulni ishlab chiqishdagi so‘nggi qadam modulni tekshirishni yakunlash (kompilyator yordamida) va modulni nosozliklarni tuzatish jarayoniga o‘tishni anglatadi.

Tarkibiy dasturlash

Modulni dasturlashda dastur nafaqat kompyuter uchun, balki odam uchun ham tushunarli bo‘lishi kerakligini yodda tutish kerak: ham modulni ishlab chiquvchi, ham modulni tekshiradigan shaxslar, shuningdek modulni xatolarini tekshirish uchun testlar tayyorlaydiganlar va modulga kerakli o‘zgarishlarni kiritgan DT texnik xizmatchilari modul mantig’ini qayta-qayta tahlil qiladi. Zamonaviy dasturlash tillarida ushbu mantiqni xohlagancha chalkashtirib yuborish uchun etarli vositalar mavjud, shu bilan modulni odamlar tushunishi qiyinlashadi va natijada uni ishonchsiz yoki saqlashni qiyinlashtiradi. Shuning uchun tegishli til vositalarini tanlash va aniq dasturlash intizomiga rioya qilish choralarini ko‘rish zarur. Shu munosabat bilan Dijkstra dasturni mantiqiy tushunchalarini sezilarli darajada oshirishi mumkin bo‘lgan bir necha turdagi boshqaruv tuzilmalari (tuzilmalari) ning tarkibi sifatida dastur tuzishni taklif qildi. Faqat shunday konstruktsiyalardan foydalangan holda dasturlash tizimli chaqirildi.

Tarkibiy dasturlashning asosiy konstruktsiyalari quyidagilardan iborat: amal qilish, tarmoqlash va takrorlash. Ushbu konstruktsiyalarning tarkibiy qismlari umumlashtirilgan operatorlar (ishlov berish tugunlari) va shart (predikat). Umumlashtirilgan operator yoki foydalaniladigan dasturlash tilining oddiy operatori (tayinlash, kiritish, chiqish, protsedurani chaqirish operatorlari) yoki dastur bo‘lagi bo‘lishi mumkin, bu asosiy tuzilgan dasturlashni boshqarish konstruktsiyalarining tarkibi. Ushbu dizaynlarning har biri boshqarish uchun faqat bitta kirish va bitta chiqishga ega bo‘lishi juda muhimdir. Shunday qilib, umumlashtirilgan operatorda faqat bitta kirish va bitta chiqish mavjud. Ushbu konstruktsiyalar allaqachon matematik ob’ektlar bo‘lishi juda muhimdir (bu mohiyatan tizimli dasturlash muvaffaqiyatining sababini tushuntiradi). Har bir tuzilmagan dastur uchun funktsional ekvivalenti (ya’ni bir xil masalani echish) tuzilgan dasturni qurish mumkinligi isbotlangan. Tuzilmaviy dasturlar uchun siz ba’zi xususiyatlarni matematik ravishda isbotlashingiz mumkin, bu dasturdagi ba’zi xatolarni aniqlashga imkon beradi. Ushbu masalaga alohida ma’ruza bag’ishlanadi.

Tarkibiy dasturlash ba’zan "GO TO-less dasturlash" deb nomlanadi. Biroq, gap GO TO bayonotida emas, balki uning tartibsiz ishlatilishida. Ko‘pincha, ba’zi dasturlash tillarida tuzilgan dasturlashni amalga oshirishda, tuzilgan konstruktsiyalarni amalga oshirish uchun o‘tish operatori (GO TO) ishlatiladi, bu esa tuzilgan dasturlash tamoyillarini buzmaydi. Dasturni chalkashtirib yuboradigan aynan "tizimli bo‘lmagan"

operatorlar, ayniqsa yuqoridagi modul matnida joylashgan operatorga o‘tish (oldinroq) o‘tish operatori bajarilmoqda. Shunga qaramay, ba’zi bir oddiy holatlarda filial operatoridan qochishga urinish juda noqulay tuzilgan dasturlarga olib kelishi mumkin, bu ularning ravshanligini yaxshilamaydi va modul matnida qo‘shimcha xatolar xavfini o‘z ichiga oladi. Shuning uchun dasturning ravshanligi evaziga emas, iloji bo‘lsa, o‘tish operatoridan foydalanishni oldini olish tavsiya qilinishi mumkin.

O‘tish operatoridan foydalanishning foydali holatlari orasida tsikl yoki protseduradan ma’lum bir tsikl yoki berilgan protsedura ishini "erta" tugatadigan, ya’ni ba’zi bir tarkibiy bo‘linmaning (umumlashtirilgan operator) ishini tugatadigan va shu bilan dasturning tuzilishini faqat mahalliy darajada buzadigan maxsus shart bilan chiqish kiradi. Katta qiyinchiliklar (va strukturaning murakkablashishi) paydo bo‘ladigan istisno (ko‘pincha noto‘g’ri) holatlarga reaktsiyani tizimli ravishda amalga oshirish natijasida yuzaga keladi, chunki bu nafaqat tarkibiy bo‘linmadan erta chiqib ketishni, balki ushbu vaziyatni zarur qayta ishlashni (chiqarib tashlashni) ham talab qiladi (masalan, mos diagnostika berish) ma ‘lumot). Istisno ishlovchisi dastur tuzilishining istalgan darajasida bo‘lishi mumkin va unga har xil quyi darajalardan kirish mumkin. Istisno holatlariga reaktsiyani quyidagi "tarkibiy bo‘lmagan" amalga oshirish texnologik nuqtai nazardan juda maqbuldir. Istisno ishlovchilari u yoki bu konstruktiv blokning oxiriga joylashtiriladi va har bir ishlov beruvchini shunday qilib dasturlash kerakki, u ishini tugatgandan so‘ng u joylashtirilgan struktura bo‘linmasidan chiqadi. Bunday ishlov beruvchiga qo‘ng’iroqni o‘tish operatori ushbu strukturaviy birlikdan (shu jumladan, har qanday ichki tuzilgan bo‘linmadan) amalga oshiradi.

2.2. Dasturiy ta’minotni konstruksiyalash bilan bog’liq vazifalar

Dasturiy ta’minotni konstruksiyalashning boshlang’ich bosqichlarida bu jarayonni ishlanma sifati va mehnat sarfini aniqlovchi prinsipial yechimlar qabul qilinishi mumkin:

- dasturiy ta’minot arxitekturasini tanlash;
- foydalanish interfeysi tipi va hujjatalr bilan ishlash texnologiyasini tanlash;
- ishlanmaga yondashuvni (tuzilmani yoki obyektni) tanlash;
- dasturlash tili va muhitini tanlash

Bu yechimlar nima konstruksiyalashini qanday xarakteristikalarga ega, qanday vositalar bilan bajarilishini aniqlaydi. Dasturiy ta’minot arxitekturasini tanlash. Dasturiy ta’minot arxitekturasi deb uni tuzishning bazaviy konsepsiyalari jamlanmasiga aytiladi. Dasturiy ta’minot arxitekturasi yechilayotgan masalalar murakkabligi, ishlanayotgan dasturiy ta’minot universallik darajasi va uning birorta nusxasi bilan bir vaqtda ishlanayotgan foydalanuvchilar soni bilan aniqlanadi.

- bir foydalanuvchili arxitektura dasturiy ta’minot personal kompyuterda ishlayotgan bitta foydalanuvchiga mo‘ljallangan;
- ko‘p foydalanuvchili arxitektura lokal yoki global tarmoqda ishlashga mo‘ljallangan.

Bundan tashqari bir foydalanuvchili arxitektura doirasida

- dasturlar;
- dastur paketlari;
- dasturiy komplekslar;
- dasturiy tizimlar.

Ko‘p foydalanuvchili arxitektura «mijoz-server» prinsipi bo‘yicha tuzilgan tizimlarni amalga oshiradi. Dastur deb kompyuterga jo‘natilgan konkret masalani yechish uchun bajarish zarur bo‘lgan amallar ketma-ketligini aniq tavsiflovchi ko‘rsatmalar majmuasiga aytiladi. Tuzilishli yondashuvda qo‘yilgan masalani yechish jarayonida bir-birini chaqiruvchi ichki dasturlar iyerarxiyasidan obyektli yondashuvda – bajarish uchun maxsus sinflar ishlab chiqilgan bir-biri bilan xabar almashinuvchi obyektlar jamlanmasi. Dastur bu holda qism dasturlar standart kutubxonalari foydalanadigan alohida kompilisiyalanuvchi dasturiy birlikdan iborat bo‘lib, odatda o‘zining kutubxonalarini tashkil etmaydi. Bu arxitekturaning unchalik katta bo‘lmagan masalalarni yechishda foydalaniladigan eng sodda turi.

Dasturlar paketlari biror amaliy soha masalalarni yechadigan dasturlar jamlanmasi. Masalan, grafik dasturlar paketi, matematik dasturlar paketi. Bunday paket dasturlar o‘zaro ma’lum amaliy sohaga tegishliligi bilan bog’langan. Dasturlar paketlari har biri o‘zi zarur ma’lumotlar va natijalarni chiqaradigan alohida dasturlar jamlanmasini amalga oshiradi. Ular – dasturlar kutubxonasidir.

Dasturiy komplekslar bitta amaliy soha murakkab masalalarni biror sinfini yechishni birgalikda ta’minlovchi dasturlar jamlanmasidan iborat. Bunday masalalarni yechish uchun kompleksning dasturlarini chaqirib bir nechta qism-masalalarni yechish zarurati paydo bo‘ladi. Dasturlar va dasturiy kompleksni tanlash maxsus dastur – murakkab bo‘lmagan

interfeysini ta’minlovchi va biror ma’lumotli axborotni berishi mumkin bo‘lgan dispetcher bajaradi. Dasturlar paketidan dasturiy komplekslar bilan farq qiladigan bir nechta dastur ketma-ket yoki siklik ravishda bitta masalani yechish uchun chaqirilishi mumkin, demak, bitta foydalanuvchi konstruksiyasi doirasida berilgan ma’lumotlar va chaqiruvlar natijalarini saqlash maqsadga muvofiq. Bu holda dasturlar alohida yoki birgalikda kompilinerlanuvchi dasturiy birliklar kabi amalga oshiriladi, berilgan ma’lumotlar esa operativ xotirada yoki fayllarda saqlanadi.

Dasturiy komplekslardan farqli dasturiy tizimga kiruvchi dasturlar umumiy ma’lumotlar orqali o‘zaro ta’sirda bo‘ladi. Dasturiy tizimlar odatda rivojlangan va ichki interfeyslarga ega, bu esa ularni puxta konsrtuksiyalashni talab etadi. Ko‘p foydalanuvchili dasturiy tizimlar odatdagi dasturiy tizimlardan farqli dasturiy ta’minot alohida komponentlar o‘zaro ta’sirini tashkil etishi lozim, bu esa uni ishlab chiqishni yanada qiyinlashtiradi. Bunday dasturiy ta’minotni ishlab chiqish uchun maxsus texnologiyalar yoki platformalar ishlatiladi. To‘rtta foydalanish interfeyslar turlari farqlanadi:

– primitiv – ishning yagona ssenariysini, masalan, ma’lumotlarni kiritish – qayta ishlash – natijalarni chiqarishni amalga oshiradi;

– menyu – amallari iyerarxik tuzilmalarga shakllangan ishning ssenariylari to‘plamini, masalan, «qo‘yish», «faylni qo‘yish», «simvolni qo‘yish» va h.k.larni bajaradi;

– erkin novigasiyali – iyerarxiya darajalariga bog’lanmagan va ishning konkret qadamida mumkin bo‘lgan amallar to‘plamini aniqlashni ko‘zda tutadigan ssenariylar to‘plamini amalga oshiradi, bu shaklning interfeyslari asosan Windows – ilovalardan foydalanadi;

– to‘g’ridan-to‘g’ri manipulasiyalash – obyektlar ustidagi amallarda taqdim etilgan ssenariylar to‘plamini bajaradi, asosiy amallar sichqoncha bilan obyektlar piktogrammalarini siljitish bilan amalga oshiriladi, bu shaklli erkin novigasiyali interfeysga muqobil bo‘lib Windows operasion tizimi o‘zining interfeysida amalga oshirilgan.

Foydalanuvchi interfeys tipi ishlanmaning murakkabligi va mehnat sarfini belgilaydi. Dasturiy ta’minotni ishlab chiqishning obyektli – yo‘naltirilgan vizual muhitlari dasturlashga hodisaviy yondashuvdan foydalanadi va erkin novigasiyali interfeyslarni yaratishga mo‘ljallangan bo‘lib, bunday interfeyslarni ishlab chiqish mehnat sarfini jiddiy kamaytirdi va to‘g’ridan-to‘g’ri manipulyasiyalash interfeyslarni amalga oshirishni soddalashtirdi. Interfeys turini tanlash hujjatlar bilan ishlash texnologiyalarini tanlashni o‘z ichiga oladi. Ikkita texnologiya mavjud:

– bir hujjatli, u bir hujjat interfeys (SDI – Single Document Interface) ni ko‘zda tutadi;

– ko‘p hujjatli, u ko‘p hujjatli interfeys (MDI – Multiple Document Interface) ni ko‘zda tutadi.

Ko‘p hujjatli texnologiya dasturiy ta’minoti bir nechta hujjatlar bilan bir vaqtda ishlaganda, masalan, bir nechta matn yoki bir nechta tasvirlar bilan ishlaganda foydalaniladi. Bir hujjatli – agar bir nechta hujjatlar bilan bir vaqtda ishlash zarurati bo‘lmasa, hozirgi kutubxonalardan foydalanib ko‘p hujjatli interfeyslarni ishlatish mehnat sarfi birinchisiga qaraganda 3-5% yuqori.

Agar erkin navigasiyali yoki to‘g’ridan-to‘g’ri manipulyasiyali interfeys tanlangan bo‘lsa, u holda hodisaviy dasturlash va obyektli yondashuvdan foydalaniladi, chunki zamonaviy vizual dasturlash muhitlari va ularga o‘xshashlar kutubxona sinflari obyektlari ko‘rinishida interfeys komponentlarni taqdim etadi. Bunda predmet soha murakkabligiga bog’liq holda dasturiy ta’minot obyektlardan foydalanish orqali yoki sof prosedurali ravishda amalga oshirilishi mumkin. Primitiv interfeys va menyu tipidagi interfeys tuzilmali obyektli yondashuvlar bilan muvofiqlashadi. Shuning uchun yondashuvni tanlash qo‘shimcha axborotdan foydalanish bilan amalga oshadi. Dasturlash tillarini quyidagi guruhlarga ajratish mumkin:

– yuqori darajali universal tillar;
– dasturiy ta’minot ishlab chiqaruvchisini maxsus tillari;
– foydalanuvchining maxsus tillari;
– quyi darajali tillar.

Dashturlash muhiti deb, dasturlarni yozish va saqlash jarayonini soddalashtiradigan kompilyatorga ichki qurilgan maxsus matn muharriri, sozlovchi, ma’lumotlar tizimi va boshqa dasturlarni o‘z ichiga olgan dasturiy kompleksga aytiladi. Vizual dasturlash muhitlarida dasturchi komponentlar maxsus kutubxonalaridan ba’zi kodlar dasturiga vizual ulanish imkoniyatiga ega bo‘ladi, bu esa obyektli yo‘naltirilgan dasturlash rivoji tufayli sodir bo‘ldi. Konstruktsiyalash ixtiyoriy texnologiyaning real qo‘llash barcha qatnashchilari rioya qilish lozim bo‘lgan dastur standartlarni shakllantirish yoki tanlashni talab etadi:

– konstruktsiyalash standarti;
– konstruktsiya hujjatni rasmiylashtirish standarti;
– foydalanuvchi interfeysi standarti.

Konstruktsiyalash standarti quyidagilarni aniqlashi lozim:

– konstruktsiyalash har bir bosqichida zarur modellar majmuasi va ularni detallashtirish darajasi;

– diagrammalarda konstruktsiya yechimlarni qayd etish qoidalarini, shu jumladan atamalar bo‘yicha obyektlarni nomlash va shartnomalar qoidalari, barcha obyektlar uchun atributlar majmuasi va ularni har bir bosqichida to‘ldirish qoidalari, obyektlar shakli va o‘lchamlarga talablar kiritilgan diagrammalarni rasmiylashtirish qoidalari;

– operasion tizim va foydalanilayotgan CASE – vositalarni sozlashni o‘z ichiga olgan ishlab chiqaruvchilar ish joylari konfigurasiyalariga talablar;

– konstruktsiya ustida birgalikda ishlashni ta’minlash mexanizmi, shu jumladan konstruktsiya qism tizimlarini integrasiyasi va konstruktsiya yechimlarni qarama-qarshilikka tahlil etish qoidalari.

Konstruktsiyalash hujjatlarini rasmiylashtrish standartiga:

– har bir bosqichda hujjatlar komplektligi, tarkibi va tuzilishi;

– uning mazmuni va rasmiylashtirilishiga talablar;

– hujjatni tayyorlash, qarab chiqish, kelishtirish va tasdiqlash qoidalari.

Foydalanuvchi interfeysi standarti:

– ekranlarini (shriftlar va ranglar), oynalar tarkibi, joylashishi va boshqaruv elementlarini rasmiylashtirish qoidalari;

– yordamchi matnlarni rasmiylashtirish qoidalari;

– standart xabarlar ro‘yxati;

– foydalanuvchi reaksiyasini qayta ishlash qoidalarini aniqlashi lozim.

Barcha yuqorida keltirilgan konstruktsiyaviy yechimlar ishlanmasi mehnat sarfi va murakkabligiga jiddiy ta’sir ko‘rsatadi. Ularni qabul qilgandan so‘nggina konstruktsiyalanaayotgan dasturiy ta’minotning talablar tahliliga va o‘ziga xos xususiyatlarini ishlab chiqishga o‘tish lozim.

Hozirda davlat hokimiyati va boshqaruvi organlari dasturiy mahsulotlarning faol iste’molchisi hisoblanadi. Keyingi vaqtda mamlakatimizda vazirlik va idoralar, xo‘jalik birlashmalari, yirik korxonalarning xarajat smetasi va biznes rejalarida ishlab chiqarish jarayonlarini avtomatlashtirish, axborot tizimlari va resurslarini yaratish, dasturiy mahsulotlar hamda interfaol davlat xizmatlarini joriy etishga qaratilgan xarajatlarni nazarda tutish yuzasidan tegishli ishlar olib borilmoqda.

2.3. Konstruksiyalash jarayonining tuzilishi

Konstruksiyalashning aspektlari obyektning bir-biriga bog’liq bo‘lgan xossalarini xarakterlaydi. Texnik obyektni izohlash ushun quyidagi aspektlar e’tiborga olinadi:

- funksional aspekt;
- konstruktorlik aspekt;
- texnologik aspekt;

Funksional aspekt obyekt ishlatilayotganda yoki xarakatlanayotganda unda ro‘y beradigan fizik yoki axborotli jarayonlarni ifodalaydi. Konstruktorlik aspekt obyektni yoki uning qismlarining shaklini, joylashishini va tuzilishini xarakterlaydi.

Texnologik aspekt obyektni berilgan sharoitda tayyorlash usullarini, imkokiyatlarini va texnologik ishlanishini xarakterlaydi. Yechimi funksional aspektga bog’liq bo‘lgan konstruksiyalash masalalari guruxi funksional konstruksiyalash deyiladi.

Funksional konstruksiyalash natijasida strukturali va funksional tizimlar yaratiladi yoki ularga o‘zgartirishlar kiritiladi. Barsha konstruktorlik ishlari esa konstruktorlik konstruksiyalashda, texnologik ishlar texnologik konstruksiyalashda amalga oshiriladi. Ba’zi xollarda izohlarni qansha ko‘p aspektlarga bo‘lishiga to‘g’ri keladi. Misol ushun hisoblash tizimlarini konstruksiyalashda algoritmni izohlash alohida aspekt bo‘lsa, programma tuzish alohida aspekt hisoblanadi. Elektromexanik tashqi qurilmalarni konstruksiyalashda ham elektron aspektlar va mexanik aspektlar mavjud bo‘ladi. Murakkab obyektlarni tasvirlashda har bir aspekt ishida ierarxik darajalarni tashkil qilish kerak bo‘ladi. Eng yuqori ierarxik daraja bir-biriga bog’liq holda harakatlanuvshi qism tizimlar to‘plami bo‘lgan murakkab obyekt turadi. Bu xolda har bir qism tizim ushun izohlar har tomonlama to‘liq bo‘lishi shart emas, shunki to‘la qilib yozilsa izoh, murakkablashib ketadi va juda ko‘p mexnat talab qiladi.

Keyingi ierarxik darajada har bir qism tizim aloxida tizim qilib quriladi va bu tizim ham o‘z navbatida ba’zi qismlarga bo‘linib shu tizimlar ushun to‘la izoh, beriladi. Bunday ierarxik darajalarga bo‘linishi to oddiy elementlarni izohlashgacha davom ettirish mumkin. Dasturiy ta’minotni konstruksiyalashda quyidagi ierarxik darajalarni ko‘rsatish mumkin:

- funksional-mantiqiy;
- komponentli.

Konstruksiyalash asosan davrga, bosqichga va proseduraga bo‘linadi. Murakkab obyektlarni konstruksiyalashda quyidagi davrlar mavjud:

- ilmiy-tekshirish ishlari;
- tajriba-konstruktorlik ishlari;
- texnik loyiha yaratish;
- ishshi loyiha yaratish;
- tajriba namunasini yaratish.

Ilmiy tekshirish ishlari (ITI) davri konstruksiyalashgacha bo‘lgan tekshirish davri, texnik vazifa davri texnik taklif kiritish davriga bo‘linadi. Bu davtlarda ketma-ket berilgan maqsadga muvofiq, yangi maxsulot ishlab shiqarish ushun kerak bo‘ladigan ma’lumotlar o‘rganiladi. Fizik axborotli konstruktorlik va texnologik prinsiplari va bu prinsiplarni tadbiq qilish imkoniyatlari tekshiriladi; obyektning ba’zi parametr va xarakteristikasining qiymatlari haqida ma’lumotlar to‘planadi. Ilmiy tekshirish ishlari davrining natijasi bo‘lib, yangi obyektni yaratishga mo‘ljallangan ishlab chiqish texnik masalani hisoblanadi.

Tajriba-konstruktrrlik ishlari davrida maxsulotni eskiz konstruktsiyalari yaratiladi va ilmiy tekshirish ishlari davrida yaratilgan prinsiplari va tutgan o‘rni tekshiriladi, aniqlanadi va kerak bo‘lsa tuzatishlar kiritiladi. Texnik loyiha davrida obyekt to‘g’risida to‘la texnik yechim qabul qilinadi va konstruktsiyaning hamma qismlari ishlab chiqariladi. Ishshi konstruktsiya yaratish davrida oxirgi natija olinib, sinab ko‘riladi va ba’zi bir xatoliklar yoki tugamay qolgan ishlar aniqlanadi va ularni yo‘qotishga harakat qilinadi va natijaviy hujjatlar ishlab chiqaruvshi korxonaga mahsulotni ko‘plab ishlab chiqarishga yuboriladi.

Konstruksiyalash bir necha bosqichlarga bo‘linadi. Konstruksiyalash bosqichi deb bitta yoki bir neshta konstruksiyalash proseduralarini amalga oshirish mumkin bo‘lgan, hosil bo‘lgan konstruksiyalash prosedurasi bitta konstruksiyalash yechimini ierarxik darajasini yoki bitta izohlash aspektiga mos keladigan, konstruksiyalashning shartli ravishda ajratilgan qismlariga aytiladi.

Konstruksiyalash iterasion xarakterga ega. Konstruksiyalashning har bir bosqichida, oldingi qabul qilingan yechim xato bo‘lib chiqishi yoki optimal yechim bo‘lmasligi mumkin, natijasida konstruksiyalash prosedsrasi boshqatdan ko‘rib chiqiladi. Prosedura va bosqichlarni qanday ketma-ketlikda bajarilishiga qarab pasayib boruvshi va yuqorilab boruvshi konstruksiyalashga ajratiladi. Pasayib boruvshi

konstruksiyalashda yuqori ierarxik darajadagi masalalar past ierarxik darajadagi masalalardan oldin yechiladi, yuqorilab boruvshi konstruksiyalashda esa aksinchadir. Murakkab tizimlarni funksional konstruksiyalash pasayib boruvshi konstruksiyalashga kiradi. Konstruktorlik konstruksiyalash masalalarini yechish yuqorilab boruvshi konstruksiyalashga kiradi.

Natijaviy konstruksiyalash yechimiga olib keladigan konstruksiyalash prosedurasining ketma-ketligi konstruksiyalash yo‘nalishi deyiladi. Konstruksiyalashni sintez qilish, konstruksiyalanilayotgan obyektni izohini yaratishni o‘z ishiga oladi. Bu izohlarda obyektning parametrlari va strukturasi tasvirlanadi. Cintez prosedurasi strukturali sintez qilish va parametrik sintez qilish prosedurasiga bo‘linadi.

Obyektning strukturasi uning elementlar tarkibi va bu elementlar o‘rtasidagi bir-biri bilan bog’lanish usullaridir. Ob’ektning parametri, obyektning qandaydir xossasini va uning ishlash rejimini x.arakterlaydigan kattalikdir. Strukturali sintez qilish prosedurasiga misol qilib, mantiqiy algoritmlarni sintez qilishni keltirish mumkin. Parametrik sintez qilish prosedurasi, obyektning berilgan strukturasidagi elementlarning parametrlarini qiymatlarini hisoblashni o‘z ishiga oladi.

Obyektni analiz qilish prosedurasi uning xossalari haqida foydali axborot olishga qaratilgan bo‘lib, loyihalanilayotgan obyektni va uning izohini aniqdashdan iboratdir. Ko‘pgina obyektni analiz qilishda bitta obyekt ushun ikki xil izoh yaratiladi. Ulardagi biri birinshi yaratilgan izoh hisoblanib, ma’lum miqdorda xatoliklari tuzatilgan hisoblanadi, Ikkinshi izoh aniqroq bo‘lgan ierarxik darajada tuzilib, uning to‘g’riligi birinchi izohga taqqoslab tekshiriladi va bu jarayon taqqoslash deyiladi.

2-bob bo‘yicha xulosalar

Modullashtirish - bu dasturiy ta’minot tizimini vazifalarni mustaqil bajarishga qodir bo‘lgan bir nechta diskret va mustaqil modullarga bo‘lish usuli. Dasturiy ta’minot modulini ishlab chiqishdagi birinchi qadam, asosan, dastur tuzilmasini pastdan tutashgan boshqarishdir: modulning spetsifikatsiyasini o‘rganib chiquvchi, uning o‘zi uchun tushunarli ekanligiga va ushbu modulni ishlab chiqish uchun etarli ekanligiga ishonch hosil qilishi kerak. Dasturiy modulni ishlab chiqishning ikkinchi bosqichida, qo‘yilgan muammoni hal qilish uchun yoki unga yaqin bo‘lgan algoritmlar allaqachon ma’lum yoki yo‘qligini aniqlash kerak.

Uchinchi bosqichda modul matni tanlangan dasturlash tilida tuziladi. Modul spetsifikatsiyasida ko‘rsatilgan funktsiyalarni amalga oshirishda hisobga olinishi kerak bo‘lgan har xil tafsilotlarning ko‘pligi osongina juda ko‘p xato va noaniqliklarni o‘z ichiga olgan juda chalkash matn yaratilishiga olib kelishi mumkin. Modulni rivojlantirishning navbatdagi bosqichi dasturiy ta’minot tizimining sifat spetsifikatsiyasiga muvofiq modul matnini to‘liq shaklga keltirish bilan bog’liq. Tarkibiy dasturlashning asosiy konstruktsiyalari quyidagilardan iborat: amal qilish, tarmoqlash va takrorlash.

Dasturiy tizimlar odatda rivojlangan va ichki interfeyslarga ega, bu esa ularni puxta konsrtuksiyalashni talab etadi. Ko‘p foydalanuvchili dasturiy tizimlar odatdagi dasturiy tizimlardan farqli dasturiy ta’minot alohida komponentlar o‘zaro ta’sirini tashkil etishi lozim, bu esa uni ishlab chiqishni yanada qiyinlashtiradi. Dasturiy ta’minotni ishlab chiqishning obyektli – yo‘naltirilgan vizual muhitlari dasturlashga hodisaviy yondashuvdan foydalanadi va erkin novigasiyali interfeyslarni yaratishga mo‘ljallangan bo‘lib, bunday interfeyslarni ishlab chiqish mehnat sarfini jiddiy kamaytirdi va to‘g’ridan-to‘g’ri manipulyasiyalash interfeyslarni amalga oshirishni soddalashtirdi.

Konstruksiyalashning aspektlari obyektning bir-biriga bog’liq bo‘lgan xossalarini xarakterlaydi. Konstruksiyalash bir nesha bosqichlarga bo‘linadi. Konstruksiyalash bosqichi deb bitta yoki bir neshta konstruksiyalash proseduralarini amalga oshirish mumkin bo‘lgan, hosil bo‘lgan konstruksiyalash prosedurasi bitta konstruksiyalash yechimini ierarxik darajasini yoki bitta izohlash aspektiga mos keladigan, konstruksiyalashning shartli ravishda ajratilgan qismlariga aytiladi.

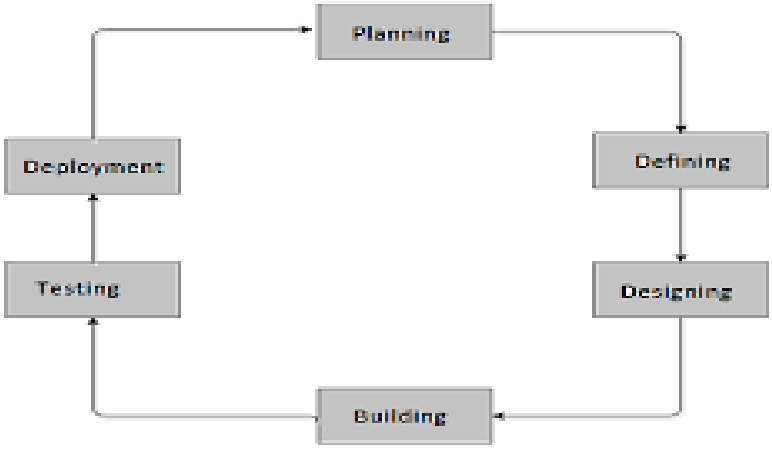
2-bob bo‘yicha nazorat savollari

1. Dasturiy ta'minotni konstruktsiyalash deganda nimani tushunasiz?
2. Dasturiy ta’minotni konstruktsiyalash qanday natijalarni beradi?
3. Modullashtirish iborasini tushuntirib bering.
4. Tarkibiy dasturlash deganda nimani tushunasiz?
5. Dashturlash muhiti deb nimga aytiladi?
6. Talablarni tahlil qilish deganda nimani tushunasiz?
7. Konstruksiyalash jarayoni qanday tuzilishga ega.
8. Funksional konstruksiyalash iborasini tushuntirib bering.
9. Konstruksiyalash qanday bosqichlarga bo‘linadi.
10. Foydalanuvchi interfeysi deganda nimani tushunasiz?

3-BOB. DASTURIY TA’MINOTNING HAYOTIY DAVRI

3.1. Dasturning klassik hayotiy davri

Dasturiy ta’minotning hayotiy davri deb atalmish holda ishlab chiqish mumkin emas. Ehtimol, oddiy foydalanuvchiga buni bilishning hojati yo‘q, lekin asosiy standartlarni o‘rganish maqsadga muvofiqdir.



3.1-rasm. Dasturiy ta’minotning hayotiy davri[3]

Har qanday tizimning hayotiy tsikli ostida, rivojlanish bosqichidan tortib, tanlangan dastur sohasida foydalanishni to‘liq rad etish paytigacha, arizani ishlatishdan to‘liq qaytarib olishigacha bo‘lgan vaqtni anglatishi odatiy holdir. Oddiy so‘zlar bilan aytganda, dasturlar, ma’lumotlar bazalari yoki hatto "operatsion tizimlar" shaklidagi axborot tizimlari faqat ma’lumotlar va ular taqdim etadigan imkoniyatlarning dolzarbligi sharoitida talabga ega. Hayotiy tsiklning ta’rifi hech qanday tarzda ishlab chiqarishda eng o‘zgaruvchan bo‘lgan beta-versiyalar kabi sinov dasturlariga taalluqli emas deb ishoniladi. Dasturiy ta’minotning hayotiy tsiklining o‘zi ko‘pgina omillarga bog’liq bo‘lib, ular orasida asosiy rollardan birini dastur ishlatiladigan muhit o‘ynaydi. Biroq, hayot aylanishi kontseptsiyasini aniqlashda foydalaniladigan umumiy shartlarni ajratib ko‘rsatish mumkin:

- Dastlabki talablar;
- muammoni shakllantirish;
- tizim uchun dasturiy ta’minotning o‘zaro talablarini tahlil qilish;
- dizayn;
- dasturlash;

[3] Данилкин Ф.А., Сычугов А.А. Конструирование программного обеспечения: учебное пособие. Изд-во ТулГУ, 2010.

– kodlash va kompilyatsiya qilish;
– sinov;
– disk raskadrovka;
– dasturiy mahsulotni amalga oshirish va unga xizmat koʻrsatish.

Dasturiy taʼminotni ishlab chiqish yuqoridagi barcha bosqichlardan iborat va hech boʻlmaganda bittasini bajarolmaydi. Ammo bunday jarayonlarni boshqarish uchun maxsus standartlar oʻrnatiladi. Dasturiy taʼminotning hayot aylanish jarayonining standartlarining shartlari va talablarini oldindan belgilab beradigan tizimlar orasida bugungi kunda biz faqat uchta asosiy narsani nomlashimiz mumkin:

– GOST 34.601-90;
– ISO / IEC 12207: 2008;
– Oracle CDM.

Ikkinchi xalqaro standart uchun rus analogi mavjud. Bu tizim va dasturiy taʼminot muhandisligi uchun javobgar boʻlgan GOST R ISO / IEC 12207-2010. Ammo ikkala qoidada tasvirlangan dasturiy taʼminotning hayot aylanishi aslida bir xil. Dasturiy taʼminotning hayotiy davri - bu dasturiy mahsulotni yaratish zarurligi toʻgʼrisida qaror qabul qilingan paytdan boshlanib, xizmatdan toʻliq chiqib ketish paytida tugaydigan vaqt.

Dasturiy taʼminotning hayotiy davri bosqichlarini aniqlash zarurati ishlab chiquvchilarning dasturiy taʼminotning sifatini yaxshilashni optimallashtirish boshqaruvi va sifatni boshqarish mexanizmlarini har xil bosqichida muammoning paydo boʻlishidan tortib dasturiy taʼminotni qoʻllab-quvvatlashgacha yaxshilash orqali yaxshilashga intilishidan kelib chiqadi. Dasturiy taʼminotning hayotiy davrining eng umumiy vakili bu asosiy bosqichlar - jarayonlar koʻrinishidagi model boʻlib, ular quyidagilarni oʻz ichiga oladi:

– Dasturiy taʼminotga talablarning tizimli tahlili va asoslash;
– Dasturiy taʼminotni dastlabki (eskiz) va batafsil (texnik) konstruksiyalash;
– Dasturiy taʼminot tarkibiy qismlarini ishlab chiqish, ularni birlashtirish va umuman dasturiy taʼminotni tuzatish;
– Sinov, sinov jarayoni va dasturiy taʼminotni nusxalash;
– Dasturiy taʼminotning muntazam ishlashi, texnik taʼminot va natijalarni tahlil qilish;
– Dasturlarga xizmat koʻrsatish, uni oʻzgartirish va takomillashtirish, yangi versiyalar yaratish.

Ushbu model odatda qabul qilinadi va dasturiy ta’minotni ishlab chiqish sohasidagi mahalliy normativ hujjatlarga ham, chet el dasturlariga ham mos keladi. Texnologik xavfsizlikni ta’minlash nuqtai nazaridan, hayot tsikli bosqichlarini xorijiy modellarda taqdim etishning o‘ziga xos xususiyatlarini batafsil ko‘rib chiqish tavsiya etiladi, chunki bu chet el dasturiy ta’minotidir, chunki u buzg’unchilik turidagi dasturiy ta’minot nuqsonlarini tashuvchisi hisoblanadi. Hayotiy tsikl modellarining grafik taqdimoti ularning xususiyatlarini va jarayonlarning ba’zi xususiyatlarini vizual ravishda ta’kidlash imkonini beradi. Dastlab, kaskadli hayot tsikli modeli yaratildi, unda avvalgi ish natijalaridan foydalangan holda asosiy bosqichlar birin-ketin boshlandi. Bu konstruksiyaning barcha bosqichlarini qat’iy belgilangan tartibda ketma-ket bajarilishini ta’minlaydi.

Keyingi bosqichga o‘tish avvalgi bosqichda ishni to‘liq yakunlashni anglatadi. Talablarni shakllantirish bosqichida aniqlangan talablar texnik shartlar shaklida qat’iy rasmiylashtiriladi va konstruksiyani ishlab chiqishning butun muddati davomida belgilanadi. Har bir bosqich rivojlanishning boshqa rivojlanish guruhi tomonidan davom ettirilishi uchun etarli bo‘lgan to‘liq hujjatlar to‘plamining chiqarilishi bilan tugaydi. Har qanday talabning noto‘g’riligi yoki uning noto‘g’ri talqin qilinishi, natijada, konstruksiyaning dastlabki bosqichiga "orqaga qaytish" zarurligiga olib keladi va kerakli qayta ko‘rib chiqish nafaqat konstruksiya jamoasini jadvaldan chetlatadi, balki ko‘pincha xarajatlarning sifat jihatidan o‘sishiga va, ehtimol, konstruksiyaning tugatilishiga olib keladi, dastlab o‘ylab topilgan shaklda. Sharshara modeli mualliflarining asosiy noto‘g’ri tushunchasi - bu konstruksiya butun jarayonni bir marta bosib o‘tadi, konstruksiyalashtirilgan arxitektura yaxshi va ishlatish uchun qulay, dastur dizayni oqilona va sinovlar o‘tishi bilan amalga oshirishda xatolar osongina yo‘q qilinadi. Ushbu model barcha xatolar amalga oshirishda to‘planishini taxmin qiladi va shuning uchun ular komponentlar va tizim sinovlari davomida teng ravishda yo‘q qilinadi. Shunday qilib, yirik konstruksiyalar uchun palapartishlik modeli juda aniq emas va faqat kichik tizimlarni yaratish uchun samarali ishlatilishi mumkin.

Spiral hayot aylanishining modeli. Ushbu modelda e’tibor dastlabki dizayn bosqichlarining takrorlanadigan jarayoniga qaratilgan. Ushbu bosqichlarda kontseptsiyalar, talablarning texnik xususiyatlari, dastlabki va batafsil dizayni ketma-ket yaratiladi. Har bir bosqichda ishning mazmuni aniqlanadi va yaratilayotgan dasturiy ta’minotning tashqi

ko‘rinishi konsentratsiyalanadi, olingan natijalarning sifati baholanadi va keyingi takrorlash ishi rejalashtiriladi.
Har bir takrorlashda quyidagilar baholanadi:

- Konstruksiya shartlari va narxidan oshib ketish xavfi;
- Yana bitta takrorlashni bajarish zarurati;
- Tizimga qo‘yiladigan talablarni tushunishning to‘liqligi va aniqligi darajasi;
- Konstruksiyani tugatish maqsadga muvofiqligi.

Hayotiy tsikl dasturiy ta’minotini standartlashtirish uch yo‘nalishda amalga oshiriladi. Birinchi yo‘nalishni Xalqaro standartlashtirish tashkiloti (ISO - Xalqaro standart tashkilot) va Xalqaro elektrotexnika komissiyasi (IEC - Xalqaro elektrotexnika komissiyasi) tashkil etadi va targ’ib qiladi. Ushbu darajada xalqaro hamkorlik uchun muhim bo‘lgan eng umumiy texnologik jarayonlarni standartlashtirish amalga oshiriladi. Ikkinchi yo‘nalish AQShda Elektrotexnika va elektronika muhandislari instituti (IEEE) tomonidan Amerika milliy standartlar instituti (ANSI) bilan birgalikda faol ravishda ishlab chiqilmoqda. ISO / IEC va ANSI / IEEE standartlari asosan tavsiyalarga ega. Uchinchi yo‘nalishni AQSh Mudofaa vazirligi (DOD) rag’batlantiradi. DOD standartlari AQSh Mudofaa vazirligi tomonidan buyurtma qilingan firmalar uchun majburiydir.

Murakkab tizim, xususan, real vaqtda ishlaydigan tizim uchun dasturiy ta’minotni konstruksiyalash uchun ko‘rib chiqilgan asosiy jarayonlar doirasida barcha ma’lum bo‘lgan ishlarni birlashtirishga asoslangan hayot tsiklining butun tizim modelidan foydalanish maqsadga muvofiqdir. Ushbu model har xil dasturiy ta’minot konstruksiyalarini rejalashtirish, rejalashtirish, boshqarishda foydalanishga mo‘ljallangan.

Ushbu hayot tsikli modelining bosqichlari majmuini jarayonlarning xususiyatlari, texnik-iqtisodiy xususiyatlari va ularga ta’sir etuvchi omillar jihatidan bir-biridan farqli ravishda ikki qismga bo‘lish maqsadga muvofiqdir. Hayotiy tsiklning birinchi qismida tizimni tahlil qilish, konstruksiyalash, ishlab chiqish, sinovdan o‘tkazish va dasturiy ta’minotni sinovdan o‘tkazish amalga oshiriladi. Ushbu bosqichlarda ishlarning ko‘lami, ularning mehnat zichligi, davomiyligi va boshqa xususiyatlari ob’ektga va rivojlanish muhitiga sezilarli darajada bog’liqdir. Dasturiy ta’minotning turli sinflari uchun bunday bog’liqliklarni o‘rganish dasturiy ta’minotning yangi versiyalari uchun ish jadvallarining tarkibi va asosiy xususiyatlarini bashorat qilishga imkon beradi.

Dasturiy ta’minotning ishlashi va ta’minlanishini qo‘llab-quvvatlovchi hayot tsiklining ikkinchi qismi ob’ekt xususiyatlari va rivojlanish muhiti bilan nisbatan zaif bog’liqdir. Ushbu bosqichlardagi ishlar ko‘lami barqarorroq bo‘lib, ularning mehnat zichligi va davomiyligi sezilarli darajada farq qilishi mumkin va dasturiy ta’minotning ommaviy ishlatilishiga bog’liq. Hayotiy tsiklning har qanday modeli uchun yuqori sifatli dasturiy ta’minot tizimlarini taqdim etish faqat ushbu bosqichlarning har birida tartibga solingan texnologik jarayondan foydalanilganda mumkin bo‘ladi. Bunday jarayonni ishlab chiqishni avtomatlashtirish vositalari qo‘llab-quvvatlaydi, ular mavjud ob’ektlar orasidan tanlash yoki yaratish ob’ekti va ishlarning etarli ro‘yxatini hisobga olgan holda yaratish maqsadga muvofiqdir.

GOST R ISO / IEC 12207-2010 standarti, yaxshi o‘rnatilgan terminologiyadan foydalangan holda, dasturiy ta’minot sohasida qo‘llanilishi mumkin bo‘lgan dasturiy ta’minotning hayotiy tsikli jarayonlari uchun umumiy asosni yaratadi. Standart dasturiy mahsulot yoki xizmatni sotib olishda, shuningdek etkazib berish, ishlab chiqish, maqsadga muvofiq foydalanishda, dasturiy ta’minot mahsulotlarini saqlash va ulardan foydalanishni to‘xtatishda ishlatiladigan jarayonlar, faoliyat va vazifalarni belgilaydi.

3.2. Dasturiy ta’minotning hayot aylanish jarayonlari

Standart dasturiy ta’minot tizimlarining hayot tsikli davomida bajarilishi mumkin bo‘lgan turli xil tadbirlarni yetita jarayon guruhiga ajratadi. Ushbu guruhlar ichidagi hayot aylanish jarayonlarining har biri maqsadlar va kerakli natijalar, ushbu natijalarga erishish uchun bajarilishi kerak bo‘lgan harakatlar va vazifalar ro‘yxati bilan tavsiflanadi.

– kelishuv jarayonlari - ikkita jarayon;

– konstruksiyani tashkiliy qo‘llab-quvvatlash jarayonlari - beshta jarayon;

– konstruksiya jarayonlari - etti jarayon;

– texnik jarayonlar - o‘n bitta jarayon;

– dasturiy ta’minotni amalga oshirish jarayonlari - etti jarayon;

– dasturiy ta’minotni qo‘llab-quvvatlash jarayonlari - sakkizta jarayon;

– dasturiy ta’minotni qayta ishlatish jarayonlari - uchta jarayon.

Asosiy:

– Xarid qilish (dasturiy ta’minotni sotib oluvchi xaridorning harakatlari va vazifalari)

– Yetkazib berish (mijozga dasturiy mahsulot yoki xizmatni etkazib beradigan etkazib beruvchining faoliyati va vazifalari)

– Konstruksiyalash (ishlab chiquvchi tomonidan amalga oshiriladigan harakatlar va vazifalar: dasturiy ta’minotni yaratish, dizayn va ekspluatatsion hujjatlar, test va o‘quv materiallarini tayyorlash va hk).

– Operatsion (operatorning harakatlari va vazifalari - tizimni boshqaradigan tashkilot)

– Eskort (eskort tashkilot tomonidan amalga oshiriladigan harakatlar va vazifalar, ya’ni eskort xizmati). Xizmat - xatolarni tuzatish, ish faoliyatini yaxshilash yoki o‘zgargan ish sharoitlari yoki talablariga moslashish uchun dasturiy ta’minotga o‘zgartirishlar kiritish.

– Hujjatlar (dasturiy ta’minotning hayotiy tsikli davomida yaratilgan ma’lumotlarning rasmiylashtirilgan tavsifi)

– Konfiguratsiyani boshqarish (dasturiy ta’minot tarkibiy qismlarining holatini aniqlash, uning modifikatsiyasini boshqarish uchun dasturiy ta’minotning butun hayot tsikli davomida ma’muriy va texnik protseduralarni qo‘llash).

– Sifatni ta’minlash (IS va uning hayot aylanish jarayonlari belgilangan talablarga va tasdiqlangan rejalarga muvofiqligini ta’minlash)

– Tasdiqlash (ba’zi bir harakatlar natijasi bo‘lgan dasturiy mahsulotlar oldingi harakatlar natijasida kelib chiqadigan talablar yoki shartlarni to‘liq qondirishini aniqlash)

– Sertifikatlash (belgilangan talablarga va yaratilgan tizimga ularning aniq funktsional maqsadlariga muvofiqligini to‘liqligini aniqlash)

– Qo‘shma baholash (konstruksiyadagi ish holatini baholash: resurslar, xodimlar, uskunalar, asboblarni rejalashtirish va boshqarishni boshqarish)

– Audit (shartnoma talablari, rejalari va shartlariga muvofiqligini aniqlash)

– Muammolarni hal qilish (ishlab chiqish, ishlatish, texnik xizmat ko‘rsatish yoki boshqa jarayonlar davomida aniqlangan muammolarni kelib chiqishi yoki manbasidan qat’i nazar, tahlil qilish va hal qilish)

Tashkiliy:

– Nazorat (o‘z jarayonlarini boshqaradigan har qanday tomon tomonidan bajarilishi mumkin bo‘lgan harakatlar va vazifalar)

– Infratuzilmani yaratish (texnologiya, standartlar va vositalarni tanlash va qo‘llab-quvvatlash, dasturiy ta’minotni ishlab chiqish, ishlatish yoki unga xizmat ko‘rsatish uchun ishlatiladigan apparat va dasturiy ta’minotni tanlash va o‘rnatish)

– Yaxshilash (hayot tsikli jarayonlarini baholash, o‘lchash, boshqarish va takomillashtirish)

– O‘qitish (kadrlarni dastlabki o‘qitish va keyinchalik uzluksiz malakasini oshirish).

Har bir jarayon bir qator harakatlarni o‘z ichiga oladi. Masalan, sotib olish jarayoni quyidagi bosqichlarni o‘z ichiga oladi:

1. Sotib olishni boshlash
2. Ariza takliflarini tayyorlash
3. Shartnomani tayyorlash va o‘zgartirish
4. Yetkazib beruvchilar nazorati
5. Ishlarni qabul qilish va tugatish

Har bir harakat bir qator vazifalarni o‘z ichiga oladi. Masalan, takliflar tayyorlash quyidagilarni o‘z ichiga olishi kerak:

1. Tizim talablarini shakllantirish
2. Dasturiy mahsulotlar ro‘yxatini shakllantirish
3. Shartlar va kelishuvlarni belgilash
4. Texnik cheklovlarning tavsifi (tizimning ishlash muhiti va boshqalar)

3.3. Dasturiy ta’minotning hayot davrining bosqichlari

Dasturiy ta’minotning hayot davri modeli - hayot tsikli davomida bajarish ketma-ketligini va jarayonlar, harakatlar va vazifalarning o‘zaro bog’liqligini belgilaydigan tuzilma. Hayotiy tsikl modeli konstruksiyaning o‘ziga xosligi, ko‘lami va murakkabligi hamda tizim yaratilishi va ishlashi sharoitlarining o‘ziga xos xususiyatlariga bog’liq. GOST R ISO / IEC 12207-2010 ma’lum hayot tsikli modelini taklif qilmaydi. Uning qoidalari DT yaratish uchun har qanday hayot tsikli modellari, usullari va texnologiyalari uchun umumiydir. U hayot jarayonlari tarkibiga ushbu jarayonlarga kiritilgan faoliyat va vazifalarni qanday amalga oshirish yoki bajarishni ko‘rsatmasdan tavsiflaydi. Hayotiy tsikli dasturiy ta’minot modeli quyidagilarni o‘z ichiga oladi:

1. Bosqichlar;
2. Har bir bosqichda ish natijalari;
3. Asosiy voqealar - yakunlash va qaror qabul qilish nuqtalari.

Dasturiy ta’minotning hayotiy davri - bu tizimni yaratish va undan foydalanish jarayonida sodir bo‘ladigan hodisalar turkumi. Bosqich - dasturiy ta’minotni ishlab chiqish jarayonining ma’lum bir muddat bilan cheklangan va ushbu bosqich uchun belgilangan talablar bilan aniqlangan ma’lum bir mahsulot (modellar, dasturiy ta’minot tarkibiy qismlari, hujjatlar) chiqarilishi bilan yakunlangan qismi. Hayotiy davr an’anaviy ravishda bir qator ketma-ket bosqichlar (yoki bosqichlar, bosqichlar) sifatida modellashtirilgan. Hozirgi vaqtda dasturiy ta’minot tizimining hayot tsiklini bosqichlarga ajratish bo‘yicha umumiy qabul qilingan bo‘linish mavjud emas. Ba’zan sahna alohida element sifatida ajratiladi, ba’zida u katta sahnaning ajralmas qismi sifatida kiritiladi. U yoki bu bosqichda amalga oshirilgan harakatlar turlicha bo‘lishi mumkin. Ushbu bosqichlarning nomlarida bir xillik yo‘q. An’anaga ko‘ra, dasturiy ta’minotning hayotiy tsiklining quyidagi asosiy bosqichlari ajratiladi:

- Talablarni tahlil qilish,
- Dizayn,
- Kodlash (dasturlash),
- Sinov va disk raskadrovka,
- Foydalanish va texnik xizmat ko‘rsatish.

Dasturiy ta’minotning hayotiy davrining kaskad modeli oldingi bosqichda ish to‘liq yakunlangandan so‘ng keyingi bosqichga o‘tishni nazarda tutadi. Palapartishlik modelining asosiy yutug’i bosqichlarning tugashidir. Bu xarajatlar va muddatlarni rejalashtirishga imkon beradi. Bundan tashqari, to‘liq va izchil bo‘lgan konstruksiya hujjatlari shakllantiriladi. Sharshara modeli aniq belgilangan va o‘zgarmas talablarga ega bo‘lgan kichik dasturiy ta’minot konstruksiyalarida qo‘llaniladi. Haqiqiy jarayon har qanday bosqichda muvaffaqiyatsizliklarni ochib berishi mumkin, bu esa oldingi bosqichlardan biriga qaytishga olib keladi. Bunday dasturiy ta’minotni ishlab chiqarish modeli kaskad-qaytarishdir.

Dasturiy ta’minotning hayotiy davrining qidiruv boshqaruv bilan bosqichma-bosqich modeli bosqichlar orasidagi teskari aloqa davrlari bilan dasturiy ta’minotni ishlab chiqishning iterativ modeli. Ushbu modelning afzalligi shundaki, qadamlararo tuzatishlar palapartishlik modeliga qaraganda kamroq mehnat talab qiladi, ammo, har bir bosqichning umri butun rivojlanish davrida davom etadi.

Dasturlashning maqsadi ma’lumotlarni qayta ishlash jarayonlarini tavsiflashdir. Ma’lumotlar (ma’lumotlar) - bu ma’lum bir jarayonda uzatish va qayta ishlash uchun mos bo‘lgan rasmiylashtirilgan shakldagi

faktlar va g’oyalarni aks ettirish, axborotlar esa ular taqdim etilganda ma’lumotlarga beriladigan ma’no. Ma’lumotlarni qayta ishlash - bu ma’lumotlar bo‘yicha harakatlar tizimli ketma-ketligini bajarish. Ma’lumotlar ma’lumotlar tashuvchilarida taqdim etiladi va saqlanadi. Ma’lumotlarni qayta ishlashning har qanday turida foydalaniladigan ma’lumotlar tashuvchilar yig’indisi ma’lumotlar muhiti deyiladi.

Axborot muhitining har qanday lahzasida mavjud bo‘lgan ma’lumotlar to‘plami axborot muhitining holatidir. Jarayonni ma’lum bir axborot muhitining ketma-ket holatlari ketma-ketligi sifatida aniqlash mumkin. Jarayonni tavsiflash uchun axborot muhiti holatlari ketma-ketligini aniqlash kerak. Kerakli jarayonni har qanday kompyuterda berilgan tavsifga muvofiq avtomatik ravishda yaratish uchun ushbu tavsif rasmiylashtirilishi kerak. Tijorat mahsuloti mijozlar talablariga javob berishi kerak. Sifat - bu xaridorning qoniqish darajasini ko‘rsatadigan mahsulot (xizmat) ning ob’ektiv xarakteristikasi. Sifat xususiyatlari:

– Ishlash qobiliyati - tizim ishlaydi va kerakli funktsiyalarni amalga oshiradi.

– Ishonchlilik - tizim nosozliklar va nosozliklarsiz ishlaydi.

– Qayta tiklanishi.

– Samaradorlik - tizim o‘z vazifalarini iloji boricha eng yaxshi tarzda amalga oshiradi.

– Iqtisodiy samaradorlik - maksimal foyda bilan yakuniy mahsulotning minimal qiymati.

– Inson omilini hisobga olgan holda - foydalanish qulayligi, DT bilan ishlashni o‘rganish tezligi, texnik xizmat ko‘rsatish qulayligi, o‘zgartirishlar kiritish.

– Portativlik (portativlik) - kodni boshqa platformaga yoki tizimga ko‘chirish.

– Funktsional to‘liqlik - ehtimol tashqi funktsiyalarni to‘liq bajarish.

– Hisoblashning aniqligi.

Algoritm xususiyatlari.

Samaradorlik cheklangan sonli operatsiyalarni bajargandan so‘ng natija olish imkoniyatini anglatadi.

Aniqlik foydalanuvchidan va qo‘llaniladigan texnik vositalardan qat’iy nazar olingan natijalar tasodifidan iborat.

Ommaviy xarakter algoritmni boshlang’ich ma’lumotlarning o‘ziga xos qiymatlari bilan farq qiladigan bir xil turdagi muammolarning butun sinfiga qo‘llash imkoniyatida yotadi.

Diskretlik - algoritm tomonidan belgilangan hisob-kitoblar jarayonini alohida bosqichlarga bo‘lish imkoniyati, ma’lum bir tuzilishga ega dastur bo‘limlarini tanlash qobiliyati.

3-bob bo‘yicha xulosalar

Har qanday tizimning hayotiy tsikli ostida, rivojlanish bosqichidan tortib, tanlangan dastur sohasida foydalanishni to‘liq rad etish paytigacha, arizani ishlatishdan to‘liq qaytarib olishigacha bo‘lgan vaqtni anglatishi odatiy holdir. Dasturiy ta’minotning hayotiy davri bosqichlarini aniqlash zarurati ishlab chiquvchilarning dasturiy ta’minotning sifatini yaxshilashni optimallashtirish boshqaruvi va sifatni boshqarish mexanizmlarini har xil bosqichida muammoning paydo bo‘lishidan tortib dasturiy ta’minotni qo‘llab-quvvatlashgacha yaxshilash orqali yaxshilashga intilishidan kelib chiqadi. Hayotiy tsikl modellarining grafik taqdimoti ularning xususiyatlarini va jarayonlarning ba’zi xususiyatlarini vizual ravishda ta’kidlash imkonini beradi.

Dasturiy ta’minotning hayot davri modeli - hayot tsikli davomida bajarish ketma-ketligini va jarayonlar, harakatlar va vazifalarning o‘zaro bog’liqligini belgilaydigan tuzilma. Hayotiy tsikl modeli konstruksiyaning o‘ziga xosligi, ko‘lami va murakkabligi hamda tizim yaratilishi va ishlashi sharoitlarining o‘ziga xos xususiyatlariga bog’liq. Dasturiy ta’minotning hayotiy davri - bu tizimni yaratish va undan foydalanish jarayonida sodir bo‘ladigan hodisalar turkumi.

Bosqich - dasturiy ta’minotni ishlab chiqish jarayonining ma’lum bir muddat bilan cheklangan va ushbu bosqich uchun belgilangan talablar bilan aniqlangan ma’lum bir mahsulot chiqarilishi bilan yakunlangan qismi. Dasturlashning maqsadi ma’lumotlarni qayta ishlash jarayonlarini tavsiflashdir. Ma’lumotlar (ma’lumotlar) - bu ma’lum bir jarayonda uzatish va qayta ishlash uchun mos bo‘lgan rasmiylashtirilgan shakldagi faktlar va g’oyalarni aks ettirishdir. Axborot muhitining har qanday lahzasida mavjud bo‘lgan ma’lumotlar to‘plami axborot muhitining holatidir. Jarayonni ma’lum bir axborot muhitining ketma-ket holatlari ketma-ketligi sifatida aniqlash mumkin. Jarayonni tavsiflash uchun axborot muhiti holatlari ketma-ketligini aniqlash kerak. Dasturiy ta’minotning samaradorligi samaradorlik cheklangan sonli operatsiyalarni bajargandan so‘ng natija olish imkoniyatini anglatadi. Dasturiy ta’minotning aniqligi foydalanuvchidan va qo‘llaniladigan texnik vositalardan qat’iy nazar olingan natijalar tasodifidan iborat.

3-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotning hayotiy davri deganda nimani tushunasiz?
2. Hayot aylanishi kontseptsiyasini aniqlashda foydalaniladigan umumiy shartlarni aytib bering.
3. Dasturiy ta’minotning hayot aylanish jarayonining qanday standartlarini bilasiz.
4. Dasturiy ta’minotning hayotiy davri bosqichlarini sanab bering.
5. Hayotiy davr modellari deganda nimani tushunasiz.
6. Standart dasturiy ta’minot tizimlarining hayot tsikli davomida bajarilishi mumkin bo‘lgan tadbirlarni aytib bering.
7. Dasturiy ta’minotning ishlash qobiliyati deganda nimani tushunasiz.
8. Dasturiy ta’minotning ishonchlilik qobiliyati deganda nimani tushunasiz.
9. Dasturiy ta’minotning iqtisodiy samaradorligi deganda nimani tushunasiz.
10. Dasturiy ta’minotning diskretlik xususiyati haqida nimalar bilasiz.

4-BOB. DASTURIY TA’MINOTNI ISHLAB CHIQISHDA STANDARTLASHTIRISH VA METROLOGIYA

4.1. Standartlashtirishning afzallik jihatlari

Hozirgi kunda tashkilotlar, mahsulot ishlab chiqaruvchilar, shu jumladan dasturiy ta’minot o‘rtasidagi raqobat kuchaymoqda, yanada qat’iyroq talablarga talablarga olib keladi. Raqobatbardosh bo‘lish uchun tashkilotlar mahsulot sifatini oshirishga olib keladigan samarali tizimlarni qo‘llashlari va o‘z mijozlarining talablarini qoniqtiruvchi samarali tizimlarni qo‘llashlari kerak. Texnik xususiyatlarga kiritilgan mijozning to‘g’ri shakllantirilgan va to‘liq talablari, bu talablar to‘liq qoniqarli bo‘lishiga kafolat bermaydi, chunki etkazib berish tizimida kamchiliklar mavjud va tashkilotni ta’minlaydi. Ushbu qarash sifat tizimlari bilan bog’liq standartlarning rivojlanishiga va mahsulotlarga qo‘yiladigan mahsulotlarga qo‘yiladigan talablarga mos kelishga olib keldi. ISO 9000 seriyasining xalqaro standartlari sifatli tizimlar uchun umumiy asos yaratishga mo‘ljallangan. ISO 8402 ma’lumotlariga ko‘ra, tashkilot tomonidan ishlab chiqarilgan mahsulot sifatini umumiy boshqarishning umumiy boshqaruvini amalga oshirish uchun zarur bo‘lgan tashkiliy tuzilma, metodologiyalar, jarayonlar va resurslar tushuniladi.

Standartlashtirish – haqiqatda mavjud yoki potentsial vazifalarga nisbatan umumiy va ko‘p karrali foydalanish uchun qoidalarni belgilash vositasida muayyan sohada tartibga keltirishning optimal darajasiga erishishga qaratilgan ilmiy-texnik faoliyat. Aloqa va axborotlashtirish sohasida standartlashtirish bo‘yicha faoliyatni boshqarish va ishlarni muvofiqlashtirishni O‘zbekiston Respublikasi Axborot texnologiyalari va kommunikatsiyalarini rivojlantirish vazirligi amalga oshiradi.

Standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha asosiy maqsad mahsulotlar va xizmatlar sifati, havfsizligi va raqobatbardoshliligini ta’minlashdir. Standartlashtirishning afzallik jihatlarini shunday ta’riflash mumkin, standartlashtirish iqtisodiyotni maksimallikka erishishiga qaratilgan. Standartlar jamiyatning turli jabhalarida bir qancha afzalliklar olib kiradi. Standartlar tovarlar va xizmatlarni yanada takomillashtirish uchun ilmiy tadqiqotlar va rivojlantirishning yangi bosqichlarini shakllanishida xizmat qiladi. Aloqa va axborotlashtirish sohasida standartlashtirish bo‘yicha ishlarni tashkil qilish va o‘tkazish uchun:

– «UNICON.UZ» Fan-texnika va marketing tadqiqotlari markazi qoshidagi Standartlashtirish tayanch tashkiloti (STT) (http://unicon.uz/bos/);

– standartlashtirish bo‘yicha texnik qo‘mita;

– O‘zbekiston Respublikasi Axborot texnologiyalari va kommunikatsiyalarini rivojlantirish vazirligining Normativ hujjatlar bo‘yicha ekspert komissiyasi (Normativ hujjatlar bo‘yicha ekspert komissiya);

– xo‘jalik yurituvchi sub’ektlardagi standartlashtirish xizmatlari (standartlashtirish bo‘yicha vakil) ish olib boradilar. Aloqa va axborotlashtirish sohasida standartlashtirish bo‘yicha faoliyatni rivojlantirish va takomillashtirish maqsadida O‘zbekiston davlat standartlashtirish tizimining tarkibiy qismi hisoblangan tarmoq «Aloqa va axborotlashtirish sohasida standartlashtirish tizimi» yaratildi.

Aloqa va axborotlashtirish sohasidagi standartlashtirish tizimining asosiy printsiplari quyidagilardan iborat:

– normativ hujjatlarning texnik, ijtimoiy zarurligi va maqbulligini hisobga olgan holda ularni ishlab chiqishning maqsadga muvofiqligi;

– standartlashtirishning o‘zaro bog’langan ob’ektlarini standartlashtirish kompleksliligi, shu jumladan ushbu ob’ektlarga qo‘yiladigan kelishilgan talablarni ishlab chiqish va normativ hujjatlarni amalga kiritish muddatlarini muvofiqlashtirish asosida metrologik ta’minot;

– aloqa va axborotlashtirish sohasida ishlab chiqiladigan normativ hujjatlarning fan, texnikaning zamonaviy yutuqlariga, aloqa va axborotlashtirish sohasini rivojlantirish tendentsiyalari, ilg’or tajriba, qonun hujjatlariga muvofiqligini ta’minlash;

– standartlashtirish bo‘yicha barcha darajadagi normativ hujjatlarning o‘zaro bog’liqligi va kelishilganligi, normativ hujjatlar ishlab chiqilishini boshqaruvning turli darajalardagi standartlashtirish identik ob’ektlariga takrorlanishini istisno etish;

– standartlashtirish bo‘yicha normativ hujjatlar, dasturlar va ishlar rejasi to‘g’risidagi axborotning ochiqligi;

– normativ hujjatlarni ko‘pchilik manfaatdor tomonlar kelishuviga erishish asosida tasdiqlash;

– normativ hujjatlardan sertifikatlashtirish va metrologiya maqsadlari uchun foydalanish imkoniyati;

– standartlashtirish sohasida zamonaviy axborot tizimlari va texnologiyalarini qo‘llash.

Xo‘jalik yurituvchi sub’ektlar tomonidan xalqaro, davlatlararo, milliy, mintaqaviy va davlat hamda tarmoq standartlarining keng qo‘llanishi uchun Standartlashtirish tayanch tashkiloti tomonidan normativ hujjatlar fondi yaratiladi va yuritiladi. Fond strukturasi standartlashtirish bo‘yicha xalqaro, mintaqaviy va xorijiy tashkilotlarning (ISO, ITU, ETSI) normativ hujjatlari, davlatlararo standartlar, O‘zbekiston davlat standartlari, rahbariy hujjatlari va tavsiyalari; qog’oz va elektron eltuvchilardagi tarmoq normativ hujjatlaridan iborat. Normativ hujjatlar «Aloqa va axborotlashtirish sohasi normativ hujjatlarining ma’lumotlar banki» avtomatlashtirilgan tizimiga (NHMB AT) kiritiladi.

Keng doiradagi foydalanuvchilarning aloqa va axborotlashtirish sohasi normativ hujjatlarining ma’lumotlar bankidan tezkorlik bilan foydalana olishi maqsadida NHMB AT Internet-versiyasi ishlab chiqildi. Joriy yilning iyul oyidan sohadagi kompaniyalar va korxonalar http://stt.unicon.uz saytida ro‘yxatdan o‘tganidan va parol olganidan keyin, ma’lumotlar bankidan normativ hujjatlarni bepul olishlari mumkin. Normativ hujjatlar bo‘yicha O‘zbekiston Respublikasi Axborot texnologiyalari va kommunikatsiyalarini rivojlantirish vazirligining ekspert komissiyasi aloqa va axborotlashtirish sohasining normativ-huquqiy bazasini takomillashtirish, normativ-huquqiy hujjatlarni ishlab chiqish va qayta ko‘rib chiqish, ularni ekspertiza qilish maqsadida tashkil qilindi.

Standartlashtirish asosiy maqsadi mahsulotlar va xizmatlar sifati, havfsizligi va raqobatbardoshliligini ta’minlashdir. Uni shunday ta’riflash mumkin, standartlashtirish iqtisodiyotni maksimallikka erishishiga qaratilgan. Standartlar jamiyatning turli jabhalarida bir qancha afzalliklar olib kiradi. Standartlashtirishning ba’zi imtiyozlari quyidagicha:

– Ishlab chiqaruvchilar uchun standartlar:

a) Ishlab chiqarishning mantiqiy jarayoni.

b) Isrof qilinayotgan material yoki mehnatni kamaytirish yoki yo‘q qilish.

c) Xom ashyo va tayyor mahsulotlar anjomlarini kamaytirish.

d) Ishlab chiqarish xarajatlarini kamaytirish.

– Mijozlar uchun standartlar:

a) Sotib olingan tovarlar va xizmatlar sifatini ta’minlash.

b) Yaxshi sifat uchun pul berish.

c) Yetkazib beruvchilar bilan har qanday nizolar bo‘lsa, ularni hal qilish uchun qulay.

– Savdogarlar uchun standartlar:

a) Tovarlar va xizmatlar uchun nizolarni qabul qilish yoki rad etish uchun asos bo‘lib xizmat qiladi.

b) Noto‘g’ri xat-xabarlar kelishini yoki to‘liq spetsifikatsiya materiallari yoki mahsulotlarini kechikishini kamaytirish.

– Texnologlar uchun standartlar:

a) Tovarlar va xizmatlarni yanada takomillashtirish uchun ilmiy tadqiqotlar va rivojlantirishning yangi bosqichlarini shakllanishida xizmat qiladi.

Standartlarning xususiyatlari

Standart asosan uchta xususiyatga ega bo‘ladi:

– Darajaviy: kompaniyada milliy va xalqaro darajalar singari.

– Ilmiy: oziq-ovqat, tekstil va menejment injiniringi kabi.

– Aspekt: qadoqlash va yorliqlashtirish, sinovdan o‘tkazish va analizlash, o‘ziga xos xususiyatlarini aniqlashtirishga o‘xshagan.

Standartlashtirishning asosiy maqsadlari quyidagilardan iborat:

– mahsulotlar, ishlar va xizmatlarning (keying o‘rinlarida mahsulotlar deb yuritiladi) aholining hayoti, salomatligi va mol-mulki, atrof-muhit uchun xavfsizligi, resurslarni tejash masalalarida iste’molchilarning va davlatning manfaatlarini himoya qilish;

– mahsulotlarning o‘zaro bir - birining o‘rnini bosishini va bir-biriga monandligini tahminlash;

– fan va texnika taraqqiyoti darajasiga, shuningdek, aholi va xalq xo‘jaligining ehtiyojlariga muvofiq mahsulotlarning sifati hamda raqobatbardoshligini oshirish;

– resurslarning barcha turlarini tejashga, ishlab chiqarishning texnikaviy-iqtisodiy ko‘rsatkichlarini yaxshilashga ko‘maklashish;

– ijtimoiy-iqtisodiy, ilmiy-texnikaviy dasturlar va loyihalarni amalga oshirish;

– tabiiy va texnogen falokatlar va boshqa favqulodda vaziyatlar yuzaga kelishi, xavf-xatarni hisobga olgan holda xalq xo‘jaligi obyektlarining xavfsizligini tahminlash;

– iste’molchilarni ishlab chiqarilayotgan mahsulotlar nomenklaturasi va sifati to‘g’risidagi to‘liq va ishonarli axborot bilan tahminlash;

– mudofaa qobiliyatini va safarbarlik tayyorgarligini tahminlash;

– o‘lchashlarning yagonaligini tahminlash;

Standartlashtirishning asosiy vazifalari:

– iste’molchi va davlatning manfaati yo‘lida mahsulotning sifati va nomlariga nisbatan eng maqbul talablarni qo‘yish;

– davlat, respublika fuqarolari va chet el ehtiyoji uchun tayyorlangan mahsulotga kerakli talablarni belgilovchi me’yoriy hujjatlar tizimini va uni ishlab chiqish qoidalarini yaratish, ishlab chiqish va qo‘llash, shuningdek hujjatlardan nazorat qilish;

– standart talablarining sanoati rivojlangan chet mamlakatlarning xalqaro, mintaqaviy va milliy standartlari talablari bilan uyg’unlashuvini tahminlash;

– bir-biriga mosligining barcha (konstruktiv, elektrik, elektromagnitli, informatsion, dasturli va boshqalar) turlarini, shuningdek mahsulotning o‘zaro almashinuvchanligini tahminlash;

– parametrik va turlar o‘lchovi katorlarini, tayanch konstruktsiyalarni, buyumlarning konstruktiv jihatdan bir xil qilingan modullashgan bloki tarkibiy qismlarini aniqlash va qo‘llash asosida birxillashtirish;

– mahsulot, uning tarkibiy kismlari, buyumlari, xom-ashyo va materiallar ko‘rsatkichlari va tavsiflarining kelishib olinishi va bog’lanishi;

– material va energiya sig’imini kamaytirish, kam chiqindi chiqaruvchi texnologiyalarni qo‘llash;

– mahsulotning ergonomik xossalariga talablarning belgilanishi;

– metrologik me’yor, qoida, nizom va talablarning belgilanishi;

– standartlashtirish bo‘yicha xalqaro tajribadan foydalanishni keng avj oldirish, mamlakatning xalqaro va mintaqaviy standartlashtirishda ishtirok etishini kuchaytirish;

– xorijiy mamlakatlarning talablari O‘zbekiston Respublikasining xalq xo‘jaligi ehtiyojlarini qondirolgan hollarda ularning xalqaro, mintaqaviy va milliy standartlarini mamlakat standartlari va texnikaviy shartlari tariqasida to‘g’ridan-to‘g’ri qo‘llash tajribasini kengaytirish;

– texnologik jarayonlarga talablarni belgilash;

– mahsulotni standartlashtirish va uning natijalaridan foydalanish sohasida xalqaro hamkorlik qilish yuzasidan ishlarni tashkil qilish;

– texnika-iqtisodiy axborotni tasniflash va kodlash tizimini yaratish va joriy qilish;

– sinovlarni me’yoriy-texnika jihatidan tahminlash, mahsulot sifatini sertifikatlashtirish, baholash va nazorat qilish;

4.2. Standartlashtirishning asosiy tushuncha va tamoyillari

Standart bu mahsulot yoki xizmatlar uchun talablar, qoidalar va ko‘rsatmalarni o‘zaro bir-biri bilan mutanosib kelishini ta’minlovchi hujjat hisoblanadi. Bu talablar, ba’zan, mahsulotlar yoki xizmatlar jarayonlari tavsifi bilan tushuntiriladi. Standartlar kelishuvlar natijasidir va ular tegishli organ tomonidan tasdiqlanadi. Standartlar maqsadi belgilangan doiradagi tartibini optimal darajasiga erishishdir. Standartlashtirish – standartlarni shakllantirish, ishlab chiqish va tadbiq etish jarayoniga aytiladi. Standart - bu ko‘pchilik manfaatdor tomonlar kelishuvi asosida ishlab chiqarilgan va ma’lum sohalarda eng maqbul darajali tartiblashtirishga yo‘naltirilgan hamda faoliyatning har xil turlariga yoki natijalariga tegishli bo‘lgan umumiy va takror qo‘llaniladigan qoidalar, umumiy qonun-qoidalar, tavsiyalar, talablar va usullarini o‘z ichiga olgan hamda, belgilangan va tan olingan idora tomonidan tasdiqlangan me’yoriy hujjatdir.

Standartlar fan, texnika va tajribalarning umumlashtirilgan natijalariga asoslangan va jamiyat uchun yo‘naltirilgan bo‘lishi kerak. Standartlashtirish deganda mavjud yoki bo‘lajak masalalarga nisbatan umumiy va ko‘p marta tatbiq etiladigan talablarni belgilash orqali ma’lum sohada eng maqbul darajada tartiblashtirishga yo‘naltirilgan ilmiy-texnikaviy faoliyat tushuniladi. Bu faoliyat standartlarni va texnikaviy talablarni ishlab chiqishda, nashr etishda va tatbiq qilishda namoyon bo‘ladi. Standartlashtirishning muhim natijalari odatda mahsulot, jarayon va xizmatlarning belgilangan vazifaga mos kelishi, savdodagi g’ovlarni bartaraf qilish hamda ilmiy-texnikaviy hamkorlikka ko‘maklashishda namoyon bo‘ladi.

Standartlashtirish obyekti sifatida standartlashtiriladigan predmet (mahsulot, jarayon, xizmat) tushuniladi. "Standartlashtirish ob’ekti" tushunchasini keng ma’noda ifodalash uchun "mahsulot, jarayon, xizmat" iboralari qabul qilingan bo‘lib, buni har qanday materialga, tarkibiy kismlarga, asbob-uskunalarga, tizimlarga, ularni mosligiga, qonun-qoidasiga, ish olib borish uslubiga, vazifasiga, usuliga yoki faoliyatiga teng darajada daxldor deb tushunmoq lozim. Standartlashtirish har qanday obyektning muayyan jihatlari (xususiyatlari) bilan cheklanishi mumkin. Standartlashtirish obyekti sifatida xizmat–xalqqa xizmat qilishni (xizmat shartlarini qo‘shib) va korxona hamda tashkilotlar uchun ishlab chiqarish xizmatini o‘z ichiga oladi.

Uyg‘unlashtirilgan standartlar (ekvivalent standartlar): Bir va ayni shu obyektga tegishli bo‘lgan va standartlashtirish bilan shug’ullanuvchi turli idoralar tomonidan tasdiqlangan standartlar bo‘lib, ular mahsulotlar, jarayonlar va xizmatlarning o‘zaro almashuvini, ushbu standartlarga muvofiq takdim etiladigan axborot yoki sinovlar natijalarining o‘zaro tushunilishini tahminlaydi. Xalqaro, mintaqaviy, milliy standartlashtirish idoralari mavjud. Xalqaro standartlashtirish faoliyatida barcha mamlakatlarning tegishli idoralari erkin holda ishtirok etishi mumkin.

Mintaqaviy standartlashtirish deganda dunyo miqyosida birgina jug’rofiy yoki iqtisodiy mintaqaga qarashli mamlakatlarning tegishli idoralari uchun erkin holda ishtirok etishlari mumkin bo‘lgan standartlashtirish tushuniladi.

Milliy standartlashtirish - bu muayyan bir mamlakat doirasida o‘tkaziladigan standartlashtirish faoliyatidir. Standartlashtirish har xil faoliyat turlari va uning natijalariga daxldor qoidalar, umumiy qonun qoidalar yoki tavsiflarni o‘zida qamrab olgan me’yoriy hujjat hisoblanadi."Me’yoriy" hujjat atamasi standartlar, texnikaviy shartlar, shuningdek umumiy ko‘rsatmalar, yo‘riqnomalar va qoidalar tushunchasini ham o‘z ichiga qamrab oladi.

Standartlashtirish maqsadlari ko‘p qirrali bo‘lib, ular asosan quyidagilardan iborat: birxillashtirish (har xillikni boshqarish), qo‘llanishlilik, moslashuvchanlik, o‘zaro almashuvchanlik, sog’liqni saqlash, xavfsizlikni tahminlash, tashqi muhitni asrash, mahsulotni himoyalash, o‘zaro tushunishlikka erishish, savdodagi iqtisodiy ko‘rsatkichlarni yaxshilash va boshqalar. Bir maqsadning amalga oshishida bir vaqtda boshqa maqsadlarning ham amalga oshishi mumkin. Standartlashtirishda mahsulotning vazifasiga muvofiqligi deganda belgilangan sharoitlarda muayyan vazifalarini buyum, jarayon yoki xizmatlar tomonidan bajarish qobiliyati tushuniladi. Moslashuvchanlik esa, ma’lum sharoitlarda belgilangan talablarni bajarish uchun nomaqbul tahsir ko‘rsatmasdan mahsulot, jarayon yoki xizmatlarni birgalikda qo‘llanishiga yaroqliligi deb tushuniladi. O‘zaro almashuvchanlik – bir xil talablarni bajarish maqsadida bir buyum, jarayon, xizmatdan foydalanish o‘rniga boshqa bir buyum, jarayon, xizmatning yaroqliligidan iborat. Har xillikni boshqarish (unifikatlashtirish yoki birxillashtirish) deb, muayyan ehtiyojini qondirish uchun zarur bo‘lgan eng maqbul o‘lchamlarni yoki mahsulot, jarayon va xizmat turlarini tanlashga aytiladi.

Standartlashtirishning asosiy tamoyillari

Standartlashtirish bo‘yicha umumiy maqsad - bu mahsulot sifati, protsesslar va xizmatlar bo‘yicha Davlat va iste’molchilarning qiziqishlarini qimoyalashdir. Standartlashtirishning ushbu umumiy maqsadidan kelib chiqqan holda uning asosiy tamoyillarini keltiramiz:

1. **Tizimlilik va komplekslilik**. Komplekslilik – bu murakkab sistemadagi barcha elementlarning muvofiqligidir.

2. **Dinamiklik va standartlashtirishning o‘suvchanligi**. Standartlar fan va texnikadagi yuz berayotgan o‘zgarishlarga «adaptatsiya», yahni moslashishi kerak. Standartlashtirish bo‘yicha xalqaro tashkiloti «ISO»ning standartlari o‘suvchandir[4].

3. **Standartlashtirishning samaradorligi** – bu normativ hujjat qo‘llanilganda iqtisodiy yoki ijtimoiy samara berishidir, yahni standartlashtirishga sarflangan bir so‘mning o‘n so‘m foyda keltirishidir.

4. **Standartlarni ishlab chiqishning ustivorligi** - bu ustivorlik tovar va xizmatlarning o‘zaro muvofiqligi, almashuvchanligi va xavfsizligini tahminlashdir.

5. Uyg’unlashtirish tamoyili - bu xalqaro savdoda to‘siq bo‘lmaydigan uyg’unlashgan standartlarni ishlab chiqishdir.

6. Afzallik tamoyili. Bu tamoyil afzalroq sonlar qatoridan foydalanishga asoslangan bo‘lib, ulardan buyum, uskunalarning nomenklaturasini kamaytirishda foydaniladi.

Standartlashtirishning usullari quyidagilardir:

1. Unifikatsiya:
   a) sistemalashtirish;
   b) klassifikatsiya yoki tasniflash;
2. Simplifikatsiya;
3. Tipizatsiya yoki turlarga ajratish;
4. Selektsiya yoki tanlash;
5. Optimallashtirish (оптимизация);
6. Agregatlash (агрегатирование).

1. Unifikatsiya - standartlashtirish shakli bo‘lib, ikki yoki undan ortiq hujjatlarni (texnik shartlarni) birlashtirishdir va bu hujjat bilan reglamentlangan buyumni iste’molda o‘zaro almashtirish mumkin bo‘lganligi, albatta, nazarda tutilgan bo‘lishi kerak. Boshqacha qilib aytganda, unifikatsiya- bu bir xil funktsional yunalishda bo‘lgan bir turdagi detalh va agregatlar sonini ratsional qisqartirish bo‘yicha

[4] ISO. "ISO/IEC Directives and ISO supplement". Archived from the original on 23 April 2005

faoliyatdir[5]. Mahsulotni unifikatsiyalash koeffitsienti K_n mavjud bo‘lib, u quyidagi ifoda bilan keltiriladi:

$$K_n = n\text{-}n_0 / n \times 100.$$

Bu yerda: n-buyumdagi detallar soni bo‘lib, unga standart, unifikatsiyalashgan hamda umummashinasozlik, tarmoqlararo va tarmoqda qo‘llaniladigan detallar kiradi. n_0-optimal detallar soni;

a) sistemalashtirish- bu buyumlar, hodisalar va tushunchalarni aniq tartib va ketma-ketlikda joylash bo‘lib, foydalanishga qulay bo‘lgan tizimni yaratadi. Masalan, bu tizimdan entsiklopedik va politexnik ma’lumotnomalar va bibliografiyalarda foydaniladi. Mashinalarning o‘lchamlari va parametrlarini hamda ularning detallari va kismlari sistemalashtirish uchun afzalroq sonlar qatori foydalaniladi.

b) klassifikatsiya- bu buyum, hodisa va tushunchalarni klasslar, razryadlarga, ularning umumiy belgilariga bog’liq holda joylanishidir. Masalan, UDK (универсальная десятичная классификация) – gumanitar va texnikaviy adabiyotlarni indekslar bilan rubrikalashning xalqaro tizimi sifatida qabul qilingan. Masalan, UDK62- texnika, UDK621- umumiy mashinasozlik va elektronika, UDK62,3- elektrotexnika va boshqalar.

2. Simplifikatsiya - standartlashtirishning shakli bo‘lib, buyumlarning turi va boshqa turli ko‘rinishlarining mikdorini qarayotgan vaqtdagi mavjud talablargacha kamaytirishdir. Simplifikatsiya obyektlariga hech qanday texnikaviy mukammallashtirishlar kiritilmaydi.

3. Tipizatsiya - bu ko‘p sonli va ko‘p turdagi mashina, pribor, instrument va boshqalardan o‘z sifati bo‘yicha ratsional bo‘lgan va o‘z tartibida umumiy
detallar bo‘lganlarini tanlashdir. Maqsadi buyumlarning ko‘pxilligini likvidatsiya qilishdir.

4. Selektsiya. Standartlashtirish obyektlarini tanlash**.** Bu ishlab chiqarishning davom ettirilishi maqsadga muvofiq deb topilgan konkret obyektlarni tanlash.

5. Optimallashtirish - bu optimal bosh parametrlarni hamda sifat va tejamkorlikning boshqa barcha kiymatlarini topishdir.

6. Agregatlash - bu alohida unifikatsiyalashgan standart uzellaridan mashina, pribor va jihozlarni yaratish usulidir.

Standartlarning bir necha turlari mavjud:

- Lug’at standartlari, masalan, lug’atlar, belgilar va ramzlar;
- Asosiy standartlar, masalan, o‘lchamlar bo‘limlari kabi;

[5] The ISO directives are published in two distinct parts: "ISO/IEC Directives, Part 1.

– Mahsulot standartlari, o‘lchamlar, bajarish uchun xususiyatlar, sog’liqni saqlash, xavfsizlik, atrof-muhitni muhofaza qilish va hujjatlashtirishni qamrab oladi;

7. Tekshirish, test usullari va tahlil qilish uchun standartlar.

4.3. Metrologiya bo‘yicha asosiy atama va tushunchalar

Metrologik me’yorlar va qoidalarning xalqaro va tarmoqlararo ahamiyatini hisobga olgan holda o‘zaro tushunish va harakatlarni kelishishda metrologik atamalarning birliligi va qabul qilingan tushunchalarga aniq rioya qilish muhim ahamiyat kasb etadi. O‘zbekiston Respublikasining "Metrologiya to‘g’risida" Qonunida, O‘z O‘DT ning asos bo‘luvchi hujjatlari O‘z DSt 8.010:2002, O‘z DSt 8.010.2:2003, O‘z DSt 8.010.3:2004, O‘z DSt 8.010.4:2002 da va aloqa va axborotlashtirish sohasida foydalaniladigan “Tst 45.025.2000. Metrologik ta’minot. Atamalar.” tarmoq standartida metrologiya bo‘yicha quyidagi asosiy atama va tushunchalar keltirilgan:

Metrologiya – O‘lchashlar, ularning birliligini ta’minlash metodlari va vositalari va talab etilgan aniqlikka erishish usullari to‘g’risidagi fan.

O‘lchashlar birliligi – O‘lchashlarning natijalari qonunlashtirilgan birliklarda ifodalangan va o‘lchashlarning xatoliklari berilgan ehtimollik bilan ma’lum bo‘lgan holat.

O‘lchash – maxsus texnik vositalar yordamida fizik kattaliklar qiymatlarini tajriba yo‘li bilan topish.

O‘lchash vositasi – normalashtirilgan metrologik tavsifga ega bo‘lgan o‘lchash asbobidir. O‘lchash vositasi, o‘z navbatida, o‘lchov, o‘lchash o‘zgartirgichlari, o‘lchov asboblari, o‘lchash axborot tizimi va o‘lchash qurilmalari kabi turkumlarga bo‘linadi.

O‘lchash asbobi – kuzatuvchi idrok qilishi uchun qulay shakldagi o‘lchov informasiyasi signalini ishlab chiqishga xizmat qiladigan o‘lchash vositasiga aytiladi.

O‘lchov – deb, berilgan o‘lchamli fizik kattaliklarni qayta tiklash uchun mo‘ljallangan o‘lchash vositasiga aytiladi. O‘lchovlar to‘plami deb, maxsus tanlangan, faqat alohidagina emas, balki turli birikmalarda turli o‘lchamli qator bir nomli kattaliklarni qayta o‘lchash maqsadida qo‘llaniladigan o‘lchovlar majmuiga aytiladi[6].

Birlik etaloni – Fizik kattalikning o‘lchamini boshqa o‘lchash vositasiga berish maqsadida fizik kattalik birligining o‘lchamini qayta

[6] Toru Yoshizava, Handbook of optical metrology, 2008.

tiklash va saqlash uchun mo‘ljallangan o‘lchashlar vositasi.

Davlat etaloni – O‘zbekiston Respublikasi hududida kattalik birligining o‘lchamini o‘rnatish uchun milliy idora vakilining qarori bilan boshlang’ich ztalon sifatida tan olingan etalon.

O‘lchash o‘zgartkichi - o‘lchashga doir axborotni uzatish, o‘zgartirish, ishlov berish va saqlash uchun qulay bo‘lgan, ammo kuzatuvchi bevosita idrok qilishi mumkin bo‘lmaydigan shakldagi signalni ishlab chiqish uchun xizmat qiladigan o‘lchash vositasidir. Namuna o‘lchash vositalari ishchi o‘lchash asboblarini tekshirish va ularni o‘zlari bo‘yicha darajalashga xizmat qiladi.

Etalonlar deb, fan va texnikaning eng yuksak saviyasida aniqlik bilan ishlangan namunaviy o‘lchovlarga aytiladi.

O‘lchov birligi o‘lchash natijasi ko‘rsatilgan birlikda ifodalangan va o‘lchash xatoligi berilgan ehtimollikda ma’lum bo‘lgan o‘lchash holatidir.

O‘lchash aniqligi – bu o‘lchash natijalarini va o‘lchanayotgan kattalik haqiqiy qiymatining mos kelish darajasidir.

O‘lchash xatoligi o‘lchash natijasining o‘lchanayotgan kattalikning asl qiymatidan farqlanishidir.

Fizikaviy kattalikning asl qiymati xatoliklardan xoli bo‘lgan qiymatdir. O‘lchanayotgan kattalikning haqiqiy qiymati yo‘l qo‘yilgan xatoliklar ta’sirida olingan natijalar qiymatidir. O‘lchash ob’ekti u yoki bu fizik kattalikdir.

Qonunlashtiruvchi metrologiya – bu metrologiyaning1 bir qismi bo‘lib metrologiya bo‘yicha milliy organ tomonidan amalga oshiriladigan faoliyatga taalluqli va birliklar, o‘lchashvositalari, o‘lchash laboratoriyalariga doir davlat talablariga ega.

Metrologiya xizmati – Davlat idoralari va yuridik shaxslarning metrologik xizmatlari tarmoqlari hamda ularning o‘lchashlar birliligini ta’minlashga yo‘naltirilgan faoliyati.

O‘lchash vositalarining qiyoslanishni olib borish uchun metrologik xizmatni akkreditlash – bu qiyoslashishlarini bajarishga davlat tomonidan, davlatning ishonchli vakili tomonidan rasmiy tan olinishi.

O‘lchash vositasini qiyoslash usuli – bu qiyoslash sxemasi bo‘yicha yuqoridagi o‘lchash vositalaridan quyidagi o‘lchash vositalariga birlikning o‘lchamini uzatish usuli.

O‘lchash vositalarining qiyoslash natijalarini rasmiylashtirish - bu o‘lchash vositalarini qiyoslash natijalari bo‘yicha rasmiy hujjatni tuzish va o‘lchash vositasini yaroqliligini ko‘rsatish.

O‘lchash vositalarini taqqoslash – bu sistematik xatoliklarni aniqlash uchun o‘lchash vositasini etalon yoki o‘sha turdagi namuna o‘lchash vositasiga solishtirish, ya’ni o‘lchovni o‘lchov bilan, priborni pribor bilan.

Bazaviy (asos) metrologiya xizmati – bu aloqa va axborotlashtirish sohasidagi xizmat bo‘lib, biriktirilgan xo‘jalik yurituvchi sub’ektlarning metrologik ta’minot masalalari bo‘yicha ish faoliyatini muvofiqlashtiruvchi xizmat.

Davlat metrologiya nazorati – bu o‘lchash vositalarining turi va qiyoslanishi, sotilishi va ularning prokatini lisenziyalash bo‘yicha davlat metrologiya xizmati organi tomonidan amalga oshiriladigan faoliyatdir.

Davlat metrologiya tekshiruvi - bu davlat metrologiya xizmati organi tomonidan amalga oshiriladigan metrologiya qoidalariga rioya qilinishi tekshirish maqsadidagi faoliyatdir.

O‘lchanadigan kattalik - o‘lchashga tortiladigan kattalik.

O‘lchash vositasini kalibrlash - bu kalibrlash laboratoriyasi tomonidan o‘lchash vositasining metrologik xarakteristikasi haqiqiy qiymatlarini va qo‘llanilishga yaroqliligini aniqlash va tasdiqlash maqsadidagi muolajalar majmuidir.

O‘lchash vositalarini qiyoslash – O‘lchash vositalarining o‘rnatilgan texnik talablarga muvofiqligini aniqlash va tasdiqlash maqsadida davlat metrologik xizmat idoralari (boshqa vakolatlangan idoralar, tashkilotlar) bajaradigan amallar majmui.

O‘lchash vositalarini tayyorlash (ta’mirlash, sotish, ijaraga berish) ga lisenziya – Ko‘rsatilgan faoliyat turlari bilan shug’ullanishga huquqini tasdiqlovchi, yuridik va jismoniy shaxslarga davlat metrologik xizmat idoralari tomonidan beriladigai hujjat.

O‘lchash vositalarini metrologik attestatlash – donalab ishlab chiqarilgan (yoki O‘zbekiston hududiga donalab keltirilgan) o‘lchash vositalarining, ularning xossalarini sinchiklab tadqiq etish asosida, qo‘llanishga huquqli ekanligini metrologik xizmat tomonidan tan olish.

Metrologik xizmatlar, markazlar, laboratoriyalarni akkreditlash –metrologik xizmatlar, markazlar, laboratoriyalarning o‘rnatilgan akkreditlash doirasida o‘lchashlar birliligini ta’minlash bo‘yicha ishlarni o‘tkazishga huquqligini rasmiy tan olish.

Yuridik shaxslarning metrologik xizmatlarini o‘lchash

vositalarini kalibrlash huquqiga akkreditlash – yuridik shaxslar metrologik xizmatlarining o‘rnatilgan doirada o‘lchash vositalarini kalibrlashni o‘tkazish huquqini rasmiy tan olish.

O‘lchashlarni bajarish metodikalarini metrologik attestatlash – O‘lchashlarni bajarish metodikasining unga qo‘yilgan metrologik talablarga muvofiqligini baholash va tasdiqlash maqsadida o‘tkaziladigan tadqiqot.

O‘lchashlarni bajarish metodikasi – O‘lchashlar natijalariga avvaldan ma’lum bo‘lgan xatolik bilan erishishni ta’minlaydigan ishlar va qoidalar majmui.

O‘lchash noaniqligi – Tekshirish- bu belgilangan talablarni to‘laqonli ta’minlash uchun obyektiv dalillar yordamida ko‘rikdan o‘tkazishni ta’minash.

4.4. Axborot texnologiyalari va kommunikasiya sohasida metrologik xizmat

O‘zbekiston Respublikasining «Metrologiya to‘g’risida»gi Qonuni (11-modda) tomonidan xo‘jalik yurituvchi sub’ektlarning metrologik xizmatlari o‘lchash birliligini ta’minlash va metrologik nazoratni amalga oshirish bo‘yicha ishlarni bajarish uchun zarur bo‘lgan hollarda tashkil etilishi belgilab qo‘yilgan. Milliy Standartlar Tuzilmasi (MST). Milliy darajada standartlarni tayyorlash, ular bilan bog’liq barcha masalalar MST orqali amalga oshiriladi. Ko‘pgina mamlakatlar MSTni tashkilotlar yoki institutlar deb yuritishadi.

Dunyo bo‘yicha ko‘plab ISO tashkiloti a’zolari o‘zlarining MSTsiga ega; biroq ko‘plab mamlakatlar MSTga ega emas, ushbu mamlakatlar ISO tashkilotining standartlari va qonunlaridan foydalanadi. Hozitgi kunda dunyo bo‘yicha 148 mamlakat ISO tashkiloti a’zolari hisoblanadi. Eng ko‘p rivojlangan mamlakatlar 1917 va 1925 yillar oralig’ida 15ta MST joriy etildi. Germaniya ular orasida birinchi bo‘lib o‘zining MSTsini tashkil etdi. Keyinchalik shu vaqtlar oralig’ida Buyuk Britaniya, Amerika Qo‘shma Shtatlari, Belgiya, Kanada, Niderlandiya, Shvetsariya va Austriya ham o‘zlarining MSTni tashkil etishdi. MST quiyidagi asosiy funksiyalarni o‘z ichiga oladi:

- Milliy standartlarni tayyorlash va chop etish;
- Sanoatda standartlarning amalga oshirilishini ta’minlash;
- Mahsulotlarni sertifikatlashtirish;

– Texnika masalalariga bog’liq bo‘lgan masalalar va bular milliy hamda xalqaro standartlar bilan o‘zaro muvofiqligini ta’minlash haqidagi ma’lulotlarni e’lon qilish;

– Standartlar bilan aloqador forumlarda va xalqaro faoliyatlarda mamlakat ishtirokini nazorat qilish.

Hozirgi vaqtda aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida o‘lchashlar birliligini ta’minlash tizimida o‘lchash vositalarini qiyoslash, kalibrlash, ta’mirlash va metrologik shahodatlash huquqiga ega bo‘lgan uchta akkreditlangan metrologiya xizmatlari mavjud:

– O‘zbekiston aloqa va axborotlashtirish agentligi UNICON.UZ markazi qoshidagi Asos metrologiya xizmati,

– Radioaloqa, radioeshittirish va televidenie markazi

– «O‘zbektelekom» AK «Shaharlararo aloqa korxonasi» filialidagi metrologiya xizmatlari.

UNICON.UZ markazi qoshidagi Asos metrologiya xizmatining qiyoslash va ta’mirlash laboratoriyasi sohaviy hisoblanadi va u sohadagi barcha xo‘jalik yurituvchi sub’ektlar uchun ta’mirlash va qiyoslash ishlarini o‘tkazish huquqiga ega. Oxirgi ikkita xizmat akkreditlash sohasiga muvofiq xo‘jalik yurituvchi sub’ektlarning ichki ehtiyojlari uchun zarur bo‘lgan o‘lchash vositalarini ta’mirlash va qiyoslash huquqiga egadirlar. Ushbu metrologik xizmatlari o‘z faoliyatlarini amalga oshirishlari uchun namunaviy o‘lchash vositalari, yordamchi uskunalar va normativtexnik hujjatlar bilan ta’minlanganlar.

Hozirgi vaqtda, metrologiya xizmatlari yo‘q bo‘lgan (o‘lchash vositalarining to‘plami kam bo‘lganligi sababli), aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida xo‘jalik yurituvchi sub’ektlarida o‘lchash vositalari holati, tegishli hujjatlarda belgilangan vazifa va majburiyatlari uchun javobgarlar tayinlangan. Barcha metrologik xizmatlarga va o‘lchash vositalari holati uchun javobgar shaxslarga asosiy talablar - bu o‘lchash vositalarini, qiyoslash va ta’mirlash ishlarini o‘z vaqtida o‘tkazish. Xo‘jalik yurituvchi sub’ektlarning metrologik xizmatlariga qiyoslash laboratoriyalari bilan birqatorda davlat metrologiya nazorati va tekshiruviga tegishli bo‘lgan sohadan tashqaridagi o‘lchov vositalari uchun kalibrlash laboratoriyalarini tuzish maqsadga muvofiqdir.

Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida o‘lchash vositalarini kalibrlashning tashkiliy va texnik asoslarini joriy qilish o‘lchash vositalari holatini nazorat qilishni osonlashtirishga

imkon beradi. «O‘zbektelekom» AK, «O‘zbekiston pochtasi» DAK, uyali aloqa kompaniyalari, Internetning yirik operator va provayderlarining qayta tashkil qilinishi sababli ularning tarkibiy bo‘linmalari va filiallarini metrologik ta’minlashda muammolar tug’ildi. Tabiiyki, akkreditlashning qisqa sohasi bilan chegaralangan mavjud uchta metrologik xizmatlarning faoliyati yangi tuzilgan tarkiblar va bo‘linmalarni ta’mirlash va qiyoslash ishlari bilan to‘liq ta’minlay olmaydi. Bu muammolarni hal qilish uchun va namunaviy o‘lchash vositalarini ta’mirlash va qiyoslash laboratoriyalarini yuqori texnologiya uskunalari bilan jihozlash yo‘li bilan mavjud metrologiya xizmatlarining akkreditlash sohasini izchillik bilan kengaytirish va mutaxassislarni yangi texnika hamda texnologiyalar bilan ishlashning zamonaviy uslublari va ishlash malakalarini oshirish uchun muntazam ravishda o‘qitish zarur.

UNICON.UZ markazi qoshidagi Asos metrologiya xizmati aloqa va axborotlashtirish sohasidagi xo‘jalik yurituvchi sub’ektlarning metrologik xizmatlari ishlarini muvofiqlashtiradi. Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasining xo‘jalik yurituvchi sub’ektlarning metrologiya xizmatlari o‘lchashlar birliligi va talab etilgan aniqligini (o‘lchashlarni bajarish uslublarini ishlab chiqish va attestatlash, normativ hujjatlar loyihalarining metrologik ekspertizasi, loyihalash, konstruktorlik va texnologik hujjatlashtirish hamda boshqa turdagi ishlar bo‘yicha) ta’minlash sohasida muayyan faoliyatni amalga oshirishda texnik vakolatliligini tan olishga, ixtiyoriy ravishda akkreditasiyadan o‘tishi mumkin.

Metrologik ta’minotning tarkibiga O‘zbekiston aloqa va axborotlashtirish agentligi, UNICON.UZ markazi qoshidagi Asos metrologiya xizmati, Telekommunikasiyalar va pochta aloqasi sohasidagi standartlashtirish bo‘yicha Texnika qo‘mitasi, Davlat aloqa inspeksiyasi, metrologik xizmatlar va xo‘jalik yurituvchi sub’ektlari o‘lchash vositalarining holati uchun mas’ul shaxslar kiradi. Metrologik ta’minot tizimi O‘zbekiston Respublikasi Davlat o‘lchashlar birliligini ta’minlash tizimi bilan, O‘zbekiston Respublikasi standartlashtirish Davlat tizimi, standartlashtirish va sertifikatlashtirish soha tizimlari bilan o‘zaro hamkorlik qiladi.

Tizimning boshqa davlatlarning o‘lchashlar birliligini ta’minlash tizimlari va xalqaro organlar bilan o‘zaro hamkorligi O‘zbekiston Respublikasining amaldagi qonun hujjatlariga, O‘zbekiston aloqa va axborotlashtirish agentligi, «O‘zstandart» agentligining normativ hujjatlariga muvofiq tartibga solinadi. Davlat metrologik nazorati va

tekshiruvi sohasida qo‘llanilayotgan o‘lchash vositalarining Davlat Reestrini, metrologik ta’minot bo‘yicha standartlar Davlat Reestrini, akkreditlangan metrologik xizmatlar va metrologik laboratoriyalar Davlat Reestrini yuritish, Tizim bo‘yicha tashkiliyuslubiy hujjatlarning kelishuvi, boshqa davlatlarning o‘lchashlar birliligini ta’minlash Davlat tizimlari bilan o‘zaro hamkorlik qilish, shu jumladan, sinovlar natijalarini o‘zaro tan olish, tur tasdiqlanishining sertifikatlari va metrologik shahodatlash masalalari bo‘yicha, shuningdek o‘lchash vositalarini qiyoslash uslublarini, o‘lchash vositalari ustidan davlat metrologik nazorati va tekshiruvini amalga oshirish vazifalari «O‘zstandart» agentligi zimmasiga yuklatilgan. Metrologik xizmatlarni ko‘rsatish bo‘yicha respublika markazi:

– o‘lchash vositalari turini tasdiqlash bo‘yicha sinovlarni o‘tkazish;

– o‘lchash vositalarini metrologik shahodatlash, o‘lchash vositalarini qiyoslash;

– metrologik xizmatlari, markazlari, laboratoriyalari o‘lchash vositalarining sinovlari va qiyoslashlarini o‘tkazish huquqini beruvchi akkreditlashni tashkil etish va o‘tkazish;

– o‘lchashlarni bajarish uslublari va metrologik faoliyatning boshqa muayyan turlarining davlat metrologik nazoratini amalga oshiradi.

Standartlashtirish, metrologiya va sertifikatlashtirish ilmiy-tadqiqot instituti belgilangan ixtisosliklar bo‘yicha kadrlar tayyorlash (qayta tayyorlash)ni ta’minlaydi va qonunlashtiruvchi metrologiya bo‘yicha ishlab chiqilayotgan soha normativ hujjatlarining kelishuvida qatnashadi, metrologik xizmatlari va sinov laboratoriyalarini akkreditlash bo‘yicha hujjatlar ekspertizasini o‘gkazadi. Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida metrologik ta’minot masalalari O‘zbekiston aloqa va axborotlashtirish agentligi tomonidan tartibga solinadi va muvofiqlashtiriladi.

O‘zbekiston Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari davlat qo‘mitasining asosiy vazifalari quyidagilar hisoblanadi:

– Metrologiya qonun hujjatlari, o‘lchov vositalari turining attestasiyasi va tan olinishi masalalari bo‘yicha «O‘zstandart» agentligi bilan o‘zaro hamkorlik, Tizim tarkibini shakllantirish va uning qatnashchilarining faoliyatini koordinatlash bo‘yicha ishlarni tashkil etish;

– Aloqa sohasidagi mintaqaviy hamdo‘stlik va boshqa xalqaro tashkilotlar bilan Tizimning qoida va me’yorlarini rivojlantirish hamda uyg’unlashtirish masalalari bo‘yicha o‘zaro hamkorlik qilish;

– umumsoha xarakteridagi masalalarni hal etish, aloqa va axborotlashtirish sohasida o‘lchashlar birliligini ta’minlashda texnik siyosatni ishlab chiqish;

– soha metrologik ta’minotining qonun hujjatlari bazasini ishlab chiqish bo‘yicha ishlarni tashkil qilish.

O‘zbekiston aloqa va axborotlashtirish Agentligi qoshidagi standartlashtirish bo‘yicha Texnik qo‘mita quyidagilarni ishlab chiqadi:

– metrologiya qonun hujjatlari hamda aloqa va axborotlashtirish sohasi xo‘jalik yurituvchi sub’ektlarining metrologik ta’minoti borasidagi strategiyani;

– xalqaro standartlar talablari bilan uyg’unlashtirilgan normativ hujjatlarni ishlab chiqish, amaldagi normativ hujjatlarga o‘zgartirishlar kiritish yoki ularni bekor qilish, chet el standartlaridan Tizimni takomillashtirish maqsadida foydalanish bo‘yicha tavsiyalarni tayyorlaydi;

– aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida metrologik xizmatlarning tashkiliy sxemasini takomillashtirish va shakllantirish bo‘yicha tavsiyalarni tayyorlaydi.

O‘zbekiston Respublikasi Aloqa vazirligi (hozirda O‘zbekiston aloqa va axborotlashtirish agentligi)ning 1997 yil 11 iyuldagi 225-son buyrug’iga asosan UNICON.UZ markazi qoshida Asos metrologiya xizmati tashkil qilindi. Asos metrologiya xizmati Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasining xo‘jalik yurituvchi sub’ektlarini metrologik ta’minoti vazifalarini amalga oshirish bo‘yicha ishlarga ilmiy-texnik va tashkiliyuslubiy rahbarlikni amalga oshiradi. Asos metrologiya xizmati nizomga muvofiq o‘lchash vositalarining soha reestrini yuritadi, «Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida qo‘llash uchun tavsiya qilingan o‘lchash vositalari katalogi»ni har yili yangilaydi.

Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasidagi Radioaloqa, radioeshittirish va televidenie markazi qoshida faoliyat ko‘rsatmoqda. Qonun hujjatlari va normativ aktlar, davlat va soha standartlari talablarini bajarish va ularga rioya qilish, lisenziya shartlari va aloqa va axborotlashtirish sohasida taqdim etilayotgan xizmatlar sifatini ta’minlash bo‘yicha Davlat nazorat organi bo‘lib Davlat aloqa inspeksiyasi hisoblanadi. Tizim ishtirokchilarining asosiy vazifalari ular

to‘g’risidagi belgilangan tartibda kelishilgan va tasdiqlangan nizomlarga muvofiq aniqlanadi.

4.5. Metrologiya bo‘yicha xalqaro tashkilotlar

1906 yil elektrotexnika sohasidagi xalqaro darajadagi ishlarni standartlashtirish maqsadida dastlabki xalqaro tashkilot tashkil etildi, ushbu tashkilotga 15 ta mamlakat vakillari ishtirok etdi, hamda tashkilotni xalqaro elektrotexnika komissiyasi (IEC) deb nomlandi. 1946 yilda Londonda uchrashgan 25 davlat delegatsiyasi “standartlar sanoatining umumlashuvini hamda o‘zaro muvofiqliligini ta’minlashga qaratilgan” barcha maqsadlarni o‘zida qamrab olgan yagona xalqaro tashkilot tizishga qaror qilishdi. Yangi tuzilgan tashkilotni Xalqaro Standartlashtirish Tashkiloti (ISO) deb nomlashdi, va ushbu tashkilot 23 fevral 1947 yildan rasman faoliyat yurita boshladi. ISO qisqartmasi Greekcha *isos* so‘zidan “tenglik” degan ma’noni beradi. Shuning uchun qaysi davlat yoki qanaqa tilda bo‘lishidan qat’iy nazar tashkilotning nomi qisqacha ISO tarzida yuritiladi. Hozirgi kunda 148 davlatning MSTsi ISO ning tarmog’i hisoblanadi, muvofiqlashtirish tizimining markaziy sekretariati Shvetsariyaning Geneva shahrida joylashgan. ISO nodavlat (nohukumat) tashkilot hisoblanadi[7].

MEK elektrotexnika, elektronika, radioaloqa, televidenie, telekommunikasiya, priborlar yaratish sohalari bo‘yicha shug’ullanadi. 1975 yildan boshlab MEK tavsiyalari xalqaro standartlar statusini oldi. MEK standartlari elektrotexnik jihozlar va elektron qurilmalarni eksport qilishda qo‘llaniladi. Hozirda 41 ta milliy komitetlar MEK ning a’zolari hisoblanadi. Bu mamlakatlarda yer kurrasining 80 % aholisi yashab, dunyodagi ishlab chiqarilayotgan elektr quvvutining 95 % ning iste’molchilari hisoblanadi. MEK ning oliy rahbar idorasi MEK kengashi hisoblanadi. Unda eng yuqori lavozimida MEK prezidenti turadi va u har 3 yilda saylanadi. MEKning ishchi tili bo‘lib, rus, ingliz va fransuz tillari hisoblanadi.

Qonunlashtiruvchi metrologiya bo‘yicha xalqaro tashkilot (MOZM) 1956 yildan beri mavjud va hozirda 50 dan ortiq qonunlashtiruvchi metrologiya bo‘yicha Xalqaro hamkorlikni ta’minlash mamlakatlarni o‘z ichiga oladi. Uning asosiy vazifasi xalqaro masshtabda o‘lchashlar birliligini ta’minlash, o‘lchash xatoliklarni baholash, o‘lchash usullari, terminologiya va shartli belgilashlar bo‘yicha tavsiya ishlab chiqishdir.

[7] Andrew Sabak and P.Makinen, Adobe Acrobat Document. Translated by,2012

Uning oliy organi – Xalqaro konferensiyadir. Bu konferensiya Xalqaro miqyosda o‘lchashlar birliligini ta’minlash amaliyoti bo‘yicha MBMV - o‘lchov va tarozi bo‘yicha xalqaro byuro hisoblanadi. MOZM organlari quyidagilar:

1. Qonunlashtiruvchi metrologiyaning xalqaro konferensiyasi
2. Qonunlashtiruvchi metrologiyaning xalqaro komiteti
3. Qonunlashtiruvchi metrologiyaning xalqaro byurosi

Bundan tashqari MOZM tarkibida ishchi-texnik guruhlar ham faoliyat ko‘rsatadi. Qonunlashtiruvchi metrologiyaning xalqaro konferensiyasi har yili bir marta chaqiriladi. Qonunlashtiruvchi metrologiyaning Xalqaro komiteti MOZM ning rahbar organi hisoblanadi. MOZM ning shtab-kvartirasi Parij shaxrida Jenevada esa ISO va MEK tashkilotlarning sekretariatlari joylashgan. MKZM-prizidenti va ikki vise-prizidenti olti yil mudatga saylanadi. MKZM-majlisi har ikki yilda chaqiriladi. MOZM ning ijro organi bo‘lib MBZM (qonunlashtiruvchi metrologiya xalqaro byurosi) hisoblanadi. Byuro shtati MOZM ga a’zo davlatlar a’zolik badali hisobiga saqlanib turadi. Byuro MOZM ning sekretariati vazifasini bajaradi, markaz hisoblanadi hamda MKZM majlisi va konferensiyalarini tayyorlaydi. Sekretariat ma’ruzachilardan tashqari “Ishchi guruhi” deb nomlangan guruh har bir mavzu bo‘yicha tavsiyalar loyihasini ishlab chiqadi va guruh a’zolari bilan kelishiladi. Milliy idoralarning faoliyatini muvofiqlashtirish, savdoda texnik to‘siqlarni bartaraf etish uchun 1992 - yilda MDH mamlakatlarining (Boltiq bo‘yi mamlakatlaridan tashqari) standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha Davlatlararo kengashi (DAK) tuzildi.

MDH mamlakatlari hukumatlarining boshliqlari 13 mart 1992-yilda standartlashtirish, metrologiya va sertifikatlashtirish sohasida kelishilgan siyosatni olib borish to‘g’risida Bitimga imzo chekdi. Bu Hamkorlik davlatlarining standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha milliy idoralarning imkoniyatlarini va boyliklarini birlashtirishga, ilgari to‘plangan tajribalar va me’yoriy hujjatlardan birgaliqda foydalanish va ularni takomillashtirishga, shuningdek faoliyatning bu sohalarida yagona texnik siyosatni amalga oshirishga imkon berdi. DAK ning standartlashtirish, metrologiya va sertifikatlashtirish masalalari bo‘yicha muvofiqlashtiruvchi idora sifatida ishlari MDH da quyidagilarni ta’minlashga qaratilgan:

– yagona me’yoriy baza - davlatlararo standartlar, tasniflagichlar va boshqa me’yoriy hujjatlarni qo‘llanish va rivojlantirish;

– yagona etalon baza va o‘lchashlar birliligini ta’minlash tizimlarini shu jumladan, vaqt va chastotalar, moddalar va materiallarning tarkibi va xossalariga oid standart ma’lumotnoma ma’lumotlari davlatlararo xizmatlarini shakllantirish;

– mahsulot va xizmatlarni sinash va sertifikatlashtirish natijalarini o‘zaro tan olish.

DAK ning texnik siyosati a’zo-davlatlarning standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha milliy idoralari, ilmiy-texnikaviy komissiyalari (ishchi guruhlari) va standartlashtirish bo‘yicha davlatlararo TQ (texnik qo‘mita) tomonidan shakllantiriladi.

DAK faoliyatining asosiy yo‘nalishlari bo‘yicha ilmiy-texnikaviy komissiyalar yoki ishchi guruhlar, vaqt va chastotaning bir xil o‘lchanishini ta’minlash bo‘yicha hamkorlik to‘g’risida hukumatlararo Bitimni bajarish bo‘yicha vakolatli vakillarining Kengashi, shuningdek standartlashtirish bo‘yicha 230 dan ortiq davlatlararo TQ doimiy ishlamoqda. Hozirgi vaqtda Kengashning ishchi idorasi Minsqda joylashgan standartlashtirish bo‘yicha Byurodan iborat. Kengashni rotasiya asosida DAK a’zo mamlakatlarining standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha milliy idoralarining rahbarlari boshqaradi.

Kengash davlatlararo stavdartlashtirish, metrologiya va sertifikatlashtirish sohasida qator hukumatlararo bitimlarni tayyorladi va bular MDH mamlakatlari hukumat boshliqlarining majlislarida qabul qilingan. Bunday bitimlar jumlasiga quyidagilar kiradi:

– Standartlashtirish, metrologiya va sertifikatlashtirish sohasida kelishilgan siyosatni o‘tkazish to‘g’risida Bitim (13.06.1992, Moskva);

– Vaqt va chastotani o‘lchash bir xilligini ta’minlash bo‘yicha hamkorlik to‘g’risida Bitim (09.10.1992, Bishkek);

– Qiyoslash va metrologik attestatlash maqsadida chegaradan olib o‘tiladigan me’yoriy hujjatlar, etalonlar, o‘lchash vositalari va standart namunalarni olib o‘tishga bojxona to‘lovlari, soliqlardan va maxsus ruxsatnomalarni berishdan ozod qilish to‘g’risida Bitim (10.02.1995, Almati);

– O‘zaro etkazib beriladigan mahsulotga mehnat muhofazasi bo‘yicha kelishilgan me’yorlar va talablarni ishlab chiqish va rioya qilish tartibi to‘g’risida Bitim (12.04,1996, Moskva);

– Erkin savdo hududida texnik to‘siqlar bo‘yicha Bitim (20.06.2000, Moskva);

– MDH davlatlarida sayohat sohasida davlatlararo standartlarni va

sertifikatlashtirish tizimlarini ishlab chiqish va joriy etish bo‘yicha Konsepsiya.

MDH mamlakatlarida amaldagi texnik qonunlarni uyg’unlashtirish maqsadida DAK da model qonunlar ishlab chiqilgan:

– "Standartlashtirish to‘g’risida" (Parlamentlararo assambleyaning (PAA) 10-yalpi majlisida qabul qilingan);

– "O‘lchashlar birliligini ta’minlash to‘g’risida" (MDH PAA ning 11-yalpi majlisida qabul qilingan).

DAK doirasida quyidagi bitimlar tuzilgan va bajarilmoqda:

– Sertifikatlashtirish bo‘yicha ishlarni o‘tkazish va o‘zaro tan olish prinsiplari to‘g’risida (04.06.1992, Krasnodar);

– Davlat sinovlari va xilini tasdiqlash, metrologik attestatlash, o‘lchash vositalarini qiyoslash va kalibrlash natijalarini, shuningdek sinovlarni, o‘lchash vositalarini qiyoslash va kalibrlash laboratoriyalarini akkreditlash natijalarini o‘zaro tan olish to‘g’risida (06.10.1992, Toshkent);

– Moddalar va materiallar tarkibi va xossalarining standart namunalarini yaratish va qo‘llanish bo‘yicha hamkorlik to‘g’risida (06.10.1992, Toshkent);

– Moddalar va materiallarning fizik konstantalari va xossalari to‘g’risida ma’lumotlarni yaratish va ulardan foydalanish bo‘yicha hamkorlik to‘g’risida (06.10.1992, Dushanbe).

DAK standartlashtirish bo‘yicha xalqaro tashkilotlar (ISO, MEK) va YEvropa Ittifoqi (SYEN) standartlashtirish bo‘yicha tashkiloti, standartlashtirish bo‘yicha hududiy tashkilot tomonidan tan olingan va unga ISO va MEK da qabul qilingan qoidalarga muvofiq "Standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha YEvroOsiyo tashkiloti (EASC) nomi berilgan. Yuqoridagi nomi keltirilgan tashkilotlar bilan hamkorlik, axborot va me’yoriy hujjatlar bilan almashinish va o‘tkaziladigan tadbirlarda ishtirok etish to‘g’risida uzoq muddatli kelishuvlar imzolangan.

EASC imzolangan kelishuv (bitimlar) ga asosan xalqaro va evropa standartlarini davlatlararo standartlar orqali, EASC ning alohida a’zomamlakatlari esa, milliy standartlar orqali qo‘llanish huquqiga ega. Bu davlatlararo va milliy standartlarni ham xalqaro, ham evropa standartlari bilan yuqori darajada uyg’unlashtirishga yordam beradi. Bunday huqukdan EASC ning a’zo-davlatlari, bu tashkilotlarda a’zolik statusidan qati nazar, foydalanadi.

Hozirgi vaqtda MDH davlatlararo standartlarining jamg’armasida

19000 dan ortiq meyoriy hujjatlar bor. 1992 - yildan boshlab 3800 dan ortiq davlatlararo me’yoriy hujjatlar ishlab chiqilgan va qabul qilingan. Jamg’arma DAK ning standartlashtirish bo‘yicha Byurosi tomonidan, DAK a’zo-davlatlarning milliy idoralari bilan hamkorlikda olib boriladi. Davlatlararo me’yoriy hujjatlarni ishlab chiqishda ularning talablari xalqaro, hududiy va ilg’or milliy standartlar bilan uyg’unlashtiriladi. Bu MDH mamlakatlarining savdo-iqtisodiy va ilmiy-texnikaviy hamkorlikda texnik to‘siqlarni bartaraf etishga yo‘naltirilgan yagona me’yoriy-texnik ta’minotni saqlash uchun sharoit yaratadi, shuningdek DAK a’zodavlatlarda ishlab chiqariladigan mahsulotni xalqaro va evropa bozoriga chiqarishga ko‘maklashadi.

4-bob bo‘yicha xulosalar

Standartlashtirish, metrologiya va sertifikatlashtirish bo‘yicha asosiy maqsad mahsulotlar va xizmatlar sifati, havfsizligi va raqobatbardoshliligini ta’minlashdir. Standartlashtirishning afzallik jihatlarini shunday ta’riflash mumkin, standartlashtirish iqtisodiyotni maksimallikka erishishiga qaratilgan. Standartlar jamiyatning turli jabhalarida bir qancha afzalliklar olib kiradi. Xo‘jalik yurituvchi sub’ektlar tomonidan xalqaro, davlatlararo, milliy, mintaqaviy va davlat hamda tarmoq standartlarining keng qo‘llanishi uchun Standartlashtirish tayanch tashkiloti tomonidan normativ hujjatlar fondi yaratiladi va yuritiladi.

Standartlashtirish asosiy maqsadi mahsulotlar va xizmatlar sifati, havfsizligi va raqobatbardoshliligini ta’minlashdir. Uni shunday ta’riflash mumkin, standartlashtirish iqtisodiyotni maksimallikka erishishiga qaratilgan. Standart bu mahsulot yoki xizmatlar uchun talablar, qoidalar va ko‘rsatmalarni o‘zaro bir-biri bilan mutanosib kelishini ta’minlovchi hujjat hisoblanadi. Standart - bu ko‘pchilik manfaatdor tomonlar kelishuvi (konsensus) asosida ishlab chiqarilgan va ma’lum sohalarda eng maqbul darajali tartiblashtirishga yo‘naltirilgan hamda faoliyatning har xil turlariga yoki natijalariga tegishli bo‘lgan umumiy va takror qo‘llaniladigan qoidalar, umumiy qonun-qoidalar, tavsiyalar, talablar va usullarini o‘z ichiga olgan hamda, belgilangan va tan olingan idora tomonidan tasdiqlangan me’yoriy hujjatdir. Standartlashtirish maqsadlari ko‘p qirrali bo‘lib, ular asosan quyidagilardan iborat: birxillashtirish (har xillikni boshqarish), qo‘llanishlilik, moslashuvchanlik, o‘zaro almashuvchanlik, sog’liqni saqlash, xavfsizlikni tahminlash, tashqi

muhitni asrash, mahsulotni himoyalash, o‘zaro tushunishlikka erishish, savdodagi iqtisodiy ko‘rsatkichlarni yaxshilashdir.

Aloqa, axborotlashtirish va telekommunikasiya texnologiyalari sohasida o‘lchash vositalarini kalibrlashning tashkiliy va texnik asoslarini joriy qilish o‘lchash vositalari holatini nazorat qilishni osonlashtirishga imkon beradi. Metrologik ta’minotning tarkibiga O‘zbekiston aloqa va axborotlashtirish agentligi, UNICON.UZ markazi qoshidagi Asos metrologiya xizmati, Telekommunikasiyalar va pochta aloqasi sohasidagi standartlashtirish bo‘yicha Texnika qo‘mitasi, Davlat aloqa inspeksiyasi, metrologik xizmatlar va xo‘jalik yurituvchi sub’ektlari o‘lchash vositalarining holati uchun mas’ul shaxslar kiradi.

4-bob bo‘yicha nazorat savollari

1. Standartlashtirish iborasining ma’nosini tushuntirib bering.
2. Aloqa va axborotlashtirish sohasidagi standartlashtirish tizimining asosiy printsiplarini aytib bering.
3. Standartlashtirishning asosiy maqsadi nimalardan iborat?
4. Metrologiya iborasining ma’nosini tushuntirib bering.
5. Sertifikatlashtirish iborasining ma’nosini tushuntirib bering.
6. Standartlar qanday xususiyatlarga ega.
7. Mintaqaviy standartlashtirish deganda nimani tushunasiz.
8. Milliy standartlashtirishning asosiy maqsadi nimalardan iborat?
9. Standartlashtirishning asosiy tamoyillarini sanab bering.
10. Metrologik xizmatlarni ko‘rsatish bo‘yicha respublika markazining vazifalarini aytib bering.

5-BOB. KONSTRUKSIYALASHNING ASOSIY ELEMENTLARI

5.1. Murakkablikni kamaytirish

Dasturiy ta’minotni konstruktsiyalash (“Software Construction”) bilim sohasi quyidagi bo‘limlarni o‘z ichiga oladi:

- murakkablikni kamaytirish (*Reduction* in Complexity),
- uslubdan og’ishning oldini olish (Anticipation of *Diversity*),
- sinovlarni tuzilmalashtirish (Structuring for Validation),
- tashqi standartlardan foydalanish (Use of External Standards).

Murakkablikni kamaytirish - bu konstruktsiyalashdagi murakkablikni minimallashtirish, kamaytirish va alohida qismlarga bo‘lish. Murakkablikni minimallashtirish, ijrochilarning murakkab tuzilmalarni va uzoq vaqt davomida katta hajmdagi ma’lumotlarni qayta ishlash qobiliyatining cheklanganligi bilan belgilanadi. Murakkablikni minimallashtirishga, xususan, konstruktsiyalash jarayonida modullar va boshqa sodda elementlardan foydalanish, shuningdek, standartlarning tavsiyalari yordamida erishiladi.

Dasturiy ta’minot konstruktsiyalashning murakkabligini kamaytirish uchun oddiy va oson o‘qiladigan kodni yaratish orqali erishiladi. Bunda kodning samaradorligini oshirish, kodning sinov qulayligini ta’minlash, kodning ishlashi va belgilangan mezonlarga javob berishiga e’tibor beriladi. Bu loyihaning funktsionalligi, xususiyatlari va cheklovlariga ta’sir qiladi. Murakkablikni kamaytirish zarurati konstruktsiyalashning barcha jihatlariga ta’sir qiladi va ayniqsa dasturiy komponentlarning konstruktsiyalash natijalarini tekshirish va sinovdan o‘tkazish uchun juda muhimdir.

Murakkablikni lokalizatsiya qilish - bu ob’ektga yo‘naltirilgan yondashuvdan foydalangan holda konstruktsiyalashning usuli bo‘lib, bu ob’ektlarning interfeysini cheklaydi, ularning o‘zaro ta’sirini soddalashtiradi, shuningdek ob’ektlarning to‘g’riligini va ular o‘rtasidagi munosabatlarni tekshirishni soddalashtiradi. Lokallashtirish kodda aniqlangan xatolarga o‘zgartirishlarni osonlashtiradi.

Dasturiy ta’minotning murakkabligi - bu dasturlashning ajralmas xususiyati bo‘lib, u dasturlarni yaratish vaqti va xarajatlarida, dastur matnining hajmida yoki uzunligida, boshqaruvni uzatuvchi operatorlari (tarmoqlanish operatorlari, davriy operatorlar, funktsiyalarni chaqirish) tomonidan o‘rnatilgan mantiqiy tuzilishining xususiyatlarida namoyon bo‘ladi. Dasturlash murakkabligining 5 ta manbasi mavjud:

– hal qilinishi kerak bo‘lgan masala;
– dasturlash tili;
– dasturni bajarish muhiti;
– hamkorlikdagi texnologik jarayoni;
– algoritmlar va ma’lumotlar turlarining ko‘p qirraliligi va samaradorligiga intilish.

Murakkablikni yo‘q qilib bo‘lmaydi, lekin uning namoyon bo‘lish xususiyatlarini o‘zgartirish mumkin. Murakkablik to‘rtta asosiy sababga bog’liq:

– ishlab chiqishga buyurtma kelib chiqadigan sohaning murakkabligi;
– ishlab chiqish jarayonini boshqarishning qiyinligi;
– dasturda yetarlicha moslashuvchanlikni ta’minlash zarurati;
– katta diskret tizimlarning xatti -harakatlarini tavsiflashning qoniqarsiz usullari.

Ishlab chiqishga buyurtma kelib chiqadigan sohaning murakkabligi. Biz dasturiy ta’minot bilan hal qilmoqchi bo‘lgan muammolar ko‘pincha muqarrar ravishda murakkab elementlarni o‘z ichiga oladi va ularga mos keladigan dasturlarga har xil, ba’zida o‘zaro istisno talablar qo‘yiladi. Foydalanuvchilar va ishlab chiquvchilar muammoning mohiyati haqida turlicha qarashlarga ega va ular mumkin bo‘lgan yechimlar haqida turlicha xulosalar chiqarishadi.

Tizimning birinchi versiyalari bilan tanishish foydalanuvchilarga haqiqatdan ham nima kerakligini yaxshiroq tushunish va ifodalash imkonini beradi. Shu bilan birga, ishlab chiqish jarayoni ishlab chiquvchilarning qaralayotgan sohadagi ko‘nikmalarini yaxshilaydi va ularga ishlab chiqilgan tizimdagi qorong’u joylarni aniqlaydigan yanada mazmunli savollar berish imkonini beradi.

Dasturiy ta’minotni ishlab chiqish jarayonini boshqarishdagi qiyinchiliklar. Ishlab chiquvchilarning asosiy vazifasi - oddiylik fantaziyasini yaratish, foydalanuvchilarni tasvirlangan mavzu yoki jarayonning murakkabligidan himoya qilish. Bugungi kunda dasturiy ta’minot tizimlari keng tarqalgan bo‘lib, ularning o‘lchami yuqori darajadagi tillarda o‘n minglab va hatto millionlab satrlarda baholanadi. Hech kim bunday tizimni to‘liq tushuna olmaydi. Shuning uchun, bunday ish hajmi ishlab chiqish guruhini jalb qilishni talab qiladi. Ishlab chiquvchilar qancha ko‘p bo‘lsa, ular orasidagi aloqa shunchalik murakkab bo‘ladi va muvofiqlashtirish qiyinlashadi, ayniqsa, ish ishtirokchilari bir -biridan geografik jihatdan uzoqda bo‘lsa.

Dasturiy ta’minotning moslashuvchanligi. Dasturlash moslashuvchan xususiyatga ega va ishlab chiqaruvchi har qanday darajadagi mavhumlik bilan bog’liq bo‘lgan barcha kerakli elementlarni o‘zi ta’minlay oladi. Bu moslashuvchanlik e’tiborni o‘ziga tortadi. Bu ishlab chiqaruvchini kelajakdagi konstruktsiyaning barcha asosiy konstruktsiyalarini mustaqil ravishda yaratishga majbur qiladi. Bu bloklardan yuqori darajadagi abstraktsiya elementlari tuziladi. Ko‘p qurilish elementlari va materiallar sifati uchun yagona standartlar mavjud bo‘lgan qurilish sanoatidan farqli o‘laroq, dasturiy ta’minot sanoatida bunday standartlar deyarli yo‘q. Bundan tashqari, ko‘pincha dasturni ma’lum bir foydalanuvchining individual talablari va tizim muhiti uchun sozlash kerak bo‘ladi. Shuning uchun, dasturiy ta’minotni ishlab chiqish juda ko‘p mehnat talab qilinadigan jarayon bo‘lib qolmoqda.

Katta diskret tizimlarning xatti -harakatlarini ifodalash muammosi. Analog tizimlar, masalan, yuqoridan tashlangan to‘pning harakati kabi, uzluksizdir. Kirish parametrlarining kichik o‘zgarishi har doim chiqishda kichik o‘zgarishlarga olib keladi. Boshqa tomondan, diskret tizimlar tabiatan cheklangan sonli mumkin bo‘lgan holatlarga ega. Biz tizimlarni qismlarga ajratish orqali loyihalashtirishga harakat qilamiz, shunda bir qismi boshqasiga minimal ta’sir ko‘rsatadi. Dastur tizimidan tashqaridagi har bir hodisa uni yangi holatga o‘tkazishi mumkin, bundan tashqari, bir holatdan ikkinchisiga o‘tish har doim ham aniq natija bermaydi. Bunday dasturlarni har tomonlama sinab ko‘rish mumkin emas. Noqulay sharoitlarda kichik tashqi hodisa tizimning noto‘g’ri faoliyatiga olib kelishi mumkin.

Murakkab tizimning beshta belgisi

1. Murakkab tizimlar ko‘pincha ierarxik tuzilmaga ega bo‘lgan bir - biriga bog’liq quyi tizimlardan iborat bo‘lib, ular ham o‘z navbatida quyi tizimlarga va boshqalarga bo‘linishi mumkin. Ko‘p murakkab tizimlar deyarli bir nech qismlarga bo‘linadigan ierarxik tuzilishga ega, bu bizga bunday tizimlar va ularning qismlarini tushunish, tasvirlash va hatto "ko‘rish" imkonini beradigan asosiy omil. Ehtimol, biz faqat ierarxik tuzilishga ega bo‘lgan tizimlarni tushuna olamiz. Murakkab tizimlarning arxitekturasi komponentlardan va bu komponentlarning ierarxik munosabatlaridan iborat.

2. Berilgan tizimdagi qaysi komponentalarni elementar deb hisoblash, nisbatan ixtiyoriy va asosan tadqiqotchining ixtiyorida. Bir

kuzatuvchi uchun eng past darajadagi komponenta boshqasi uchun yetarlicha yuqori darajada bo‘lishi mumkin.

3. Komponent ichidagi boglanish odatda komponentlar orasidagi boglanishdan kuchliroqdir. Bu holat komponentlar ichidagi "yuqori chastotali" o‘zaro ta’sirlarni "past chastotali" dinamikadan ajratishga imkon beradi. Komponent ichidagi va komponentlararo o‘zaro ta’sirning bu farqi tizim qismlari orasidagi funktsiyalarni ajratilishini aniqlaydi va har bir qismni alohida ajratib o‘rganishga imkon beradi.

4. Ierarxik tizimlar, odatda, har xil kombinatsiyalangan va uyushgan bir necha turdagi quyi tizimlardan iborat. Boshqacha aytganda, turli xil murakkab tizimlar bir xil tarkibiy qismlarni o‘z ichiga oladi. Bu qismlar ham o‘simlik, ham hayvonlarda uchraydigan hujayralar yoki katta tuzilmalar, masalan, qon tomir tizimlari kabi umumiy kichikroq komponentlardan foydalanishi mumkin.

5. Ishlaydigan har qanday murakkab tizim ishlagan oddiy tizimning rivojlantirilishi natijasidir. Noldan ishlab chiqilgan murakkab tizim hech qachon ishlamaydi. Demak, murakkab tizimni tuzishni ishlaydigan oddiy tizimdan boshlash kerak. Tizimni ishlab chiqish jarayonida dastlab murakkab deb hisoblangan ob’ektlar elementar bo‘lib qoladi va ulardan murakkab tizimlar quriladi. Bundan tashqari, elementar ob’ektlarni bidaniga to‘g’ri yaratish mumkin emas: tizimning haqiqiy xatti - harakatlari haqida ko‘proq bilib olish uchun avval ularni o‘rganish kerak va keyin ularni takomillashtirish kerak.

5.2. Murakkab tizimni ishlab chiqish

Dekompozitsiya (bo‘linish)ning roli. Deykstra ta’kidlaganidek, "murakkab tizimlarni boshqarish usuli qadim zamonlardan ma’lum bo‘lgan - divide et impera (разделяй и властвуй bo‘lib tashla va hukmronlik qil)". Murakkab dasturiy ta’minot tizimini loyihalashda uni iloji boricha kichikroq tizimlarga bo‘lish kerak, ularning har biri mustaqil ravishda takomillashtirilishi mumkin. Bunday holda, biz inson miyasining imkoniyatlaridan oshmaymiz: tizimning har qanday darajasini tushunish uchun biz uning bir necha qismlari haqidagi ma’lumotlarni bir vaqtning o‘zida yodda tutishimiz kerak (hammasi ham emas). Tizimning murakkabligi tizimni bo‘lishga majbur qiladi.

Bo‘limlarni ajratish uchun algoritmik yoki ob’ektga asoslangan dekompozitsiyadan foydalanish mumkin. Algoritm bo‘yicha bo‘linishi hodisalar tartibiga qaratilgan, ob’ektlar bo‘yicha bo‘linishi esa ob’ekt

yoki harakat sub’ekti bo‘lgan agentlarga asosiy e’tiborni qaratadi. Biroq, biz murakkab tizimni bir vaqtning o‘zida ikkita usulda qura olmaymiz, bu usullar asosan ortogonaldir. Biz tizimni algoritmlar yoki ob’ektlar bo‘yicha bo‘lishni boshlashimiz kerak, so‘ngra hosil bo‘lgan tuzilmadan foydalanib, muammoga boshqa nuqtai nazardan qarashga harakat qilishimiz kerak. Birinchi holda, biz o‘zgaruvchilarni, massivlarni, oddiy ma’lumotlar tuzilmalarini qayta ishlaydigan murakkab dasturlarni olamiz, ikkinchi holatda - nafaqat ma’lumotlarni, balki ularni qayta ishlash uchun kichik dasturlarni ham o‘z ichiga olgan murakkab tuzilmalarga ega bo‘lamiz. Tajriba shuni ko‘rsatadiki, ob’ektni bo‘laklarga bo‘lishdan boshlash foydali bo‘ladi. Shu tarzda boshlash bizga dasturiy tizimlarning murakkabligini tartibga solishga yordam beradi.

Abstraktsiyaning roli

Millerning tajribalarida, odatda, odam bir vaqtning o‘zida atigi 7 ± 2 birlik ma’lumotni qabul qilishi aniqlandi. Bu raqam ma’lumotlarning mazmuniga bog’liq emas. Millerning o‘zi ta’kidlaganidek: "Xotiramiz hajmi biz qabul qila oladigan, qayta ishlash va eslab qolishimiz mumkin bo‘lgan axborot miqdoriga jiddiy cheklovlar qo‘yadi. Bir vaqtning o‘zida bir nechta kanallar orqali va alohida bo‘limlar ketma -ketligi ko‘rinishida kirish ma’lumotlarining oqimini tashkil qilib, biz bu axborot oqimini to‘xtashimiz mumkin. Zamonaviy terminologiyada buni bo‘linish yoki abstraktsiyani ajratib olish deb ataladi. Odamlar murakkablikni yengish uchun juda samarali texnologiyani ishlab chiqdilar. Biz undan xulosa chiqaramiz.

Murakkab ob’ektni to‘liq yarata olmasak, biz unchalik muhim bo‘lmagan tafsilotlarni e’tiborsiz qoldiramiz va shuning uchun biz ob’ektning umumlashtirilgan, ideallashtirilgan modeli bilan shug’ullanamiz. Va biz hali ham bir vaqtning o‘zida katta miqdordagi ma’lumotni qamrab olishimiz kerak bo‘lsa-da, abstraktsiya tufayli biz ancha katta semantik hajmdagi axborot birliklaridan foydalanamiz. Bu, ayniqsa, biz dunyoni ob’ektga yo‘naltirilgan nuqtai nazardan qaraganimizda to‘g’ri keladi, chunki ob’ektlar haqiqiy dunyoning abstraktsiyasi sifatida alohida to‘yingan izchil axborot birliklari hisoblanadi.

Ierarxiyaning roli

Axborot birliklarini kengaytirishning yana bir usuli - tizimidagi sinflar va ob’ektlar ierarxiyasini tashkil qilishdir. Ob’ektli tuzilma muhimdir, chunki u o‘zaro ta’sir mexanizmlari orqali ob’ektlarning bir -

biri bilan qanday o‘zaro ta’sirini tasvirlaydi. Sinf tuzilishi bir xil darajada muhim: u tizim ichidagi tuzilmalar va xatti -harakatlarning umumiyligini belgilaydi. Nima uchun, masalan, bitta o‘simlik bargining har bir hujayrasining fotosintezini o‘rganish kerak, qachonki bitta hujayrani o‘rganish kifoya qilsa. Chunki boshqalari ham o‘zini xuddi shunday tutadi.

Garchi biz bir turdagi ob’ektlarni alohida deb hisoblasak ham, ularning xatti -harakatlari bir xil turdagi boshqa ob’ektga o‘xshash bo‘ladi deb taxmin qilishimiz mumkin. Ob’ektlarni bir -biriga bog’liq bo‘lgan abstraktsiyalar guruhiga (masalan, hayvon hujayralariga nisbatan o‘simlik hujayralari turlarini) ajratib, biz har xil ob’ektlarning umumiy va o‘ziga xos xususiyatlarini aniq ajratib turamiz, bu esa ularning o‘ziga xos murakkabligini yengishga yordam beradi.

Murakkablik nima

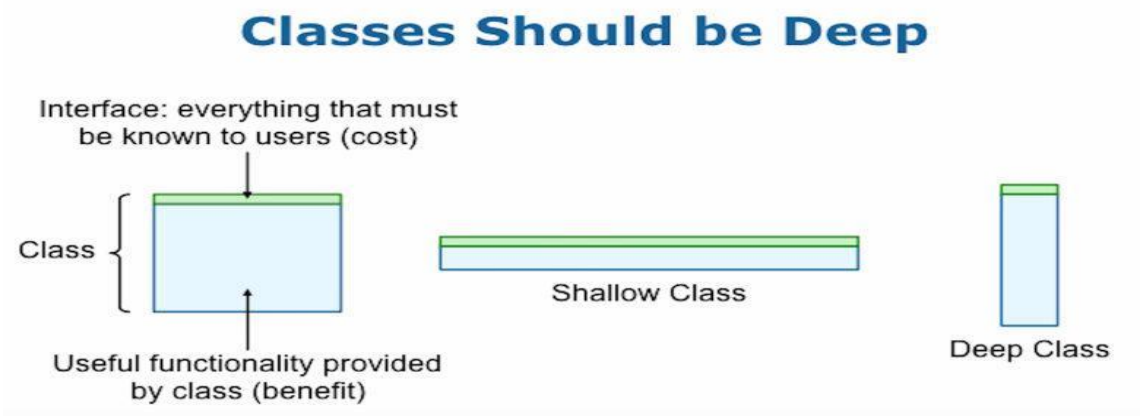
Professor Austerxout, dasturiy ta’minotni ishlab chiqishning eng katta maqsadi-bu murakkablikni kamaytirish, deb ta’kidlagan. Dasturiy ta’minotni tushunish va o‘zgartirishni qiyinlashtiradigan har qanday omil murakkablik deb ataladi. Murakkablikning ikkita asosiy manbai bor:

- noaniqlik va
- o‘zaro bog’liq kod.

Noaniqlik kodda muhim ma’lumotlarni ko‘rish mumkin emasligini anglatadi. Bog’liqlik ma’lum modulning kodini boshqa modullar bilan birlashtirilmasdan tushunish mumkin emasligini bildiradi. Murakkablikning xavfi shundaki, u ko‘payib boradi. Siz noto‘g’ri qaror qabul qildingiz, natijada kod oldingi noto‘g’ri ishlab chiqilishiga asoslangan holda, butun dasturiy ta’minot tobora murakkablashib bormoqda. "Avval mahsulotni ishlab chiqaramiz, keyin yaxshilaymiz", bu imkonsiz.

Ikkinchidan, murakkablikni tashqi muhitdan ajratish. Murakkablikni kamaytirishning asosiy usuli - murakkablikni tashqi muhitdan ajratish. “Agar siz bitta moduldagi murakkablikni tashqi muhitdan ajratib olsangiz, bu modul boshqa modullar bilan o‘zaro aloqada bo‘lmasa, murakkablikni bartaraf etish maqsadiga erishishingiz mumkin”. Dasturiy ta’minot dizayni o‘zgarganda, kod qanchalik kam o‘zgarsa, dasturiy ta’minotning murakkabligi shunchalik past bo‘ladi. Murakkablik yuzaga chiqmasligi uchun imkon qadar modulda tashqi muhitdan ajratiladi. Agar bir nechta modul ulangan bo‘lsa, bu modullarni bittaga birlashtirish maqsadga muvofiqdir.

Uchinchidan, interfeys va uni amalga oshiruvchi ilova. Modullar interfeys va uni amalga oshiruvchi ilovaga bo‘linadi. Interfeys sodda bo‘lishi kerak va uni amalga oshiruvchi ilova murakkab bo‘lishi mumkin.



5.1-rasm. Klasslar va interfeyslar[8].

Yaxshi sinf “kichik interfeys, katta funktsiya” bo‘lishi kerak, yomon sinf esa “katta interfeys, kichik funksiya” bo‘lishi kerak. Yaxshi dizayn shundaki, ko‘p sonli funktsiyalar oddiy interfeys ostida yashiringan va foydalanuvchilarga ko‘rinmas va foydalanuvchilar buni murakkab sinf deb hisoblamaydilar. Eng yaxshi misol - Unix fayllarini o‘qish va yozish interfeysi bo‘lib, u o‘qish va yozishning barcha variantlarini o‘z ichiga olgan beshta usulni ta’minlaydi.

To‘rtinchidan, xatolar sonini kamaytirish. Ba’zi dasturiy ta’minot ishlab chiqaruvchilari muammoga duch kelganda xatolarni generatsiya qilishni va istisnolarni generatsiya qilishni yoqtirishadi. Bu ham murakkablikka olib keladi, foydalanuvchilar barcha istisnolarga duch kelishlari kerak. Biror narsa noto‘g’ri ketsa, buni qanday hal qilish dasturchining vazifasidir. To‘g’ri yondashuv, foydalanuvchiga xabar berilishi kerak bo‘lgan xatolardan tashqari, boshqa xatolar dasturiy ta’minotda iloji boricha ko‘rib chiqilishi va tashlanmasligi kerak. Masalan, TCL tilining asl dizayni shundaki, unset() usuli mavjud o‘zgaruvchilarni olib tashlash uchun ishlatiladi. Agar o‘zgaruvchi mavjud bo‘lmasa, usul xato beradi. Professor Ousterhoutning aytishicha, bu dizayn xato va uni tashlab yubormaslik kerak, agar unset () aniqlansa, o‘zgaruvchi mavjud bo‘lmasa, muammo hal qilinadi.

Yana bir misol, Windows ochiq fayllarni o‘chira olmaydi va xato haqida eslatma beradi. Bu ham loyihalashdagi xato. Ba’zi foydalanuvchilar bu fayllarni o‘chira olmaydi va tizimni qayta ishga

[8] Брауде Дж. Технология разработки программного обеспечения. СПб.: Питер, 2004.

tushirishi kerak bo‘ladi. Unix yondashuvi har doim foydalanuvchilarga fayllarni o‘chirishga ruxsat beradi, lekin xotirani tozalashga ruxsat bermaydi. Ochiq fayllar xotirada qolishda davom etadi, shuning uchun ular boshqa dasturlarga xalaqit bermaydi. Bu dasturlar fayllarni saqlashni tugatgandan so‘ng, ular yo‘qligini aniqlaydilar. Xato haqida xabar beradi. Ushbu dizayn yaxshiroq.

5.3. Murakkablikni baholash usullari

Dasturiy ta’minot tizim loyihasini amalga oshirishda bu tizimining murakkabligini baholash katta ahamiyatga ega. Tizimning murakkabligi ishlab chiqaruvchilar mehnatining mahsuldorligini, shuningdek, ish hajmini, ishlab chiqish vaqtini va loyihaning narxini belgilaydi. Shuningdek, dasturiy ta’minot tizimining murakkabligi va ishonchliligi o‘rtasida yaqin bog’liqlik mavjud. Shubhasiz, dasturiy ta’minot tizimining murakkabligi qanchalik yuqori bo‘lsa, uning ishonchliligini ta’minlash shunchalik qiyin bo‘ladi. Dasturlarning murakkabligini baholashda, qoida tariqasida, uchta asosiy ko‘rsatkichlar guruhi ajratiladi: dasturlar hajmining o‘lchovlari, dasturlarni boshqarish oqimining murakkabligi ko‘rsatkichlari va dasturlarning ma’lumotlar oqimi murakkabligining ko‘rsatkichlari. Birinchi guruh baholari eng sodda va shuning uchun keng qo‘llaniladi.

Dastur o‘lchamining an’anaviy xarakteristikasi - bu boshlang’ich yozilgan dastur kodi satrlari soni. Satr sifatida har qanday dastur operatori tushuniladi. Halsted metrikasini dasturlar hajmini baholash guruhiga kiritish mumkin. Dasturda ishlatiladigan operatorlar va operandlar sonini hisoblash asos sifatida olinadi, ya’ni. dastur hajmini aniqlanadi. Dastur hajmini to‘g’ridan -to‘g’ri o‘lchash, soddaligiga qaramay, yaxshi natijalar beradi. Dastur hajmini taxmin qilish uning murakkabligi to‘g’risida qaror qabul qilish uchun yetarli emas, lekin hajmi jihatidan bir - biridan farq qiladigan dasturlarni tasniflash juda o‘rinli. Dastur hajmidagi farqlar kamayishi bilan murakkablikka ta’sir qiluvchi boshqa omillarning baholari ajratib ko‘rsatiladi. Shunday qilib, dastur hajmini baholash - bu nominal shkala bo‘yicha baholash, uning asosida har bir toifaga baho ko‘rsatilmasdan faqat dasturlar toifalari aniqlanadi.

Dasturlashning asosiy qoidalaridan biri uni soddalashtirishdir. Oddiy dastur - bu kompilyatsiya, yuritish va o‘zgartirish uchun tushunish osonroq dastur. Biroq, dasturni soddalashtiradigan umumiy qabul qilingan fikr yo‘q. Oddiy dastur – bu belgi tizimidan foydalanadigan dastur. Kam sonli ichma-ich joylashtirish darajalari bo‘lgan dastur,

bunday darajadagi ko‘p dasturlarga qaraganda sodda. Dasturlarning soddaligiga erishish uchun bir qator boshqa qoidalar mavjud, ehtimol, sodda - bu puxta ishlab chiqilgan dasturiy ta’minot tizimi. Biroq, har qanday holatda, dasturiy ta’minot tizimining murakkablik darajasini aniqlay olish maqsadga muvofiqdir.

Oddiy holatda, tizimning murakkabligi uning modullarining murakkabligi o‘lchovlari yig’indisi sifatida aniqlanadi. Modulning murakkabligini turli usullar bilan hisoblash mumkin. Masalan, M. Xolsted modulning N uzunlik o‘lchovini taklif qildi[9].

$$N \approx n1 * \log2\,(n1) + n2 * \log2\,(n2),$$

bu yerda n1 - har xil operatorlar soni, n2 - har xil operandlar soni.

Ikkinchi o‘lcham sifatida M. Xolsted modulning **V** hajmini (dastur matnining barcha operatorlari va operandlarini yozish uchun belgilar soni) ko‘rib chiqdi:

$$V = N * \log 2\,(n1 + n2).$$

Shu bilan birga, ma’lumki, har qanday murakkab tizim elementlar va elementlar orasidagi bog’lanish tizimidan iborat va tizim ichidagi aloqalarni e’tiborsiz qoldirish asossizdir. Dasturlarning murakkabligini baholashning ikkinchi guruhi - bu dasturlarni boshqarish oqimining murakkabligi ko‘rsatkichlari. Qoida tariqasida, bu hisob -kitoblar dasturlar ichidagi boshqaruv o‘tishlarining zichligi yoki bu o‘tishlarning o‘zaro bog’liqligi uchun ishlatiladi. Dasturlarning murakkabligini eng oddiy, lekin juda samarali baholaganlardan biri bu T. Gilb metrikasi, bunda dasturning mantiqiy murakkabligi IF_THEN_ELSE ifodalari bilan dasturning to‘yinganligi sifatida belgilanadi. Bu ikkita xususiyatni ko‘rsatadi:

1. CL - shartli bayonotlar soni bilan tavsiflanadigan dasturning mutlaq murakkabligi;

2. cl - dasturning nisbiy murakkabligi, dasturning shartli operatorlar bilan to‘yinganligi bilan tavsiflanadi, ya’ni. cl CL ning operatorlarning umumiy soniga nisbati sifatida aniqlanadi.

Jilb metrikasidan foydalanib, uni yana bir komponent bilan to‘ldirildi, ya’ni CLI operatorining maksimal ichma – ich joylashtirish

[9] Назаров С.В. Архитектуры и проектирование программных систем: монография /С.В. Назаров. — М.: ИНФРА-М, 2013

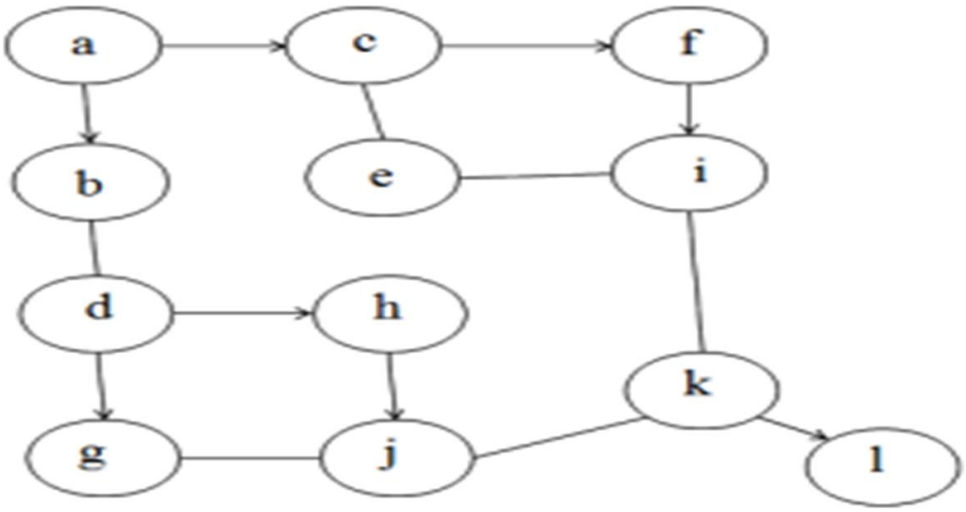
darajasining xarakteristikasi, bu esa Jilb metrikasini tsiklik tuzilmalarni tahlil qilishda qo‘llash imkonini berdi.

T. Makkeyb dasturiy ta’minotning murakkabligini baholashda ichki ulanishlar topologiyasi tomonidan qarashni taklif qildi. Shu maqsadda u davriy murakkablik (цикломатической сложности) metrikasini ishlab chiqdi:

$$V(G) = E – N + 2,$$

bu yerda E - yoylar soni, N – dasturiy ta’minot boshqaruv grafidagi cho‘qqilar (tugunlar) soni.

Bu to‘g’ri yo‘nalishdagi qadam edi. Murakkablikni baholashni yanada takomillashtirish har bir modulni elementlar va ular orasidagi bog’lanishlardan tashkil topgan lokal tuzilma sifatida ko‘rsatilishini talab qildi. Dasturlarning murakkabligini chegaraviy qiymatlar usuli bilan baholash katta qiziqish uyg’otadi. Bu usul dastur grafi bilan bog’liq bir nechta qo‘shimcha tushunchalarni kiritadi. Dasturning faqat bitta boshlang’ich va bitta oxirgi cho‘qqiga ega bo‘lgan G = (V, E) yo‘naltirilgan grafi mavjud bo‘lsin (5.2-rasm).



5.2-rasm. Dastur grafi[10].

Bu grafda cho‘qqiga kiradigan yoylar cho‘qqining manfiy darajasi, cho‘qqidan chiqayotgan yoylar esa cho‘qqining musbat darajasi deyiladi. Keyin grafning cho‘qqilar to‘plamini ikki guruhga bo‘lish mumkin:
musbat darajali cho‘qqilar <= 1;
musbat darajali cho‘qqilar > = 2.

Birinchi guruhning cho‘qqilari qabul qilish cho‘qqilari, ikkinchi guruhning cho‘qqilari esa tanlash cho‘qqilari deb ataladi. Chegaraviy

[10] Б.В. Черников. Управление качеством программного обеспечения: Учебник - М.: ИД ФОРУМ: ИНФРА-М, 2012.

qiymatlar usuli bilan baho olish uchun G grafni maksimal sondagi G’ quyi graflarga bo‘lish kerak, ular quyidagi shartlarni qanoatlantirishi kerak:

- quyi grafga kirish faqat tanlash cho‘qqisi orqali amalga oshiriladi;
- har bir quyi grafda boshqa quyi grafning har qanday cho‘qqisidan erishish mumkin bo‘lgan cho‘qqi (quyi grafning pastki chegarasi deb ataladi) mavjud.

Masalan, tugun yoylari orqali o‘zi bilan o‘zi bog’langan tanlash cho‘qqilari quyi grafni hosil qiladi. Bunday quyi grafni tashkil etuvchi cho‘qqilar soni tanlangan cho‘qqining moslashtirilgan murakkabligiga teng. Misol uchun, 5.2 - rasmdagi graf bo‘yicha quyi graflarning xarakteristikalari jadvalini tuzish mumkin (5.1-jadval).

5.1-jadval

Quyi graflarning xarakteristikalari

Dastur quyi graflarining xarakteristikalari	***Tanlov cho‘qqilari***			
	a	b	c	d
O‘tish cho‘qqilari	b,c	b,d	e,f	g,h
Graf cho‘qqilarining moslangan murakkabligi	10	2	3	3
Quyi graf cho‘qqilari	b,c,d,e,f,g,h,i,j	b	e,f	g,h
Quyi grafning pastki chegarasi	k	d	i	j

Oxirgi cho‘qqining moslangan murakkabligi 0 ga teng, qolgan qabul qilayotgan har bir cho‘qqining moslangan murakkabligi 1 ga teng. G grafning barcha cho‘qqilari moslangan murakkabliklarining yig’indisi olinadi, bu esa dasturning mutlaq chegaraviy murakkabligini hosil qiladi. Shundan so‘ng, dasturning nisbiy chegaraviy murakkabligi aniqlanadi:

$$S_0=1-(v-1)/S_a,$$

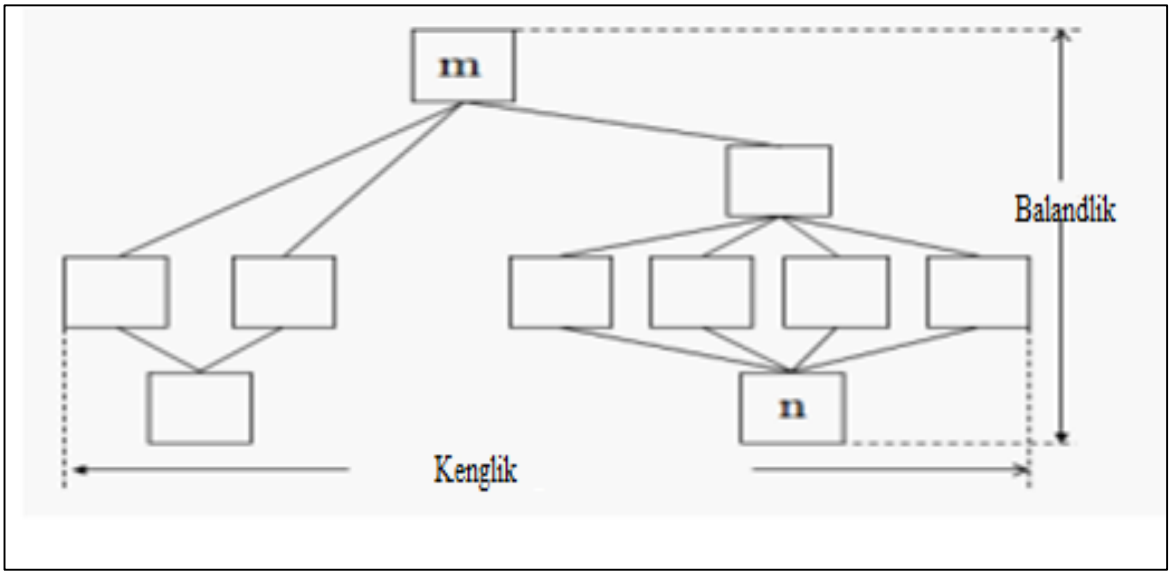
bu yerda S_0 - dasturning nisbiy chegaraviy murakkabligi;
S_a - dasturning mutlaq chegaraviy murakkabligi;
v - dastur grafigidagi umumiy cho‘qqilar soni.

Shunday qilib, dasturning nisbiy murakkabligi

$$S_0=1-(11/25)=0,56.$$

Dasturiy ta’minot tizimining murakkabligini har tomonlama baholashda modullarning murakkabligi o‘lchovini, tashqi aloqalarning murakkabligini (modullar orasidagi) va ichki aloqalarning murakkabligini (modul ichida) o‘lchashni hisobga olish kerak. An’anaga ko‘ra, tashqi

aloqalar bilan "birlashtirish“ xarakteristikasi, ichki aloqalar bilan esa "bog’lanish" xarakteristikasi taqqoslanadi.
Dasturiy ta’minot tizimining ierarxik tuzilishi dastlabki loyihalshning asosiy natijasidir. U dasturiy ta’minot modullarining tarkibini va modullar orasidagi boshqaruv munosabatlarini aniqlaydi. Bu tuzilmada yuqori darajadagi modul (boshliq) quyi darajali modulni (bo‘ysunuvchi) boshqaradi. Ierarxik tuzilma dasturiy ta’minot tizimining protsedurali xususiyatlarini aks ettirmaydi, ya’ni operatsiyalar ketma - ketligi, ularning takrorlanishi, tarmoqlanishi va hokazo. 5.3-rasmda ko‘rsatilgan ierarxik strukturaning asosiy xususiyatlarini ko‘rib chiqamiz.



5.3-rasm. Dastur tizimining ierarxik tuzilishi[11].

Birlamchi xarakteristikalar – cho‘qqilar (modullar) va qirralarning (modullar orasidagi bog’lanishlar) soni. Ularga ikkita global xarakteristikalar qo‘shiladi: balandlik - boshqarish darajalari soni; va kengligi - boshqarish darajalarida joylashgan modullar sonining maksimal qiymati. 6-rasmdagi misolda balandlik = 4, kenglik = 6. Struktura modullarining lokal xarakteristikalari kirishni birlashtiruvchi koeffitsient va chiquvchi tarmoqlanish koeffitsientidir. kirishni birlashtiruvchi koeffitsienti Fan_in(i) - i-nchi modulni bevosita boshqaruvchi modullar soni. Misolda n moduli uchun: Fan_in (n) = 4. Chiqish bo‘yicha tarmoqlanish koeffitsienti Fan_out(i) i-nchi modul tomonidan to‘g’ridan-to‘g’ri boshqariladigan modullar soni. Misolda m moduli uchun: Fan_out(m) = 3. Savol tug’iladi: strukturaning sifatini

[11] Данилкин Ф.А., Сычугов А.А. Конструирование программного обеспечения: учебное пособие. Изд-во ТулГУ, 2010.

qanday baholash mumkin? Loyihalsh amaliyotidan ma’lumki, eng yaxshi yechim daraxt shaklidagi ierarxik tuzilish bilan ta’minlanadi. Haqiqiy loyiha tuzilishi va daraxt o‘rtasidagi farq darajasi bog’lanish bo‘lmagan tuzilish bilan tavsiflanadi.

Bog’lanish yo‘qligini qanday aniqlash mumkin? Ma’lumki, n cho‘qqili to‘liq graf qirralarining soni $e_c = n * (n -1) / 2$ ga teng, shunday sonli qirraga ega daraxt nisbatan ancha kam qirralar soniga $e_t = n - 1$ ega bo‘ladi. U holda bog’lanmaslik formulasini to‘liq, haqiqiy graf va daraxt qirralari sonini solishtirib tuzish mumkin. n cho‘qqili va e qirrali loyiha tuzilishini aniqlash uchun bog’lanmaslik quyidagi ifoda bilan aniqlanadi:

$$Nev = \frac{e - e_t}{e_c - e_t} = \frac{(e - n + 1) \times 2}{n \times (n-1) - 2 \times (n-1)} = \frac{2 \times (e - n + 1)}{(n-1) \times (n-2)}.$$

Bog’lanmaslik qiymati 0 dan 1 gacha bo‘lgan oraliqda yotadi. Agar Nev = 0 bo‘lsa, u holda loyiha tuzilishi daraxt, agar Nev = 1 bo‘lsa, u holda loyiha tuzilishi to‘liq grafdir. Bog’lanmaslik strukturaning taxminiy bahosini berishi aniq. Baholashning aniqligini oshirish uchun bog’lanish va birlashish xususiyatlarini qo‘llash kerak. Yaxshi tuzilish past birlashma va yuqori bog’lanishga ega bo‘lishi kerak. L.Konstantin va E.Jordan (1979) strukturani Fan_in (i) va Fan_out (i) modullari [5] koeffitsientlari yordamida baholashni taklif qilishdi. Fan_in(i) ning katta qiymati yuqori ulanishning dalilidir, chunki bu modulga bog’liqlik o‘lchovidir. Fan_out(i) ning katta qiymati chaqiruvchi modulining yuqori murakkabligini ko‘rsatadi. Sababi, quyi modullarni muvofiqlashtirish uchun murakkab boshqaruv mantig’i talab qilinadi. Fan_in(i) va Fan_out(i) koeffitsientlarining asosiy kamchiligi shundan iboratki, ular aloqa og’irligiga e’tibor bermaydilar. Bu yerda faqat boshqaruv oqimlari (modulli qo‘ng’iroqlar) muhokama qilinadi. Shu bilan birga, strukturaning qirralarini yuklaydigan axborot oqimi sezilarli darajada o‘zgarishi mumkin, shuning uchun nafaqat qirralarning sonini, balki ular orqali o‘tadigan ma’lumotlarning miqdorini ham hisobga oladigan o‘lchov zarur.

S. Genri va D. Kafura (1981) ifan_in (i) va ifan_out (j) axborot koeffitsientlarini kiritdilar. Ular i-modul ma’lumot oladigan va shunga mos ravishda j-modul tomonidan yangilanadigan elementlar va ma’lumotlar tuzilmalari sonini hisobga oladi. Axborot koeffitsientlari

sfan_in (i) va sfan_out (j) tizimli koeffitsientlari bilan yig'iladi, ular faqat modulni chaqirishni hisobga oladi. Natijada koeffitsientlarning to'liq qiymatlari shakllanadi:

Fan_in (i) = sfan_in (i) + ifan_in (i),

Fan_out (j) = sfan_out (j) + ifan_out (j).

Strukturaning umumiy murakkabligi metrikasi modullarning umumiy koeffitsientlari asosida hisoblanadi:

$$S = \sum_{i=1}^{n} \text{length(i)} * (\text{Fan\_in(i)} + \text{Fan\_out(i)})^2,$$

bu yerda: length(i) - i-modulning o'lchamining bahosi (LOC yoki FP-baholash shaklida).

5.4. Modul ulanishi va birlashishi asosida murakkablikni baholash

Modulli dasturiy ta'minot tizimi murakkabligining mumkin bo'lgan modellaridan biri uning modullarining asosiy xususiyatlariga asoslangan - har bir modulning ulanishi va har bir modul juftini birlashtirish.

5.2- jadval

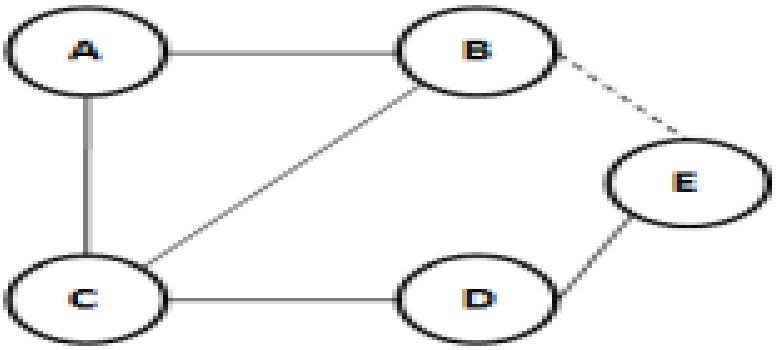
Modul ulanish darajasi

№ n/n	***Modulli ulanish***	**Ulanish darajasi**
1.	Funktsional	0 (kuchli aloqa)
2.	Axborot (ketma -ket)	0,1
3.	Kommunikativ	0,3
4.	Protsedurali	0,5
5.	Vaqt bo'yicha	0,7
6.	Mantiqiy	0,9
7	Moslik bo'yicha	1 (zaif aloqa)

Yuqorida, modullarning ulanishi va birlashtirilishi uchun raqamli baho (0 dan 10 gacha) berilgan. Aytgancha, har xil mualliflar ushbu diapazonda modulning har bir turi uchun biroz boshqacha qiymatlar berishini unutmang. Dasturiy ta'minot tizimining murakkabligini S baholash uchun biz 0 dan 1 gacha bo'lgan raqamli diapazonni tanlaymiz. Bunday holda biz yuqori ulanish va zaif birlashma nolga yaqin raqam

bilan tavsiflanadi deb taxmin qilamiz. Shunday qilib, S qiymati biriga qanchalik yaqin bo‘lsa, PS shunchalik murakkab bo‘ladi. Raqamli bahoning qiymatiga bu yondashuv bilan, CC (ulanish kuchi) va SC (yopishish kuchi) koeffitsientlarining qiymatlarini 5.2-jadvalda ko‘rsatilgan tartibda o‘zgartirish zarur.

Agar A moduli o‘zgarsa, B va C modullarini o‘zgartirish kerak bo‘ladi (birinchi darajali bog’liqliklar). Biroq, modulni o‘zgartirgandan so‘ng, C moduli o‘zgartirilgan bo‘lsa ham, B modulini o‘zgartirish kerak bo‘lishi mumkin (ikkinchi darajali qaramlik: qirralar ac - cb). 4-rasmdan ko‘rinib turibdiki, B moduli uchun uchinchi darajali bog’liqlik ham bo‘lishi mumkin: ac - cd - de - eb qirralari (chekka eb nuqta chiziq bilan ko‘rsatilgan)



5.4-rasm. Dastur modullari grafi[12]

Har qanday modul juftligi orasidagi bog’liqliklarni to‘liq tasvirlash uchun to‘liq bog’liqlik matritsasi hisoblanadi. Bu quyidagi bosqichlar ketma -ketligi yordamida amalga oshiriladi. 1. Grafikdagi har bir modul jufti orasidagi barcha yo‘llarni (tsikllardan tashqari) toping. 2. Har bir topilgan yo‘lning ehtimolliklarini mos keladigan yoylar uchun ehtimolliklar hosilasi sifatida hisoblang. 3. Modullar orasidagi bog’liqliklarni yo‘llar ehtimolini hisobga olgan holda hisoblang, lekin yo‘llarni bir -birini inkor etuvchi deb hisoblamang. Natijada, modullardan birini, agar boshqasi o‘zgarsa, o‘zgartirish kerak bo‘ladi (ehtimol, bu nisbat nosimmetrik deb taxmin qilinadi).

Misol. 5.4-rasmda ko‘rsatilgan modullar grafi uchun modullarning ulanishi va birlashishini tavsiflovchi quyidagi dastlabki ma’lumotlar berilgan bo‘lsin:

[12] Данилкин Ф.А., Сычугов А.А. Конструирование программного обеспечения: учебное пособие. Изд-во ТулГУ, 2010.

$s_A = 0,5$; $s_B = 0,7$; $s_C = 0,3$;

$s_D = 0,9$; $s_E = 1,0$; $c_{AB} = 0,5$;

c

$\overset{A}{\underset{C}{c}}_{DE} = 0.3$.

Graf yoylarining ehtimolligiga mos keladigan birinchi darajali matritsaning elementlarini hisoblaymiz:

$d_{AB} = 0,15 * (0,5 + 0,7) + 0,7 * 0,5 = 0,53;$
$d_{AC} = 0,15 * (0,5 + 0,3) + 0,7 * 0,7 = 0,61;$
$d_{CB} = 0,15 * (0,3 + 0,7) + 0,7 * 0,3 = 0,36;$
$d_{CD} = 0,15 * (0,3 + 0,9) + 0,7 * 0,1 = 0,25;$
$d_{DE} = 0,15 * (0,9 + 1,0) + 0,7 * 0,3 = 0,495$

Olingan ma’lumotlarga asoslanib, siz birinchi darajali matritsani qurishingiz mumkin (5.5-rasm). To‘liq bog’liqlik matritsasini yaratishga o‘tamiz. Buni amalga oshirish uchun (bizning vizual oddiy misolimizda) biz har bir modul jufti uchun barcha yo‘llarni belgilaymiz:

1. A, B juftlik: A – B yo‘l va A – C – B yo‘l.
2. A, C juftlik: A – C yo‘l va A – B – C yo‘l.
3. A, D juftlik: A – C – D yo‘l va A – B – C – D yo‘l.
4. A, E juftlik: A – C – D – E yo‘l va A – B – C – D – E yo‘l.
5. B, C juftlik: B – C yo‘l va B – A – C yo‘l.
6. B, D juftlik: B – C – D yo‘l va B – A – C – D yo‘l.
7. B, E juftlik: B – C – D – E yo‘l va B – A – C – D – E yo‘l.
8. D, C juftlik: C – D yo‘l.
9. D, E juftlik: D – E yo‘l.
10. E, C juftlik: E – D – C yo‘l.

	A	B	C	D	E
A	1,00	0,53	0,61	0	0
B	0,53	1,00	0,36	0	0
C	0,61	0,36	1,00	0,25	0
D	0	0	0,25	1,00	0,495
E	0	0	0	0,495	1,00

5.5-rasm. Birinchi darajali modullarning qaramlik matritsasi[13].

Biz topilgan yo‘llar uchun ehtimollik qiymatlarini grafik yoylarining ehtimollik qiymatlari yordamida hisoblaymiz:

1. $P_{AB} = d_{AB} = 0,53$; $P_{ACB} = d_{AC} * d_{CB} = 0,61 * 0,36 = 0,22$. 2. $P_{AC} = d_{AC} = 0,61$; $P_{ABC} = d_{AB} * d_{BC} = 0,53 * 0,36 = 0,19$. 3. $P_{ACD} = d_{CA} * d_{CD} = 0,61 * 0,25 = 0,15$; $P_{ABCD} = d_{AB} * d_{BC} * d_{CD} = =0,53 * 0,36 * 0,25 = 0,05$. 4. $P_{ACDE} = d_{AC} * d_{CD} * d_{DE} = 0,61 * 0,25 * 0,495 = 0,15$; $P_{ABCDE} = d_{AB} * *d_{BC} * d_{CD} * d_{DE} = 0,53 * 0,36 * 0,25 * 0,495 = 0,02$.

5. $P_{BC} = d_{BC} = 0,36$; $P_{BAC} = d_{BA} * d_{AC} = 0,53 * 0,61 = 0,32$. 6. $P_{BCD} = d_{BC} * d_{CD} = 0,36 * 0,25 = 0,09$; $P_{BACD} = d_{BA} * d_{AC} * d_{CD} = =0,53 * 0,61 * 0,25 = 0,08$. 7. $P_{BCDE} = d_{BC} * d_{CD} * d_{DE} = 0,36 * 0,25 * 0,495 = 0,044$; $P_{BACDE} = d_{BA} * * d_{AC} * d_{CD} * d_{DE} = 0,53 * 0,61 * 0,25 * 0, 495 = 0,04$.

8. $P_{CD} = d_{CD} = 0,25$. 9. $P_{DE} = d_{DE} = 0,495$. 10. $P_{EDC} = d_{ED} * d_{DC} = 0,495 * 0,25 = 0,12$.

Endi, topilgan yo‘l ehtimolliklaridan foydalanib, modullar juftligi orasidagi yo‘llar bir-birini istisno qilmasligini hisobga olib, biz modullar orasidagi bog’liqlikni aniqlaymiz.

[13] Милованов И.В., Лоскутов В.И. Основы разработки программного обеспечения вычислительных систем: учебное пособие. – Тамбов: Изд-во ГОУ ВПО ТГТУ, 2011.

1. A, B juftlik. Bu yerda ikki yo‘l bo‘lishi mumkin, shuning uchun:

$d_{AB} = P_{AB} + P_{ACB} - P_{AB} + P_{ACB} = 0,53 + 0,22 – 0,53 * 0,22 = 0,63.$

2. A, C juftlik. Bu yerda ham ikki yo‘l bo‘lishi mumkin, shuning uchun:

$d^1_{AC} = P_{ABC} + P_{ACB} - P_{ABC} + P_{AC} = 0,19 + 0,61 – 0,19 * 0,61 = 0,68.$ 3. A, D juftlik. Mumkin bo‘lgan o‘zgarishlar uchun ikkita yo‘l bor. Ushbu modullar orasidagi bog‘liqlik quyidagi qiymat bilan belgilanadi:

$d^1_{AD} = P_{ACD} + P_{ABCD} - P_{ACD} * P_{ABCD} = 0,15 + 0,05 – 0,15 * 0,05 = 0,19.$ 4. A, E juftlik. Xuddi shunday, ikkita yo‘l bor, shuning uchun:

$d^1_{AE} = P_{ACDE} + P_{ABCDE} - P_{ACDE} * P_{ABCDE} = 0,15 + 0,02 – 0,15 * 0,02 = 0,17.$

5. B, C juftlik. Mumkin bo‘lgan o‘zgarishlar uchun ikkita yo‘l bor, shuning uchun:

6. B, D juftlik. Xuddi shunday, ikkita yo‘l bor, shuning uchun:

$d^1_{BD} = P_{BCD} + P_{BACD} - P_{BCD} * P_{BACD} = 0,09 + 0,08 – 0,09 * 0,08 = 0,16.$ 7. B, E juftlik. Ikkita yo‘l bor, shuning uchun:

$d^1_{BE} = P_{BCDE} + P_{BACDE} - P_{BCDE} * P_{BACDE} = 0,044 + 0,04 – 0,044 * 0.04 = 0,08.$

8. C, D juftlik. Bu yerda faqat bitta yo‘l bor, shuning uchun:

$d_{1CD} = P_{CD} = d_{CD} = 0,25.$

9. D, E juftlik. Bu yerda ham faqat bitta yo‘l bor, shuning uchun:

$d^1_{DE} = P_{DE} = d_{DE} = 0,495.$ 10. E, C juftlik. Bu yerda ham faqat bitta yo‘l bor, shuning uchun:

d

E

Bu hisoblashlardan so‘ng tizim modullari o‘rtasida to‘liq bog’liqlik matritsasini tuzishingiz mumkin (5.6-rasm).

	A	B	C	D	E
A	1,00	0,63	0,68	0,19	0,17
B	0,63	1,00	0,56	0,16	0,08
C	0,68	0,56	1,00	0,25	0,12
D	0,19	0,16	0,25	1,00	0,49
E	0,17	0,08	0,12	0,49	1,00

5.6-rasm. To‘liq modulga bog’liqlik matritsasi[14]

Olingan matritsadan, masalan, A moduli o‘zgarganda, E modulining o‘zgarishi ehtimoli 0,17 ga teng ekanligini aniqlashimiz mumkin. Har qanday satr elementlarini (diagonal elementdan tashqari) xulosa qilib, siz dasturning tegishli modulini o‘zgartirganda o‘zgartirilishi kerak bo‘lgan modullar sonini hisoblashingiz mumkin. Masalan, agar C moduli o‘zgarsa, o‘zgarishlarning kutilayotgan soni 1,61 ni tashkil qiladi. Diagonalni olib tashlab, satrning har bir elementini 1.0 dan chiqarib, natijalarni ko‘paytirib, modulni o‘zgartirish boshqa modullarning o‘zgarishiga olib kelmasligi ehtimolini olamiz. Masalan, C moduli uchun bu ehtimollik

$$P = (1 – 0,68)\ (1 – 0,56)\ (1 – 0,25)\ (1 – 0,12) = 0.09.$$

Matritsaning barcha elementlarini sarhisob qilib, yig’indini modullar soniga bo‘lish orqali siz dasturiy ta’minot tizimining murakkabligi to‘g’risida taxminiy baho olishingiz mumkin (diagonal va takrorlanuvchi elementlar hisobga olinmaydi). Bizning misolimizda biz 0.49 qiyinchilik qiymatini olamiz. Qiymat qanchalik baland bo‘lsa, tizim shunchalik murakkab bo‘ladi. Xulosa qilib shuni ta’kidlash kerakki, ko‘rib chiqilayotgan modelning jozibadorligiga qaramay, dastur tuzilmasining asosiy xususiyatlariga asoslanganligini hisobga olsak ham, model hali ham etarli darajada tasdiqlanmagan ko‘plab taxminlarga asoslangan.

5-bob bo‘yicha xulosalar

Murakkablikni kamaytirish - bu konstruktsiyalashdagi murakkablikni minimallashtirish, kamaytirish va alohida qismlarga

[14] Милованов И.В., Лоскутов В.И. Основы разработки программного обеспечения вычислительных систем: учебное пособие. – Тамбов: Изд-во ГОУ ВПО ТГТУ, 2011.

bo‘lish. Murakkablikni minimallashtirish, ijrochilarning murakkab tuzilmalarni va uzoq vaqt davomida katta hajmdagi ma’lumotlarni qayta ishlash qobiliyatining cheklanganligi bilan belgilanadi. Dasturiy ta’minot konstruktsiyalashning murakkabligini kamaytirish uchun oddiy va oson o‘qiladigan kodni yaratish orqali erishiladi. Biz dasturiy ta’minot bilan hal qilmoqchi bo‘lgan muammolar ko‘pincha muqarrar ravishda murakkab elementlarni o‘z ichiga oladi va ularga mos keladigan dasturlarga har xil, ba’zida o‘zaro istisno talablar qo‘yiladi.

Murakkab tizimlar ko‘pincha ierarxik tuzilmaga ega bo‘lgan bir - biriga bog’liq quyi tizimlardan iborat bo‘lib, ular ham o‘z navbatida quyi tizimlarga va boshqalarga bo‘linishi mumkin. Ishlaydigan har qanday murakkab tizim ishlagan oddiy tizimning rivojlantirilishi natijasidir.

Deykstra ta’kidlaganidek, "murakkab tizimlarni boshqarish usuli qadim zamonlardan ma’lum bo‘lgan - divide et impera (разделяй и властвуй bo‘lib tashla va hukmronlik qil)". Murakkab dasturiy ta’minot tizimini loyihalashda uni iloji boricha kichikroq tizimlarga bo‘lish kerak, ularning har biri mustaqil ravishda takomillashtirilishi mumkin.

Millerning tajribalarida, odatda, odam bir vaqtning o‘zida atigi 7 ± 2 birlik ma’lumotni qabul qilishi aniqlandi. Bu raqam ma’lumotlarning mazmuniga bog’liq emas. Millerning o‘zi ta’kidlaganidek: "Xotiramiz hajmi biz qabul qila oladigan, qayta ishlash va eslab qolishimiz mumkin bo‘lgan axborot miqdoriga jiddiy cheklovlar qo‘yadi. Bir vaqtning o‘zida bir nechta kanallar orqali va alohida bo‘limlar ketma -ketligi ko‘rinishida kirish ma’lumotlarining oqimini tashkil qilib, biz bu axborot oqimini to‘xtashimiz mumkin. Professor Austerxout, dasturiy ta’minotni ishlab chiqishning eng katta maqsadi-bu murakkablikni kamaytirish, deb ta’kidlagan. Dasturiy ta’minotni tushunish va o‘zgartirishni qiyinlashtiradigan har qanday omil murakkablik deb ataladi.

Dasturiy ta’minot tizim loyihasini amalga oshirishda bu tizimining murakkabligini baholash katta ahamiyatga ega. Tizimning murakkabligi ishlab chiqaruvchilar mehnatining mahsuldorligini, shuningdek, ish hajmini, ishlab chiqish vaqtini va loyihaning narxini belgilaydi.

5-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotning murakkabligi deganda nimani tushunasiz.
2. Dasturiy ta’minotning murakkabligini kamaytirish deganda nimani tushunasiz.
3. Murakkablikni lokalizatsiya qilish jarayonini tushuntirib bering.
4. Dasturiy ta’minotning moslashuvchanligiga ta’rif bering.

5. Murakkab tizimning belgilarini sanab bering.
6. Murakkab tizimni ishlab chiqish jarayonini aytib bering.
7. Dasturiy ta’minotni ishlab chiqishda dekompozitsiyaning rolini tushuntirib bering.
8. Dasturiy ta’minotni ishlab chiqishda murakkablikni tashqi muhitdan ajratish deganda nimani tushunasiz.
9. Murakkablikni baholashning qanday usullarini bilasiz.
10. Modul ulanishi va birlashishi asosida murakkablikni baholashni tushuntirib bering.

6-BOB. DASTURIY TA’MINOTNI KONSTRUKSIYALASH TAMOYILLARI

6.1. O‘zgarishlarni kutish

Ko‘pgina dasturiy ta’minot vaqt o‘tishi bilan o‘zgaradi va o‘zgarishlarni kutish dasturiy ta’minotni konstruktsiyalashning ko‘p jihatlarini boshqaradi. O‘zgarishlarni kutish (Ожидание изменений - anticipating change) dasturiy ta’minot muhandisligining harakatlantiruvchi kuchlaridan biridir. Dasturiy ta’minot ham tizimli, ham faoliyat sohasi nuqtai nazaridan tashqi muhitdan ajralmas hisoblanadi. Bundan tashqari, dasturiy ta’minot tizimlari o‘zgaruvchan muhitning bir qismi bo‘lib, u bilan o‘zgarishi kerak, ba’zan esa muhitning o‘zida o‘zgarishlar manbai bo‘lishi kerak. O‘zgarishlarni oldindan ko‘rish ko‘plab maxsus usullar bilan qo‘llab-quvvatlanadi:

- Aloqa usullari (masalan, hujjat formatlari va mazmuni standartlari);
- Dasturlash tillari (masalan, Java va C++ kabi tillar uchun til standartlari);
- Platformalar (masalan, operatsion tizim murojaatlari uchun dasturchi interfeysi standartlari);
- Instrumentlar (masalan, UML (Unified Modeling Language) kabi yozuvlar uchun diagrammatik standartlar).

O‘zgarishlarga tayyor turish dasturiy ta’minot muhandislariga moslanuvchan dasturiy ta’minotni yaratishga yordam beradi, ya’ni ular dasturiy mahsulotni bazaviy tuzilmani buzmasdan yaxshilashlari mumkin. O‘zgarishlarni oldindan ko‘rish ko‘plab maxsus usullar bilan ta’minlanadi. Ikki turdagi o‘zgarishlar mavjuddir[15]:

1) Dastur kodini takomillashtirish(refactoring);
2) Yangi funksionallikni qo‘shish(reinjenering).

Refactoring — bu mavjud kodlar majmuasini qayta tuzish, tashqi tuzilishini o‘zgartirmasdan ichki tuzilishini o‘zgartirish uchun ajoyib texnologiya. Bu funksiyalarni **tranformatsiya** (funksiyaning vazifasi o‘zgarmaydi, balki bajarish usuli o‘zgaradi. Masalan, bank tranformatsiyasi deganda, bank qiladigan funksiyalar o‘zgarmaydi, lekin uslub o‘zgaradi) qilish va algoritmlarni qayta ko‘rib chiqish orqali ichki kod tuzilishini yaxshilaydi. Bu takroriy jarayon. Refactoringa

[15] Фаулер М. Рефакторинг: улучшение существующего кода. - Пер. с англ. - СПб: Символ Плюс, 2003.

qamrov(scope)ni qisqartirish, murakkab ko‘rsatmalarni oddiylashtirish va bir nechta bayonot(statement)larni bitta bayonotga birlashtirish kiradi. Kodni refactoring texnikasi orqali qayta ishlash, uni o‘zgartirish, bajarish va yuklab olish jarayonini tezlashtiradi. Hosildorligini(productivity) oshirishni xohlaydigan dasturchilar uchun bu eng yaxshi amaliyotdir.

Martin Fauler refaktoring qilish uchun to‘rtta asosiy sababini aytib o‘tdi. Refactoring dasturiy ta’minot dizaynini yaxshilaydi, dasturiy ta’minotni tushunishni osonlashtiradi, xatolarni topishga yordam beradi va dasturni tezroq bajarilishiga yordam beradi. Refactoringning qo‘shimcha foydasi bor. Refactoringsiz, dasturchi dastur haqida o‘ylash tarzini o‘zgartiradi. Refactoringning quyidagi 3 ta turi mavjud[16]:

1. Kodni qayta ishlash: Bu dasturning manba kodini qayta ishlash.
2. Ma’lumotlar bazasini qayta tuzish: bu ma’lumotlar bazasini o‘zgartirish, bu uning axborot semantikasini saqlab, dizaynini yaxshilaydi.
3. Foydalanuvchi interfeysi (UI)ni qayta tuzilish: bu foydalanuvchi interfeysi uchun oddiy o‘zgarish bo‘lib, uning semantikasini saqlab qoladi.

Refactoring bizga quyidagi qulayliklarni beradi: dasturiy ta’minotning dizaynini yaxshilaydi, dasturiy ta’minotni tushunishni osonlashtiradi, dasturiy ta’minot xatolarni topishda va tezroq dasturlashda yordam beradi. Dasturlashda juda foydali bo‘lgan bir gap bor – «If it ain’t broken, don’t fix it» («Agar nimadir buzilmagan bo‘lsa, uni sozlamang«).

Shunday holatlar bo‘ladiki, dasturning ishlab turishiga asos bo‘lgan kodni ko‘rganingizda uni yaxshiroq usulda qayta yozgingiz keladi. Aslida esa dastur buzilmagan, qanday ishlashi kerak bo‘lsa shunday ishlab turgan bo‘ladi. Bunday holatda agar «ishlab turgan kodga tegmaslik» tamoyiliga amal qilsak, kodni o‘z holatida qoldirishimiz va ishlab turgan dasturni sozlashga urinmasligimiz kerak.

Foydasi: Agar ishlab turgan kodga tegmasak, ko‘p vaqtimizni tejab qolgan bo‘lamiz. Chunki dastur shu holatda ham ishlab turibdi, uning buzilmagan kodini sozlaganimizdan keyin ham ishlab turaveradi. Albatta, avvalgiga nisbatan optimalroq ishlay boshlashi mumkin. Ammo oldindan ko‘rilmagan muammolar kelib chiqmasligini hisobga olsak, ishlab turgan kodga tegmaganimiz ma’qul.

[16] Фаулер М. Рефакторинг: улучшение существующего кода. - Пер. с англ. - СПб: Символ Плюс, 2003.

Zarari: Yuqorida aytilganidek, dastur ishlab turganligiga qaramasdan uni yanada optimallashtirish mumkin. Dastur hozirgiga nisbatan yaxshiroq, yengilroq, samaraliroq ishlasa, buning nimasi yomon? Bundan tashqari kodi yaxshilangan dastur ustida keyinchalik boshqa dasturchilar ish olib borishsa, ularning kodga tushunishlari ancha osonlashishi mumkin.

Refactoring nima? Yuqorida yozilgan holatning aksi, ya’ni ishlab turgan dasturni sozlashga urinish, uning kodidagi kamchiliklarni tuzatish va optimallashtirish refactoringga misol bo‘lishi mumkin. Refactoring bilan shug’ullanishni yaxshi ko‘ruvchi dasturchilar boshqa dasturchilar hayotini osonlashtiradi. Refactor (qayta ishlab chiqish) – kodning o‘qilishi, yengil ishlashi va har tomonlama ijobiy tomonga o‘zgarishini ta’minlaydi.

Foydasi: Refactoringdan o‘tgan dastur (odatda) avvalgiga nisbatan yaxshiroq ishlaydi. Shu jihati dasturdan foydalanuvchilarga sezilishi mumkin. Dasturchilar uchun esa refactoringdan o‘tgan kod ancha oson o‘qilishi, optimal usulda yozilganligi bilan ma’qul keladi.

Zarari: Ba’zan refactoringdan so‘ng dastur avval qanday ishlagan bo‘lsa shunday ishlayveradi. Uning ishida hech qanday optimallashish kuzatilmaydi. Vaziyat yomon tus olganda esa dasturda avval kuzatilmagan ba’zi muammolar ham paydo bo‘lib qolishi mumkin. Bundan tashqari natijasi sezilmagan ish uchun dasturchi ancha vaqt sarflaydi. Shu jihatlarini hisobga olsak, refactoring dasturchi uchun vaqtni behuda o‘tkazishdek bo‘lib ko‘rinadi.

Siz dasturchi sifatida qaysi yo‘lni tanlashingiz faqat o‘zingizga bog’liq. Menimcha, aynan yuqorida berilgan fikrlarga aloqador holatda dasturchining qaysi usulda ish yuritishi uning odatiy hayotdagi xarakteriga bog’liq. Bo‘sh vaqtingiz ko‘p bo‘lsa, boshqa kerakliroq ish Sizni kutib turmagan bo‘lsa refactoring bilan shug’ullanishingiz mumkin. Tig’iz vaqt rejimida ishlayotgan dasturchilar esa odatda ishlab turgan kodni zaruratsiz o‘zgartirishmaydi.

Refaktoring yoki kodni qayta loyihalash va konstruktsiyalsh, kodni qayta ishlash, algoritmlarning o‘zgarishi bilan ekvivalent - dasturning tashqi xatti-harakatlariga ta’sir qilmaydigan va uning ishini tushunishni osonlashtirishga qaratilgan ichki strukturasini o‘zgartirish jarayoni. Refaktoring bir qator kichik, ekvivalent (ya’ni xatti-harakatni saqlaydigan) o‘zgarishlarga asoslanadi. Har bir o‘zgarish kichik bo‘lganligi sababli, dasturchi uchun uning to‘g’riligini kuzatish osonroq bo‘ladi va shu bilan birga, butun ketma-ketlik dasturni sezilarli darajada

qayta konstrutsiyalashga olib kelishi va uning izchilligi va ravshanligini oshirishi mumkin.

Refaktoring – bu dasturning tashqi xulq-atvoriga ta’sir qilmasdan ichki strukturasini o‘zgartirish jarayonidir. Bunda asosiy maqsadlardan biri dastur ishini tushunishni osonlashtirishdir. Dastur konstruktsiyasiga oson kuzatib borish imkonini beruvchi kichik o‘zgarishlar qilinadi. Refaktoring zarurligining sabablari:

- Dastur kodining takrorlanishi;
- Uzun usul;
- Katta sinf;
- Parametrlarning uzun ro‘yhati;
- Boshqa obyekt ma’lumotlariga tez-tez murojaat qiluvchi metodlar;
- Ortiqcha vaqtinchalik o‘zgaruvchilar;
- Ma’lumotlar sinflari;
- Guruhlanmagan ma’lumotlar.

Dasturiy ta’minot reinjiniringi - bu ishlayotgan dasturiy ta’minotdan foydalanib, yangi funksionallikni yaratish yoki katta o‘zgarishlar orqali xatolarni yo‘q qilish jarayoni. Chikovski va Kross 1990 yilgi maqolalarida reinjiniring jarayonini “Tizimni yangi shaklda qayta tiklash uchun tekshirish va o‘zgartirish” deb ta’riflab berishgan. Kamroq rasmiy ravishda aytganda, reinjiniring - bu dasturiy inginiring jarayonidan so‘ng teskari ravishda dasturiy ta’minot tizimini o‘zgarishdir.

6.2. Reinjiniringning murakkabligi

Qoidaga ko‘ra, "yangi dasturiy mahsulotni ishlab chiqish osonroq" deb ta’kidlanadi. Bu quyidagi muammolarga bog’liq:

1. reinjiniring, ko‘pincha yangi dasturiy ta’minotni ishlab chiqishdan ko‘ra qimmatroqdir, chunki oldingi versiyalarning cheklovlarini olib tashlash, ular bilan muvofiqlikni saqlash kerak;
2. reinjiniringni past va o‘rta malakali dasturchi amalga oshira olmaydi - hatto professionallar ham ko‘pincha uni yuqori sifat bilan amalga oshira olmaydi, shuning uchun bu dasturlarni qayta ishlash bo‘yicha katta tajribaga va turli texnologiyalarni bilishga ega bo‘lgan dasturchilarning ishini talab qiladi;
3. Dastur ishlab chiquvchiga boshqa dasturchining kodini tushunish qiyin bo‘lishi mumkin - bu uni notanish dasturlash uslubini idrok etishga moslashishga majbur qiladi, loyihada amalga oshirilgan kontsepsiyalarni har tomonlama tahlil qilish va kutubxonalardan foydalanishni

o‘zlashtirishga vaqt ajratishi kerak bo‘ladi. Bunda barcha yomon hujjatlashtirilgan kod bo‘limlarining ishlash printsipini sinchkovlik bilan o‘rganishni talab qiladi - va bularning barchasi mahsulotning yangi arxitekturaviy yechimlarga o‘tish jarayonini murakkablashtiradi;

4. Bundan tashqari, faoliyatning o‘ziga xos xususiyati qo‘shimcha motivatsiyani talab qiladi: yangi mahsulotlarni yaratish bilan solishtirganda, mavjudlarini qayta ishlash har doim ham bir xil vizual va ta’sirchan natijalarni keltirib chiqarmaydi, ko‘pincha texnik ishlarni yuklaydi va ishlab chiqarishda professional fikrlash uchun kam imkoniyat qoldiradi.

Shu bilan birga, agar dastlab dastur qat’iy va aniq arxitekturaga ega bo‘lsa, reinjiniring juda oson bo‘ladi. Shuning uchun, loyihalashda, qoida tariqasida, nima foydaliroq bo‘lishi tahlil qilinadi - oldingi loyihaning materiallarini qayta ishlash yoki shunga o‘xshash dasturiy mahsulotni "noldan" ishlab chiqish. O‘zgarishlarni boshqarish dasturiy ta’minotni ishlab chiqishni boshqarishning boshqariladigan yondashuvini o‘z ichiga oladi, shu bilan yetkazib beriladigan mahsulot bilan bog’liq muammolar xavfini kamaytiradi.

O‘zgarishlarni boshqarishning yaxshi ko‘rsatkichlaridan biri takrorlanadigan jarayonlardir. Agar yondashuv nazorat qilinadigan bo‘lsa, uni bir xil natijalar bilan va loyiha natijalariga ko‘proq ishonch bilan takrorlash mumkin. O‘zgarishlarni boshqarish va ularning instrumentlari turli mutaxassislar uchun turli ma’nolarni anglatishi mumkin. Aslida, o‘zgarishlarni boshqarish instrumentlarining uchta darajasi mavjud:

– mahsulot versiyalarining asosiy nazoratini ta’minlovchi versiyani boshqarish instrumentlari;

– asosan ishlab chiquvchilarning unumdorligini asosiy ustuvor vazifa sifatida ko‘radigan dasturchilarga yo‘naltirilgan instrumentlar;

– muayyan jarayonni avtomatlashtiradigan, tartibga soluvchi va kuzatuvchi jarayonga asoslangan instrumentlar.

Ushbu uch turdagi instrumentlardan faqat jarayonga asoslangan o‘zgarishlarni boshqarish instrumentlari jarayonning takrorlanishi va dasturiy ta’minot chiqarilishidan oldin tegishli kodni tasdiqlash imkonini beradi.

Jarayonga asoslangan o‘zgarishlarni boshqarish yakuniy natijaga erishish uchun mutaxasislar, protseduralar, usullar, uskunalar va instrumentlar qanday birlashtirilganligiga e’tibor qaratadi. Boshqariladigan o‘zgarishlarni boshqarish jarayonining zaruriy sharti,

agar tegishli jarayon kuzatilsa, yaxshi natija kafolatlanadi. Dasturiy ta’minotning sifati butunlay uni ishlab chiqish jarayonlarining sifatiga bog’liq.

Sifatni ta’minlash jarayonlari to‘xtamaydi. Jarayonni uzluksiz optimallashtirish, jumladan, muammolar yoki yaxshilanishlarni aniqlash va shunga mos ravishda jarayonlarni sozlash muhim ahamiyatga ega. Bu tartibga solinadigan va takrorlanadigan jarayonlarni talab qiladi. Agar jarayonlar beqaror bo‘lsa, jarayonlar samaradorligini pasaytirmasdan kamchiliklarni tuzatuvchi o‘zgartirishlar mumkin emas.

Ilovalarni ishlab chiqish hayotiy tsiklida sifat kafolati uchun arxitekturani taklif qiluvchi bir qator metodologiyalar mavjud. Metodologiyalar, jumladan CMM modeli va ISO 9000, aniq belgilangan jarayonlarning muhimligini ta’kidlaydi.

Xuddi shunday, Axborot texnologiyalari infratuzilmasi kutubxonasi (ITIL) kabi xizmatlarni boshqarish standartlari xizmat ko‘rsatishning eng yaxshi amaliyotlarini taqdim etishning bir qismi sifatida jarayonga asoslangan o‘zgarishlarni boshqarishdan foydalanishni talab qiladi.

Jarayonga asoslangan o‘zgarishlarni boshqarish, aslida rivojlanishning sifati va tezligida sezilarli yaxshilanishlarga olib kelishi mumkin, natijada rivojlanish tezroq va arzonroq bo‘ladi.

O‘zgarishlarni boshqarish jarayonlari dasturiy ta’minotni ishlab chiqish sifatini yaxshilash uchun mo‘ljallangan, shuning uchun sifatni yaxshilash o‘zgarishlarni boshqarishga xos bo‘lishi kerak. Jarayonga asoslangan o‘zgarishlarni boshqarish dasturiy ta’minotni ishlab chiqish jarayonida izchil amaliyotlarni ta’minlash orqali sifatni oshirishi mumkin. Jarayonga bog’liq bo‘lmagan o‘zgarishlarni boshqarish barqaror bo‘lmagan sifat yaxshilanishiga olib kelishi mumkin va bu yondashuvlar jarayonning faqat alohida qismlariga qaratilganligi sababli, ishlab chiqishda nomuvofiqliklar va ishlab chiquvchining xatosi uchun shartlar bo‘ladi. Jarayonga asoslangan o‘zgarishlarni boshqarish dasturiy ta’minotni ishlab chiqish tezligini ikki yo‘l bilan oshirishi mumkin:

Birinchidan, o‘zgartirish jarayonlari samaraliroq amalga oshiriladi, shuning uchun o‘zgarishlarni boshqarishga sarflangan konstruktsiyalash vaqti kamayadi.

Ikkinchidan, ishlab chiqilgan dasturiy ta’minotdagi nuqsonlar darajasi sezilarli darajada kamayadi (sifatni yaxshilash natijasida), shuning uchun kamchiliklarni bartaraf etish va qayta sinovdan o‘tkazish va eng muhimi, ishlab chiqarishda ushbu nuqsonlarni keltirib chiqaradigan muammolarni bartaraf etish uchun sezilarli

konstruktsiyalash vaqti tejaladi. Dasturiy ta’minotni ishlab chiqishda sifat aslida tezlikka mos keladi.

Olib borilgan tadqiqot jarayonlarga asoslangan o‘zgarishlarni boshqarishning samarali yondashuviga ega bo‘lmagan bir qator o‘zgarishlarni boshqarish xarajatlarini aniqladi.

– Yangi rivojlanish konstruktsiyasida har kuni harakatning taxminan 15% o‘zgarishlarni boshqarish faoliyatiga sarflanadi.

– Qo‘llab-quvvatlovchi konstruktsiyalarda bu ko‘rsatkich 25% ga yaqin.

– Taqsimlangan muhitdagi yangi konstruktsiyalarda kuchning 20% o‘zgarishlarni boshqarishga sarflanishi mumkin.

– Tarqalgan ilovalarni qo‘llab-quvvatlash o‘zgarishlarni boshqarish xarajatlarini 40% gacha oshirishi mumkin.

Quyidagi jadval, agar bu jarayonlar avtomatlashtirilmasa va jarayonga asoslangan o‘zgarishlarni boshqarish vositasi yordamida samarali boshqarilmasa, konstruktsiyaning umumiy qiymatidagi o‘zgarishlarni boshqarish bo‘yicha tadbirlar qiymatining mumkin bo‘lgan ulushi ko‘rsatilgan. Jarayonga asoslangan o‘zgarishlarni boshqarish vositalari nafaqat ushbu o‘zgarishlar jarayonlarining samarali bajarilishini (takomillashtirilishini) ta’minlashi, balki ushbu operatsiyalarning narxini ham kamaytirishi mumkin.

Tadqiqotlar shuni ko‘rsatdiki, jarayonga asoslangan o‘zgarishlarni boshqarishning samarali yechimi konstruktsiyalash xarajatlarini 30 foizga yoki undan ko‘proqqa kamaytirishi mumkin. Quyidagi 6.1-jadvalda AllFusion Harvest Change Manager yordamida o‘zgarishlarni boshqarish xarajatlarining umumiy konstruktsiya xarajatlaridagi ulushini qanday kamaytirishingiz mumkinligi ko‘rsatilgan.

6.1-jadval

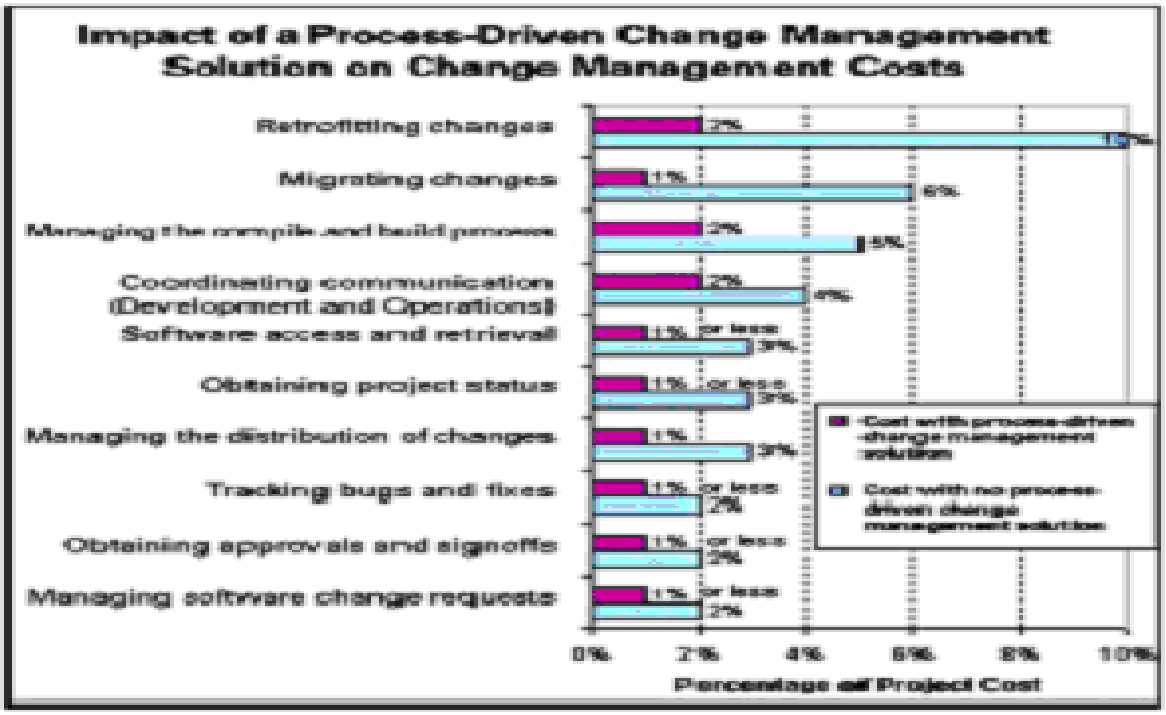
Konstruktsiyalash bo‘yicha xarajatlar.

O‘zgarishlarni boshqarish masalasi	Jarayonlar	Konstruktsiyalash bo‘yicha xarajatlar (%)
Dasturiy ta’minotga kirish	Dasturiy ta’minot komponentining turli versiyalarini, ma’lum komponentlarga kirish huquqiga ega bo‘lgan odamlarni va kirish usullarini boshqarish	3%

Rivojlanish muhitidagi o‘zgarishlarni sinxronlashtirish	Barcha faol komponentlarni sinxronlashtirish uchun bir vaqtda va hamkorlikda ishlab chiqishni boshqarish	10%
O‘zgarishlarni ko‘chirish	Hayotiy tsikldagi o‘zgarishlarning ham oldinga, ham orqaga ko‘chishini boshqarish	6%
Kompilyatsiya va yig’ish jarayonini boshqarish	Manba komponentlarini yig’ish va ulash orqali yaratilgan bajariladigan fayllar yoki yuklash modullari; manba o‘zgarganda bir nechta bajariladigan fayllarga yakuniy ta’sirni aniqlash	5%
O‘zgarishlarni tarqatishni boshqarish	Elektron manbalarni uzatish uchun tayyorlash va muayyan xodimlar tomonidan ishlatiladigan dastur versiyalarini kuzatib borish	3%
Tasdiqlash va ruxsat berish	Tasdiqlash va/yoki yozma ruxsatnomalarni olish, masalan, migratsiya, tarqatish kabi operatsiyalardan oldin	2%
Dasturiy ta’minotni o‘zgartirish so‘rovlarini boshqarish	O‘zgartirish so‘rovlarining borishini kuzatish	2%
Rivojlanish va operatsiyalarning izchilligi uchun aloqani muvofiqlashtirish	Axborot oqimini ta’minlash, ayniqsa geografik jihatdan tarqoq guruhlar o‘rtasida va amalga oshirishga yondashuvlar sifatida	4%
Loyiha holati haqida ma’lumot olish	Vaziyatni kuzatish ma’lumotlari va holat hisoboti olinmoqda	3%

Xatolarni kuzatish va ularni tuzatish	Sinov guruhi, tahlilchilar va qo‘llab-quvvatlash markazi xodimlaridan ma’lumotlarni boshqaring	2%

Jarayonga asoslangan o‘zgarishlarni boshqarish yechimi bilan o‘zgarishlarni boshqarish xarajatlari va vaqtni tejash loyiha turiga va ilgari qo‘llanilgan o‘zgarishlarni boshqarish yondashuviga bog’liq bo‘lishi mumkin. Biroq, ushbu jadval shuni ko‘rsatadiki, jarayonga asoslangan o‘zgarishlarni boshqarishning samarali vositalari qo‘shimcha vaqtdan ko‘proq narsani tejash imkonini beradi.



6.1-rasm. Jarayonga asoslangan o‘zgarishlarni boshqarish[17].

Jarayonga asoslangan o‘zgarishlarni boshqarish orqali dasturiy ta’minotni ishlab chiqish tezligining eng sezilarli yaxshilanishiga nuqsonlar sonini kamaytirish orqali erishish mumkin. Haqiqatan ham, agar sifatni yaxshilashga katta e’tibor berilsa, bu maqsadga erishish unumdorlikni oshiradi, chunki muammolarni hal qilish uchun kamroq vaqt sarflanadi. Ko‘pgina tadqiqotlar shuni ko‘rsatdiki, CMM (Change Management Model)ga asoslangan dasturiy ta’minotni ishlab chiqishni takomillashtirishni amalga oshirgan tashkilotlardagi dasturiy ta’minotlar

[17] Орлов С.А. Цилькер Б.Я. Технологии разработки программного обеспечения: современный курс по программной инженерии: учебник для вузов. — 4-е изд.— СПб: Питер, 2012.

kamroq nuqsonlarga ega bo‘lgan va umumiy samaradorlikni oshirgan. CMMga asoslangan dasturiy ta’minotni ishlab chiqishni takomillashtirishni amalga oshirgan 13 ta tashkilotni chuqur o‘rganish sifat yaxshilanishi 39% ga chiqarilgandan keyingi nuqsonlar 19%ga kamaygan va samaradorlikning o‘rtacha 35% ga yaxshilanishini ko‘rsatdi.

CMM ni joriy qilgan 807 ta tashkilotning yana bir tadqiqoti shuni ko‘rsatdiki, tashkilot CMMning bir darajasidan ikkinchi darajasiga o‘tganda nuqsonlarning kamayishi va samaradorlikning oshishi kuzatiladi. Tadqiqot CMM modelining birinchi darajasidan ikkinchi darajasiga o‘tishda nuqsonlar sonining 50% ga (loyiha boshqaruvi ishlaydi) va uchinchi darajaga o‘tishda yana 20% ga kamayganligini ko‘rsatdi (o‘zgarish jarayonlari tartibga solinadi va ishlaydi). Xuddi shunday, CMM modelining birinchi darajasidan ikkinchi darajasiga o‘tishda samaradorlik 30% ga, uchinchi darajaga o‘tishda esa yana 20% ga oshdi.

Samarali, boshqariladigan o‘zgarishlarni boshqarish jarayonining yana bir jihati loyiha jadvaliga yaxshiroq rioya qilishdir. Loyihalar izchil jarayonlarga amal qilganligi va samarali jarayon muammolarni erta aniqlash va hal etishi sababli, loyihani rejalashtirish ancha aniqroq bo‘ladi.

Loyiha jadvalining ko‘rsatilgan to‘g’riligi dasturiy ta’minotni ishlab chiqish tashkilotlariga raqobatbardosh ustunlikka ega bo‘lish, shuningdek, o‘zlarining ishlab chiqish guruhlariga ko‘proq ishonish imkonini beradi.

AllFusion tizimi

Computer Associates kompaniyasining AllFusion dasturiy instrumentlar oilasi - bu korxonada axborot tizimlarini loyihalash, joylashtirish va boshqarish uchun integratsiyalashgan yechimlarni taqdim etish vositasidir. AllFusion ishlab chiqish jarayonlarini avtomatlashtirish va soddalashtirish orqali keng ko‘lamli ishlab chiqish metodologiyalarini qo‘llab-quvvatlaydi.

Modellashtirish vositalari (CASE) va dasturiy ta’minotni ishlab chiqishning barcha bosqichlarini qo‘llab-quvvatlaydi. Dasturiy ta’minotni ishlab chiqish uchun modellashtirish va o‘zgartirish va konfiguratsiyani boshqarish vositalari tashkilotlarga korporativ axborot tizimlarini modellashtirish, loyihalash va joriy etish imkonini beradi. Bularning barchasi loyiha va jarayonlarni boshqarish vositalari bilan

birgalikda o‘zgarishlarga o‘z vaqtida javob berishga va dasturiy ta’minotni ishlab chiqish vaqtini tezlashtirishga yordam beradi.

AllFusion tizimining asosiy komponentlari:

– Ma’lumotlar bazalari, biznes jarayonlari va dasturiy ta’minot komponentlarini modellashtirish vositalari (CASE);

– O‘zgartirish va konfiguratsiyani boshqarish vositalari (CCM);

– Loyiha va jarayonlarni boshqarish vositalari (PM).

6.2-rasm. AllFusion dasturiy instrumentlar oilasi[18].

Murakkab korporativ ilovalarni ishlab chiqish jarayonlarini boshqarish yangi texnologiyalar va ko‘p platformali muhitlarga moslasha oladigan yechimni talab qiladi. CA Technologies kompaniyasi “AllFusion Change Management Suite” nomli keng qamrovli o‘zgarishlar va konfiguratsiyalarni boshqarish (CMM) uchun vositalar to‘plamini taklif qiladi. Bu vositalar tashkilotlarga dasturiy ilovalarni ishlab chiqish hayotiy siklining barcha bosqichlarini boshqarish va korxona bo‘ylab integratsiyalashgan dasturiy ta’minotni moslashtirish vazifalarini, jumladan, ko‘p platformali va veb-ilovalarni ishlab chiqish imkonini beradi. To‘plam quyidagi dasturiy mahsulotlarni o‘z ichiga oladi: AllFusion Harvest Change Manager, AllFusion Change Manager

[18] Дубейковский, В. И. Эффективное моделирование с CA ERwin® Process Modeler: BPwin; AllFusion Process Modeler : практическое – 2-е изд., испр. и доп. – Москва : Диалог-МИФИ, 2009.

Enterprise Workbench va asosiy kompyuter dasturlari: AllFusion Endevor Change Manager, AllFusion CA-Librarian for z / OS and OS / 390 va AllFusion CA-Panvalet for z / OS va OS / 390.

Ushbu tizimdagi dasturiy mahsulotlar:

– AllFusion Harvest Change Manager (CCC / Harvest) - o‘zgarishlar, versiyalar, konfiguratsiyalarni boshqarish;

– AllFusion Change Manager Enterprise Workbench – server dasturlari uchun o‘zgartirish, versiya, konfiguratsiyani boshqarish;

– AllFusion Endevor Change Manager - asosiy kadrlar uchun o‘zgartirish, versiya, konfiguratsiyani boshqarish;

– AllFusion CA-Librarian - server dasturlari ishlab chiqish uchun kutubxona boshqaruvi;

– AllFusion CA-Panvalet - ishlab chiqilgan ilovalar kodini boshqarish (server dasturlari uchun).

AllFusion Change Management (ca.com/allfusion/) asosiy, taqsimlangan va turli ilovalarni ishlab chiqish muhitlari uchun vositalarni taklif qiladi va quyidagi dasturiy mahsulotlardan iborat:

– AllFusion Harvest Change Manager - taqsimlangan o‘zgarishlar va konfiguratsiyalarni boshqarish;

– AllFusion Endevor Change Manager - meynfreymlar uchun o‘zgartirish va konfiguratsiyani boshqarish;

– AllFusion Change Manager Enterprise Workbench - bir nechta platformalar uchun ishlab chiqish boshqaruvi.

AllFusion Change Management Suite bir qator qo‘shimcha vositalarni o‘z ichiga oladi:

– konfiguratsiyani boshqarishni avtomatlashtirish;

– parallel yoki qo‘shma manba versiyalarining integratsiyasi;

– AllFusion Endevor Change Manager nazorati ostida kodni bevosita tahrirlash;

– ob’ektlar va ilovalarni o‘zgartirish boshqaruvi;

– NATURAL va PREDICT muhitlaridan foydalangan holda rivojlanishni boshqarish.

AllFusion o‘zgarishlarni boshqarish vositalari o‘zgarishlarni boshqarish funktsiyalarining to‘liq spektrini o‘z ichiga oladi - versiyani nazorat qilish, o‘zgarishlarni kuzatish, konfiguratsiyani boshqarish, tuzilishni boshqarish, versiyalarni boshqarish hamda bir vaqtning o‘zida va birgalikda ishlab chiqishni boshqarish. AllFusion Change Management dasturiy to‘plamiga foydalanish mumkin bo‘lgan eng yaxshi jarayonlarni

o‘zgartirish kutubxonasi kiritilgan bo‘lib, u tashkilotning jarayon talablariga javob berish imkoniyatini oshiradi. Ushbu moslashuvchanlik bilan jarayonlar ham doimiy ravishda rivojlanishi mumkin, bu esa doimiy optimallashtirishga olib keladi.

Maqsad dasturiy ta’minotni ishlab chiqish sifati va tezligini oshirish bo‘lsa, o‘zgarishlarni boshqarish vositalarining jarayon bilan to‘liq integratsiyalashuvi juda muhimdir. AllFusion to‘liq integratsiyalashgan vosita bo‘lib, u ko‘plab mashhur ishlab chiqish vositalari, jumladan WebSphere Studio va Visual Studio.NET bilan integratsiyani ham taklif etadi. Bundan tashqari, CAning Unicenter ServicePlus Service Desk va Unicenter Software Delivery vositalari o‘zgarishlarni boshqarishni kengroq korporativ tizim ma’muriyati bilan birlashtirib, mahsulotning to‘liq hayot aylanishini boshqarishni taklif qiladi.

O‘zgarishlarni boshqarish va o‘zgartirish jarayonini soddalashtirish orqali ishlab chiqish jarayonining murakkabligini nazorat qilish orqali AllFusion o‘zgarishlarni boshqarish vositalari axborot tizimlari samaradorligini oshiradi va biznes qiymatini maksimal darajada oshiradi.

6.3. Tekshirish uchun konstruksiyalash

Tekshirish uchun konstruksiyalash deganda dasturiy ta’minotni shunday yaratish tushuniladiki, xatolar dasturiy ta’minotni yozuvchi muhandis tomonidan hamda mustaqil sinov va operatsion faoliyat davomida osongina topilishi mumkin. Tekshirish uchun konstruksiyalashni qo‘llab-quvvatlaydigan o‘ziga xos usullar quyidagilarni o‘z ichiga oladi: kodni tekshirishni qo‘llab-quvvatlash uchun kodlash standartlari, modulli test, avtomatlashtirilgan testni qo‘llab-quvvatlash uchun kodni tashkil qilish, murakkab yoki tushunish qiyin til tuzilmalaridan foydalanishni cheklash va boshqalar.

“Tekshirish uchun konstruksiyalash” dasturiy ta’minotni shunday qurish kerakligini anglatadiki, dasturiy ta’minotning o‘zi nosozlik sababini topishga yordam beradi, mustaqil sinov bosqichida ham turli tekshirish usullarini qo‘llash uchun shaffof bo‘ladi (masalan, sinov muhandislari) va ish paytida, yuzaga keladigan xatolarni tezda aniqlash va tuzatish qobiliyati ayniqsa muhimdir.

Verifikatsiya odatda qoida, standart yoki spetsifikatsiyaga muvofiqligini ta’minlash uchun ichki sifat boshqaruvi jarayonidir. Validatsiya va verifikatsiya o‘rtasidagi farqni eslab qolishning oson usuli shundan iboratki, validatsiya "siz to‘g’ri mahsulotni yaratganingizni"

tasdiqlaydi va verifikatsiya "siz mahsulotni o‘zingiz xohlagan tarzda yaratganingizni" tasdiqlaydi.

– verifikatsiya - deyarli har doim amalga oshiriladi, belgilangan talablarga ega bo‘lgan mahsulotlarning xususiyatlarini tekshirish usuli bilan amalga oshiriladi, natijada mahsulotning belgilangan talablarga muvofiqligi to‘g’risida xulosa chiqariladi.

– validatsiya - kerak bo‘lganda amalga oshiriladi, belgilangan foydalanish shartlarini tahlil qilish va mahsulot xususiyatlarining ushbu talablarga muvofiqligini baholash orqali amalga oshiriladi, natijada mahsulotning xavfsizligi va undan foydalanish imkoniyati to‘g’risida xulosa chiqariladi.

Validatsiyaning keng ta’rifi uni dasturiy ta’minotni sinovdan o‘tkazishga tenglashtiradi. Bunday holda, tekshirishning ikkita asosiy usuli mavjud:

Dinamik tekshirish, shuningdek, eksperiment, dinamik test yoki oddiygina testlash sifatida ham tanilgan. Bu dasturiy ta’minotdagi xatolarni topish uchun foydalidir.

Statik tekshirish, shuningdek, tahlil yoki statik test sifatida ham tanilgan. Bu dasturning to‘g’riligini tekshirish uchun foydalidir. Garchi bu dasturiy ta’minot aslida ishlaydigan jarayon va statik tekshirish taklif qilgan narsa o‘rtasida bir yoki bir nechta ziddiyatlar mavjud bo‘lganda noto‘g’ri ijobiy natijalarga olib kelishi mumkin.

Dinamik tekshirish dasturiy ta’minotning ishlash vaqtida amalga oshiriladi va dasturiy ta’minotning harakatini dinamik ravishda tekshiradi; Bu odatda “Test bosqichi” deb nomlanadi. Tekshirish – bu umumiy ko‘rib chiqish jarayonidir. Sinovlar hajmiga qarab, biz ularni uchta oilaga bo‘lishimiz mumkin:

– Kichik hajmda test: bitta funktsiya yoki sinfni sinovdan o‘tkazuvchi test (Unit Test)

– Katta hajmdagi test: Masalan, sinflar guruhini sinovdan o‘tkazadigan test

– Modul testi (bitta modul).

– Integratsiya testi (bir nechta modul).

– Tizim testi (butun tizim).

– Qabul qilish testi: dasturiy ta’minotni qabul qilish mezonlarini tekshirish uchun rasmiy defi testi zarur.

– Funktsional test.

– Funktsional bo‘lmagan test (ishlash, stress testi).

Dinamik dasturiy ta’minotni tekshirishning maqsadi - faoliyat natijasida yuzaga kelgan xatolarni aniqlash (masalan, biokimyoviy ma’lumotlarni tahlil qilish uchun tibbiy dasturiy ta’minotning mavjudligi); yoki bir va bir nechta amallarni qayta bajarish orqali (masalan, veb-server uchun stress testi, ya’ni amalning joriy natijasi harakat boshida bo‘lgani kabi to‘g’ri yoki noto‘g’riligini tekshirish).

Statik tekshirish - bu dasturni ishga tushirishdan oldin kodni tekshirish orqali muvofiqligini tekshirish jarayoni. Masalan:

- Tekshirish bo‘yicha shartli belgilar
- Noto‘g’ri amaliyotlarni aniqlash
- Dasturiy ta’minot ko‘rsatkichlarini hisoblash
- Rasmiy tekshirish

Tahlil yo‘li bilan tekshirish - Tahlilning tekshirish usuli darsliklardan klassik usullar yoki umume’tirof etilgan kompyuter usullaridan foydalangan holda tekshirish, matematik hisoblar, mantiqiy baholash va hisob-kitoblar bilan tekshirishda qo‘llaniladi. Tahlil moslikni aniqlash uchun namunalar olish va o‘lchangan ma’lumotlar va kuzatilgan test natijalarini hisoblangan kutilgan qiymatlar bilan taqqoslashni o‘z ichiga oladi.

Jiddiyroq aniqlangan holda, tekshirish faqat statik sinovga teng va artefaktlarga nisbatan qo‘llanilishi uchun mo‘ljallangan. Bundan tashqari, tekshirish (butun dasturiy ta’minot mahsuloti) dinamik sinovga teng bo‘ladi va u ishlaydigan dasturiy mahsulotga (va uning artefaktlariga emas, talablardan tashqari) qo‘llanilishi uchun mo‘ljallangan. E’tibor bering, talablarni tekshirish statik yoki dinamik tarzda amalga oshirilishi mumkin.

Dasturiy ta’minotni verifikatsiya qilish quyidagicha savol qo‘yadi: "Biz mahsulotni to‘g’ri ishlab chiqaryapyotibmizmi?", ya’ni dasturiy ta’minot o‘z xususiyatlariga mos keladimi? (Uy o‘z chizmalariga qanday mos keladi).

Dasturiy ta’minotni validatsiya qilish vaqtida savol beriladi: "Biz to‘g’ri mahsulot ishlab chiqayotibmizmi?"; ya’ni dastur foydalanuvchi xohlagan narsani qiladimi? (Uy egasining ehtiyojlari va xohishlariga mos kelsa).

Bugungi kunda dasturiy ta’minot sifatini ta’minlash muammosi har qachongidan ham dolzarb bo‘lib, uni hal qilish uchun bugungi kunda ko‘plab verifikatsiya qilish va kodlarni validatsiya qilish vositalari taklif etiladi. Shu bilan birga, nafaqat instrumentlarning o‘zini amalga oshirish, tegishli vakolatlarni rivojlantirish va sinov strategiyasini yaratish

muhimdir. Inson omili bilan bog’liq xavflarni bartaraf etish muhim nuqtalar va nuqsonlarni avtomatik ravisgda aniqlashni talab qiladi.

Dasturiy ta’minot deyarli har qanday infratuzilmaning markazida bo‘lib, sifatni ta’minlash har qachongidan ham dolzarbdir. Deyarli barcha kompaniyalar o‘z faoliyatida narsalar interneti, biznes intellekti, sun’iy intellekt, bulutlar, ijtimoiy tarmoqlar va hokazolardan allaqachon foydalanmoqda. An’anaviy axborot texnologiyalari va o‘rnatilgan tizimlar o‘z o‘rnini hamma joyda keng tarqalgan dasturiy ta’minotga bo‘shatib bermoqda va korxonalarning raqamli transformatsiyasining muvaffaqiyati xizmatlarning ishonchliligi va mavjudligi uchun barcha sanoat talablariga javob beradigan dasturiy ta’minot tizimlarining ishlashiga bog‘liq.

Tashkilotlar bugungi kunda axborot texnologiyalari byudjetining qariyb 30 foizini sifatni ta’minlash va sinovdan o‘tkazishga sarflaydi, bu ajablanarli emas - barcha tizimlarning yarmidan ko‘pi biznes uchun muhim. Shu bilan birga, kompaniyalar va tashkilotlar o‘zgarishlarga javob berish va xavfsiz dasturiy ta’minotni nazorat ostida chiqarish uchun rasmiylashtirilgan jarayonlar va usullarni amalga oshirish uchun imkon qadar moslashuvchan bo‘lishi kerak. Internet orqali dasturiy ta’minotni avtomatik yangilash mexanizmlari, DevOps usullari va "uzluksiz dasturiy ta’minot injiniringi" faol joriy etilmoqda, bu esa umumiy ish qobiliyatini so‘nggi paytlardagiga qaraganda ancha puxtaroq talab qilinadigan doimiy tekshirish va tasdiqlash jarayonlariga bo‘lgan ehtiyojni oshiradi.

Foydalanuvchi talabi va sifatni yaxshilash uchun tezkor va uzluksiz dasturiy ta’minotni ishlab chiqish usullari dasturiy ta’minotni ishlab chiquvchilar va foydalanuvchilarga ishlatish uchun qulay vositalar bilan qo‘llab-quvvatlanishi kerak. Bundan tashqari, sifatni baholash uchinchi shaxs - akkreditatsiyalangan laboratoriya yoki sertifikatlashtirish markazi tomonidan amalga oshirilishi mumkin.

Dasturiy ta’minotni verifikatsiya va validatsiya qilish usullarini tanlash ishlab chiqish modeliga (V-model, kaskad, spiral va boshqalar) va standartga (ISO / IEC 25000 SQUARE, ISO / IEC 12207: 2017) bog’liq. Dasturiy ta’minotni ishlab chiqish jarayonining sifatini tekshirish va tasdiqlash jarayonlarini tartibga soluvchi ISO / IEC 12207 standarti, shuningdek, har bir ishlab chiqish vazifasi sinov jarayoni bilan bog’liq bo‘lgan V-modeli bilan tartibga solinadi. Masalan, modulli testlar manba kodining past darajadagi arxitekturaga muvofiqligini tasdiqlaydi, integratsiya testlari avval sinovdan o‘tgan komponentlarning muvofiqligini (integratsiyasini) tasdiqlaydi, tizim testlari to‘liq

integratsiyalangan mahsulot spetsifikatsiyalarga javob beradimi yoki yo‘qligini tasdiqlaydi va qabul qilish testlari mahsulot foydalanuvchi kutganiga mos keladimi yoki yo‘qligini tasdiqlaydi.Dasturiy mahsulot sifati bilan bog’liq ISO 25000 seriyali standartlarda quyidagi dasturiy ta’minot xususiyatlari ajralib turadi:

– funksional yaroqlilik – mahsulot yoki tizimning belgilangan sharoitlarda dasturiy ta’minotdan foydalanishda e’lon qilingan funksiyalarga, e’lon qilingan va nazarda tutilgan ehtiyojlarga muvofiqligi darajasi;

– texnik xizmat ko‘rsatish - mahsulot yoki tizimni o‘zgartirish, uni sozlash yoki atrof-muhit yoki talablarning o‘zgarishiga moslashtirishning qulayligi va moslashuvchanligi;

– foydalanish qulayligi – foydalanishning berilgan kontekstida foydalanuvchilar tomonidan belgilangan maqsadlarga samarali erishish uchun mahsulot yoki tizim bilan ishlashning qulayligi;

– Xavfsizlik – odamlar yoki tizimlar o‘z imtiyozlariga muvofiq ma’lumotlardan foydalanish imkoniyatiga ega bo‘lishi sharti bilan mahsulot yoki tizimning ma’lumotlarni himoya qilish darajasi;

– nisbiy unumdorlik - berilgan sharoitlarda unumdorlik va foydalaniladigan resurslar miqdori nisbati.

Kodlash bosqichida ko‘pincha statik tahlil vositalaridan foydalaniladi, bu sizga kodning standartlarga muvofiqligini to‘g’ridan-to‘g’ri integratsiyalashgan ishlab chiqish muhitida nazorat qilish imkonini beradi. Shu bilan birga, statik tahlil doirasida modullar tuzilishi va dasturiy ta’minot arxitekturasining xususiyatlari tekshiriladi, shuning uchun 2-rasmda. 2, statik tahlil vositalari maydoni kod yozish bilan bir xil darajada joylashgan, ammo arxitektura verifikatsiyasi bilan bog’liq testlarni oladi. Statik tahlil vositalari barcha to‘rtta sifat atributlari bilan bog’liqdir.

6.4. Tekshirish vositalari

XUnit vositalari ishlab chiqilgan har bir modulning to‘g’ri ishlashini tekshirish uchun ishlatiladi va shu bilan birga maksimal kod qamrovini ta’minlaydi. XUnit frameworklari testlarni avtomatlashtirish texnologiyalari orasida eng keng tarqalgan bo‘lib, test vaziyatlarni maxsus tillarda tasvirlash va ularni avtomatik ravishda bajarish imkonini beradi.

Belgilangan test vositalari test ma’lumotlarini yaratadi, bu esa turli xil integratsiyalangan modullarning to‘g’ri ishlashini tekshirish imkonini

beradi. Belgilash va almashtirish vositalari tizimning to‘g’riligi va to‘liqligini tekshirish uchun, shu jumladan qabul qilish sinovlari paytida qo‘llaniladi. Ushbu vositalar test qiluvchilarning ilova bilan o‘zaro aloqalarini qayd etib, keyinchalik avtomatik ravishda ishga tushirilishi mumkin bo‘lgan test ssenariylarini yaratadi. Dasturlash tillari kabi test vositalari ham shunchalik ko‘p.

Xizmat ko‘rsatishni nazorat qilish vositalari. Ular dastlabki kodni tahlil qilish va modullilik, o‘qish va hokazolar qoidalariga muvofiqligini tekshirish imkonini beradi. Foydalanish qulayligini boshqarish vositalari. Ular haqiqiy foydalanuvchilar bilan ishlash jarayonida dasturiy mahsulotni baholash uchun ishlatiladi. Shu bilan birga, bunday vositalar foydalanuvchi interfeysini foydalanuvchilarning o‘zlari ishtirokisiz tekshirish imkonini beradi.

Xavfsizlikni nazorat qilish vositalari. Tizimdagi zaifliklarni aniqlash va kerakli funksionallikni saqlab qolgan holda ma’lumotlarni himoya qilishi yoki yo‘qligini aniqlash imkonini beradi. Xususan, himoya sinov vositalari skanerlash va zaifliklarni topish va ulardan foydalanishga qaratilgan boshqa harakatlar orqali dasturiy ta’minot tizimi yoki tarmoqqa hujumlarni imitatsiya qiladi. Ular axloqiy yoki oq xakerlik vositalari deb ham ataladi.

Unumdorlikni tekshirish vositalari. Tizimning ish bosimi ostida qanchalik tez ishlashini aniqlash imkonini beradi. Ushbu vositalar ilovadagi nuqsonlarni topish uchun mo‘ljallanmagan, lekin o‘lchanadigan xususiyatlarni baholash uchun ishlatiladi: javob vaqti, o‘tkazish qobiliyati va boshqalar.

Uzluksiz verifikatsiya va validatsiya vositalari. Uzluksiz sifatni ta’minlash tamoyili rivojlanishning barcha bosqichlarida verifikatsiya va validatsiya jarayonlarining yaqin integratsiyasini nazarda tutadi. Odatiy misol sinov asosida ishlab chiqish bo‘lib, unda dasturiy ta’minot ishlab chiqilishidan oldin sifat maqsadlari belgilanadi. DevOps asoslaridan biri bo‘lgan uzluksiz integratsiyaning afzalliklarini hisobga olgan holda, nafaqat verifikatsiya va validatsiyani avtomatlashtirishga, balki ushbu jarayonlarni ishlab chiqish tsikliga integratsiyalashga yordam beradigan vositalarga katta e’tibor qaratilmoqda.

Jenkins, Travis CI, Bamboo, GoCD, Ansible vositalari uzluksiz integratsiyalashgan muhitda sifat xususiyatlarini tekshirish imkonini beradi. Verifikatsiya va validatsiyani dasturiy ta’minotning hayot aylanishiga kiritish talab qilinadi, bu esa jarayonlarni ishlab chiquvchilar uchun mos keladigan avtomatik va shaffof bajarilishini ta’minlaydi. Bu

instrumentlar bir nechta sifat xususiyatlarini tasdiqlaydi va global baholash va vizualizatsiya uchun ma’lumotlarni taqdim etadi. Uzluksiz verifikatsiya va validatsiyajarayonlari modulli sinov, belgilash va almashtirish, hamda statik tahlil vositalaridan ham foydalanishi mumkin.

Ro‘yxatdagi barcha turdagi vositalar yordamida bajarilgan harakatlarni boshqarish va umuman verifikatsiya va validatsiya jarayonini boshqarish uchun test ishini boshqarish vositalaridan foydalaniladi: Test Link, Test Rail, Microfocus Quality Center, VSTS, IBM Rational Quality Manager, XStudio va boshqalar.

Zamonaviy jamiyat iqtisodiyotning barcha tarmoqlarining dvigateliga aylangan dasturiy ta’minotga tobora ko‘proq qaram bo‘lib bormoqda va shuning uchun dasturiy ta’minot sifatiga talablar ortib bormoqda. Tegishli jarayonlarni avtomatlashtirish usullari va texnologiyalaridan foydalanish infratuzilmalarni himoya qilishning ajralmas elementiga aylanmoqda. Zamonaviy dunyoda kiberxavfsizlik tahdidlari sonining ortib borishi va ilovalardan foydalanishdagi noqulayliklar tufayli yuzaga keladigan muammolar bilan bog’liq xavflar tobora ko‘payib bormoqda. Bu shuni anglatadiki, biz tekshirish va tasdiqlash texnologiyalarini rivojlantirishni jadallashtirishimiz, shuningdek, tizimlar ishonchliligini oshirish bo‘yicha sa’y-harakatlarni ikki baravar oshirishimiz kerak. Xususan, dasturiy ta’minot imkoniyatlarining saqlanib qolishi va uning nosozlik sharoitida ishlashi stsenariylari talab qilinadi.

Dasturiy ta’minot sifatini ta’minlash strategiyasining asosini tashkil etishi kerak bo‘lgan doimiy verifikatsiya va validatsiya mahsulot chiqarilgandan so‘ng muhim zaifliklarni aniqlash va yangi hujumlar paydo bo‘lishining oldini olishga yordam beradi. Simsiz tarmoq ulanishlari yordamida tuzatishlar va o‘zgartirishlar kiritish uchun sizga moslashuvchanlik kerak. Tizimlarda eng so‘nggi dasturiy ta’minot yangilanishlari o‘rnatilmagan bo‘lsa, ishga tushishining oldini olish uchun mexanizmlar kerak bo‘ladi. Ushbu tizimlarga transport vositalari, ishlab chiqarish liniyalari va ayniqsa yuqori darajadagi xavfsizlikni talab qiladigan boshqa uskunalar kiradi. Bu tibbiy texnologiyalarga ko‘proq taalluqlidir - bu sifatni nazorat qilishning ierarxik tizimini talab qiladi.

Verifikatsiya va validatsiya tizimlarini tanlash ko‘plab omillar bilan belgilanadi. Amaldagi rivojlanish muhitiga qarab, turli tashkilotlardagi tegishli jarayonlar o‘ziga xos xususiyatlarga ega va turli xil vositalar kombinatsiyasiga asoslanadi. Shu bilan birga, nafaqat asboblarning o‘zini amalga oshirish, tegishli vakolatlarni rivojlantirish va sinov strategiyasini

yaratish muhimdir. Inson omili bilan bog'liq xavflarni bartaraf etish muhim nuqtalar va nuqsonlarni ilg'or avtomatik aniqlashni talab qiladi.

Sun'iy intellekt texnologiyalarining joriy etilishi bilan tegishli tizimlar faoliyatining shaffofligini ta'minlash zarurati tug'iladi - mijozga kredit berish mumkinmi yoki qanday qilib neyron tarmoq qanday qoidalardan foydalanishini tushunish kerak. robot-avtomobil bir nechta xavfli vaziyatlarning qo'shilishida reaksiyaga kirishadi. Bunday hollarda klassik regressiya testlari va kuzatuvlar yordam bermaydi. Yangi avlodlarni verifikatsiya va validatsiya vositalarida katta ma'lumotlarga asoslangan, tahlil qilish, o'z-o'zini o'rganish va dasturiy ta'minot sifatini avtomatik ravishda yaxshilashga qodir bo'lgan aqlli mexanizmlar tobora ko'payib boradi.

Aristotel: "Komillik harakat emas, balki odatdir" degan. Instrumentlarning keng doirasi muhim, lekin asosiysi, dasturiy ta'minot sifatini ta'minlash madaniyatini yaratish va tegishli ko'nikmalarni egallashdir.

Dasturiy ta'minotni ishlab chiqishdagi asosiy muammolardan biri bu dasturiy ta'minotni tekshirishdir. Dasturiy ta'minotni tekshirish vositalari e'lon qilingan yakuniy dasturiy mahsulot talablarini tasdiqlash uchun maxsus yaratilgan, dasturiy ta'minotni tekshirishning maqsadi xatolar, noto'g'ri xususiyatlar va dastur zaifligini aniqlashdir.

Dasturiy ta'minotni tekshirish usullarining yangi tasnifini shakllantirish dolzarb muammo bo'lib, mavjud dasturiy ta'minotni tekshirish usullarini ko'rib chiqish, ularning afzalliklari va kamchiliklarini aniqlash imkonini beradi. Mavjud usullarni tasniflash va tahlil qilish SMT - hal qiluvchi printsipiga muvofiq kelajakdagi tadqiqotlar va dasturiy ta'minotni tekshirishning sintetik usulini ishlab chiqish uchun talablar va tavsiyalar ro'yxatini yaratishga imkon beradi.

Zamonaviy tekshirish usullarini empirik (ekspertiyadan foydalanadiganlar), rasmiy (dasturiy ta'minotni tekshirishning matematik apparatidan foydalanadigan) va rasmiy (dasturni ishga tushirish orqali tekshiradigan) va avtomatlashtirish darajalariga bo'lish mumkin, avtomatlashtirish darajalari qo'lda, avtomatik va avtomatlashtirilgan. Dasturiy ta'minotni tekshirishning asosiy maqsadlaridan biri yaratilgan dasturiy kodni texnik xususiyatlar va uning funksionallik talablariga muvofiqligini tekshirishdan iborat. Hujjatlar va dasturiy kodlarni me'yorlar va standartlar va ijrolar bilan tekshirishda mamlakatda, tarmoqda va tashkilotda o'rnatilgan ekspertizadan foydalaniladi. Imtihonning o'zi ixtisoslashtirilgan va umumiy bo'lishi mumkin.

6-bob bo‘yicha xulosalar

Ko‘pgina dasturiy ta’minot tizimlari vaqt o‘tishi bilan o‘zgaradi. Buning sabablari ko‘p. O‘zgarishlarni kutish (Ожидание изменений - anticipating change) dasturiy ta’minot muhandisligining harakatlantiruvchi kuchlaridan biridir. O‘zgarishlarga tayyor turish dasturiy ta’minot muhandislariga moslanuvchan dasturiy ta’minotni yaratishga yordam beradi, ya’ni ular dasturiy mahsulotni bazaviy tuzilmani buzmasdan yaxshilashlari mumkin. Refactoring — bu mavjud kodlar majmuasini qayta tuzish, tashqi tuzilishini o‘zgartirmasdan ichki tuzilishini o‘zgartirish uchun ajoyib texnologiya.

Martin Fauler refaktoring qilish uchun to‘rtta asosiy sababini aytib o‘tdi. Refactoring dasturiy ta’minot dizaynini yaxshilaydi, dasturiy ta’minotni tushunishni osonlashtiradi, xatolarni topishga yordam beradi va dasturni tezroq bajarilishiga yordam beradi. Refactoring bizga quyidagi qulayliklarni beradi: dasturiy ta’minotning dizaynini yaxshilaydi, dasturiy ta’minotni tushunishni osonlashtiradi, dasturiy ta’minot xatolarni topishda va tezroq dasturlashda yordam beradi.

Refaktoring – bu dasturning tashqi xulq-atvoriga ta’sir qilmasdan ichki strukturasini o‘zgartirish jarayonidir. Bunda asosiy maqsadlardan biri dastur ishini tushunishni osonlashtirishdir. O‘zgarishlarni boshqarish dasturiy ta’minotni ishlab chiqishni boshqarishning boshqariladigan yondashuvini o‘z ichiga oladi, shu bilan yetkazib beriladigan mahsulot bilan bog’liq muammolar xavfini kamaytiradi. Verifikatsiya uchun konstruksiyalash deganda dasturiy ta’minotni shunday yaratish tushuniladiki, xatolar dasturiy ta’minotni yozuvchi muhandis tomonidan hamda mustaqil sinov va operatsion faoliyat davomida osongina topilishi mumkin. “Verifikatsiya uchun konstruksiyalash” dasturiy ta’minotni shunday qurish kerakligini anglatadiki, dasturiy ta’minotning o‘zi nosozlik sababini topishga yordam beradi, mustaqil sinov bosqichida ham turli tekshirish usullarini qo‘llash uchun shaffof bo‘ladi.

Verifikatsiya va validatsiya tizimlarini tanlash ko‘plab omillar bilan belgilanadi. Amaldagi rivojlanish muhitiga qarab, turli tashkilotlardagi tegishli jarayonlar o‘ziga xos xususiyatlarga ega va turli xil vositalar kombinatsiyasiga asoslanadi. Shu bilan birga, nafaqat asboblarning o‘zini amalga oshirish, tegishli vakolatlarni rivojlantirish va sinov strategiyasini yaratish muhimdir. Inson omili bilan bog’liq xavflarni bartaraf etish muhim nuqtalar va nuqsonlarni ilg’or avtomatik aniqlashni talab qiladi. Yangi avlodlarni verifikatsiya va validatsiya vositalarida katta ma’lumotlarga asoslangan, tahlil qilish, o‘z-o‘zini o‘rganish va dasturiy

ta’minot sifatini avtomatik ravishda yaxshilashga qodir bo‘lgan aqlli mexanizmlar tobora ko‘payib boradi.

6-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotni ishlab chiqishda o‘zgarishlarni kutish tamoyilini tushuntirib bering.
2. Refactoring so‘zining ma’nosini aytib bering.
3. Refactoringning qanday turlarini bilasiz.
4. Refactoringning foydasini aytib bering.
5. Refactoringning zararini aytib bering.
6. Reinjiniring so‘zining ma’nosini aytib bering.
7. O‘zgarishlarni boshqarish deganda nimani tushunasiz?
8. O‘zgarishlarni boshqarish vositalarining darajalarini sanab bering.
9. Jarayonga asoslangan o‘zgarishlarni boshqarish deganda nimani tushunasiz?
10. Verifikatsiya uchun konstruksiyalash deganda nimani tushunasiz?

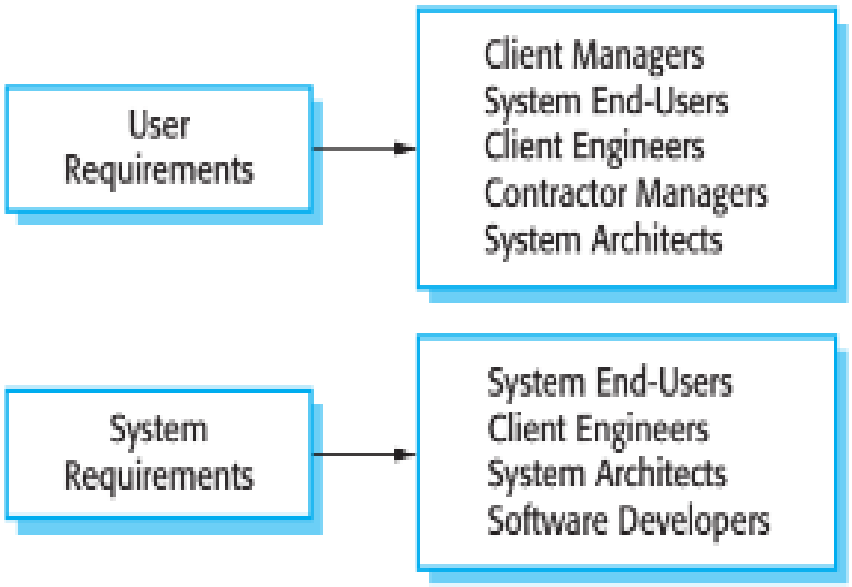
7-BOB. DASTURIY TA’MINOTNI KONSTRUKSIYALASHGA TAYYORLANISH

7.1. Dasturiy ta’minot uchun talablarni shakllantirish

Dasturiy ta’minot uchun talablar - bu tizim nima ish bajarish lozimligini tasvirlashdir. Talablar dasturiy ta’minot mijozlarini ehtiyojlarini aks ettiradi. Talablar injiniringi jarayonida ko‘pgina muammolar ko‘tariladi. Foydalanuvchi talablari’ va ‘Dasturiy ta’minot talablari’ terminlari orasida farq mavjud. Foydalanuvchi talablari va dasturiy ta’minot talablari quyidagicha izohlanishi mumkin:

1. Foydalanuvchi talablari bu diagrammalar bilan tabiiy tildagi bayonotlar.

2. Dasturiy ta’minot talablari bu dasturiy ta’minot tizimi funksiyalari, servislari va operativ cheklanishlarining batafsil tasvirlanishi. Siz talablarni turli xil darajada yozishingiz kerak chunki turli xil o‘quvchilar turli xil yo‘lda foydalanishadi.



7.1-rasm. Foydalanuvchi talablari va dasturiy ta’minot talablari[19].

Dasturiy ta’minot tizimi talablari funksional va funksional bo‘lmagan talablar sinflariga ajratiladi.

1. Funksional talablar Bu dasturiy ta’minot taminlashi lozim bo‘lgan servislarning bayonoti. Kiritilgan ma’lumo tlarga dasturiy ta’minot

[19] Маккарти, Джим. Правила разработки программного обеспечения: монография.Пер. с англ. - М.: СПб. Нижний Новгород : Русс. редакция; Питер, 2007.

qanday reaksiya ko‘rsatishi lozim, dasturiy ta’minot o‘zini bunday holatlarda qanday tutushi lozim

2. Funksional bo‘lmagan talablar Bu dasturiy ta’minot tomonidan taklif qilinayotgan servislar va funksiyalardagi cheklovlar. U o‘z ichiga vaqt cheklanishi, ishlab chiqarish jarayoni cheklanishi, beriladigan standartlar tomonidan cheklanishlarni olishi mumkin.

Dasturiy ta’minot talablari hujjati

Dasturiy ta’minot hujjati bu tizimni ishlab chiquvchilar nimani oshirishi lozimligini ifodalovchi rasmiy hujjatdir. U tizim uchun foydalanuvchi talablarini ham tizim talablarining batafsil spesifikatsiyasini ham o‘z ichiga oladi. Ba’zida foydalanuvchi va dasturiy ta’minot talablari bitta qilib tavsiflanadi. Bazi hollarda esa foydalanuvchi talablari hujjatning kirish qismi va tizim talablari asosiy qismni tashkil qiladi.

Talablar hujjatidan foydalanuvchi

Dasturiy ta’minot mijozlari

- Talablarni ko‘rsatish va talablar bajarilganligiga tekshirish uchun o‘qish.
- Mijozlar shuningdek talablarni o‘zgartirishi mumkin.

Boshqaruvchilar

- Tizimni narxlash va ishlab chiqishni rejalashritish uchun talablar hujjatidan

foydalanish.

Tizim injinerlari

- Ishlab chiqarilayotgan tizimni tushunish uchun talablardan foydalanish.

Tizimni testlovchi injinerlar

- Tizimni haqiqiylikka tekshirish uchun tizm talablaridan foydalanish

Tizimga xizmat ko‘rsatuvchi injinerlar

- Tizim va uning qismlari munosabatini tushunish uchun talablardan foydalanadi.

Dasturiy ta’minotning tahlili va konstruktsiyalash talablar spetsifikatsiyasini qo‘shimchaga aylantirishga yordam beradigan barcha tadbirlarni o‘z ichiga oladi. Talab spetsifikatsiyalari dasturiy ta’minotning barcha funktsional va funktsional bo‘lmagan taxminlarini belgilaydi. Ushbu spetsifikatsiya talablari kompyuterning hech qanday aloqasi bo‘lmagan aniq va tushunarli hujjatlar shaklida keladi. Dasturiy ta’minotni tahlil qilish va konstruktsiyalash - bu odamlar o‘qiy oladigan talablarni haqiqiy kodga aylantirishga yordam beradigan oraliq bosqich.

7.1-jadval

Talablar hujjatining strukturasi

Bo‘lim	Tavsifi
Muqaddima	Hujjatni kutilgan o‘quvchilarini aniqlash lozim
Kirish	Tizim muhimligini tasvirlash. Tizim funksionalligi qisqacha tasvirlanadi.
Glossariy	Hujjatda foydalanilgan texnik terminlarni aniqlash
Foydalanuvchi talablari	Foydalanuvchi uchun taminlangan servislarni tasvirlash
Tizim arxitekturas	Kutilgan tizim arxitekturasini yuqori darajali ko‘rinishi
Tizim talablari	Funksional va funksional bo‘lmagan talablarning batafsil ko‘rinishi
Tizim modellari	Tizim componentalari orasidagi munosabatlarni grafik tizimini ko‘rsatish
Tizim evolutsiyasi	Tizimga asoslanib fundamental taxminlarni tasvirlash
Ilova	Ishlab chiqarilayotgan ilova haqida batafsil ma’lumotlar; masalan, apparat ta’minot va ma’lumotlar bazasi
Index	Hujjat indekslari

Axborot oqimining diagrammasi - bu axborot tizimidagi ma’lumotlarni uzatishning grafik tasviri. U kiruvchi axborot oqimini, kommunikativ axborot oqimini va saqlangan ma’lumotlarni ko‘rsatishga qodir. DFD ma’lumotlarning tizim orqali qanday o‘tishi haqida hech narsa demaydi.

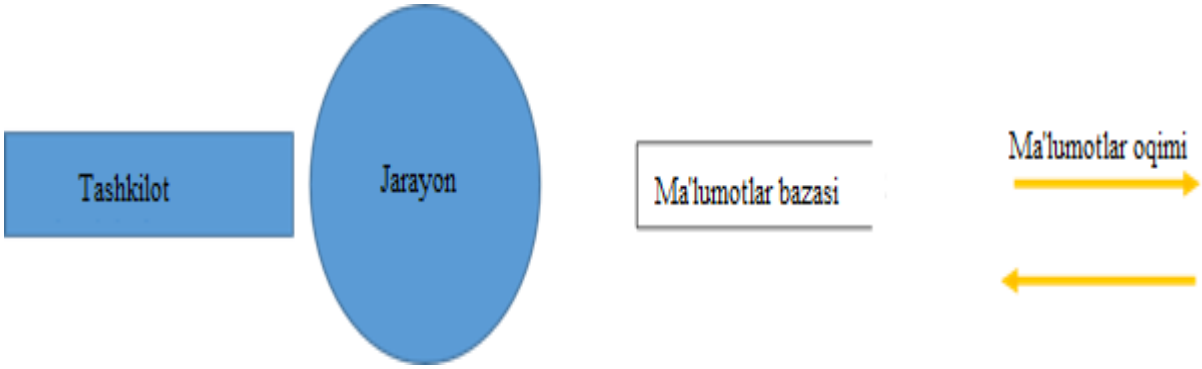
DFD va oqim diagrammasi o‘rtasidagi farq ko‘rinadi. Oqim diagrammasi dastur modullarida boshqaruv oqimini ko‘rsatadi. DFDlar turli darajadagi tizimdagi ma’lumotlar oqimini ko‘rsatadi. DFD -da hech qanday boshqaruv yoki tarmoqlanishlar mavjud emas.

DFD turlari

Ma’lumotlar diagrammalarining grafik yoki mantiqiy yoki jismoniy ko‘rinishi. Mantiqiy DFD - DFDning bu turi tizim jarayoniga va tizimdagi ma’lumotlar oqimiga qaratilgan. Masalan, bankning dasturiy ta’minot tizimida ma’lumotlar turli xil voqeliklar o‘rtasida qanday o‘tkaziladi.

DFD komponentlari

DFD manba, manzil, saqlash va taqdimotni quyidagi komponentlar to‘plami yordamida ifodalashi mumkin(7.2-rasm).



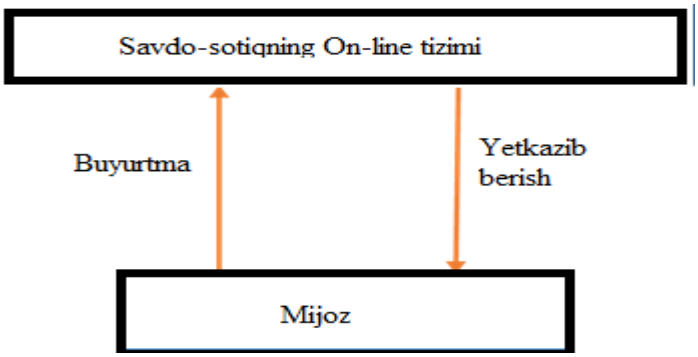
7.2-rasm. DFD komponentlari[20].

Haqiqiylik - bu ma’lumotlarning manbasi va manzilidir. Haqiqatlar o‘z nomlari bilan to‘rtburchaklar bilan ifodalanadi. Jarayon - Ma’lumotlar bo‘yicha bajariladigan harakatlar va harakatlar aylana yoki dumaloq qirrali to‘rtburchaklar bilan ifodalanadi. Ma’lumotni saqlash - ma’lumotlarni saqlashning 2 varianti - uni ikkala tomoni ham bo‘lmagan to‘rtburchaklar shaklida yoki bitta tomoni yo‘q bo‘lgan ochiq qirrali to‘rtburchaklar shaklida ko‘rsatish mumkin. Axborot oqimi - ma’lumotlarning harakatlanishini ko‘rsatgichli o‘qlar bilan ko‘rsatadi. Ma’lumotlarning o‘q tagidan harakatlanishi uning manbasi sifatida o‘qning boshigacha ko‘rsatiladi.

DFD darajasi

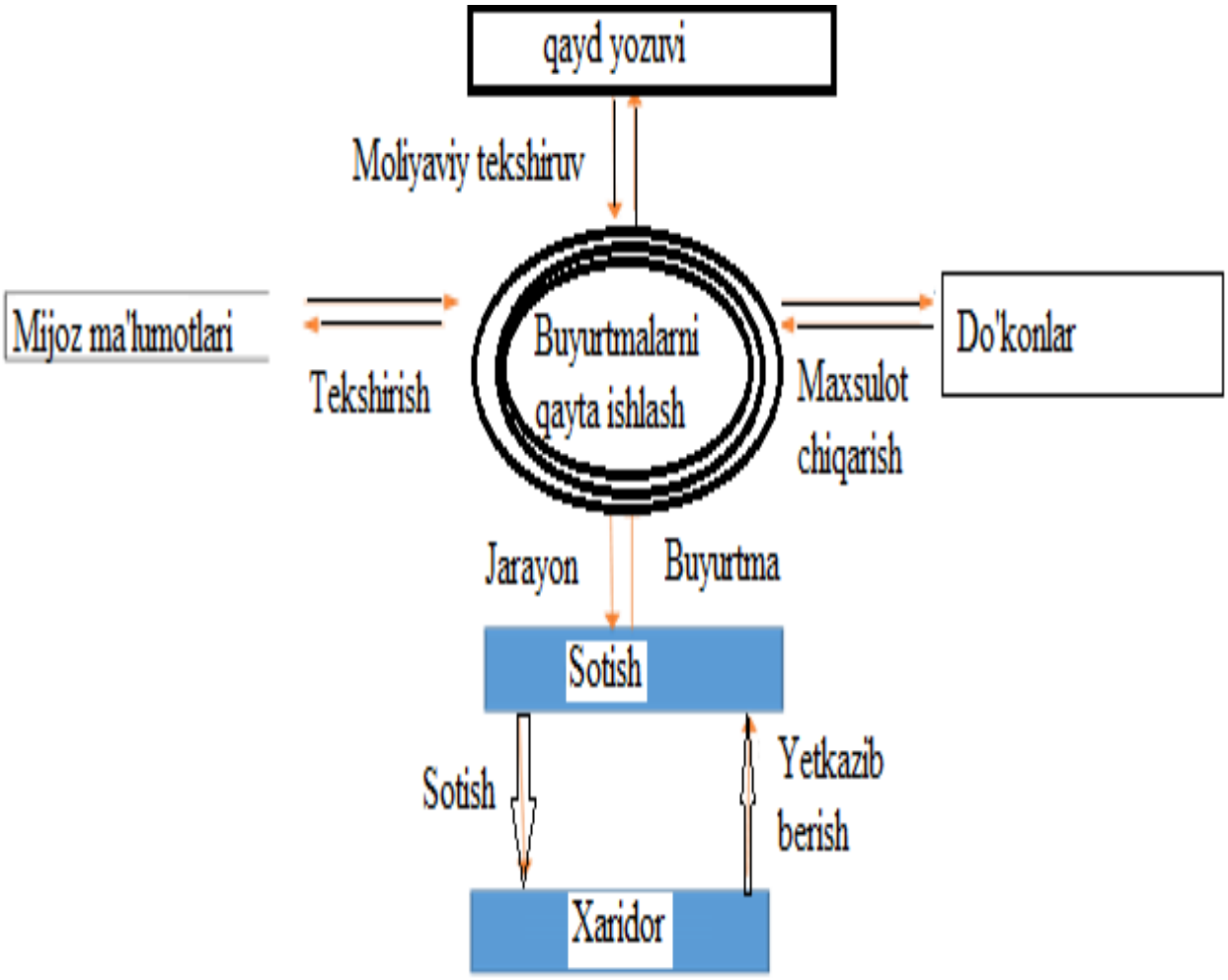
0 -daraja - DFD 0 -darajali DFD abstraktsiyasining eng yuqori darajasi, bu butun axborot tizimini bitta diagramma sifatida ko‘rsatadi, barcha asosiy tafsilotlarni yashiradi. 0 -darajali DFD -lar kontekst darajasidagi DFD -lar bilan bir xil(7.3-rasm).

[20] Маккарти, Джим. Правила разработки программного обеспечения: монография.Пер. с англ. - М.: СПб. Нижний Новгород : Русс. редакция; Питер, 2007.



7.3-rasm. DFD 1- darajasi[21].

1 -darajali - 0 -darajali DFD aniq, 1 -darajali DFDga bo‘linadi. 1 -darajali DFD tizimdagi asosiy modullarni va turli modullar orasidagi ma’lumotlar oqimini ko‘rsatadi. DFD 1 -darajasi, shuningdek, asosiy jarayonlar va axborot manbalarini eslatib o‘tadi.



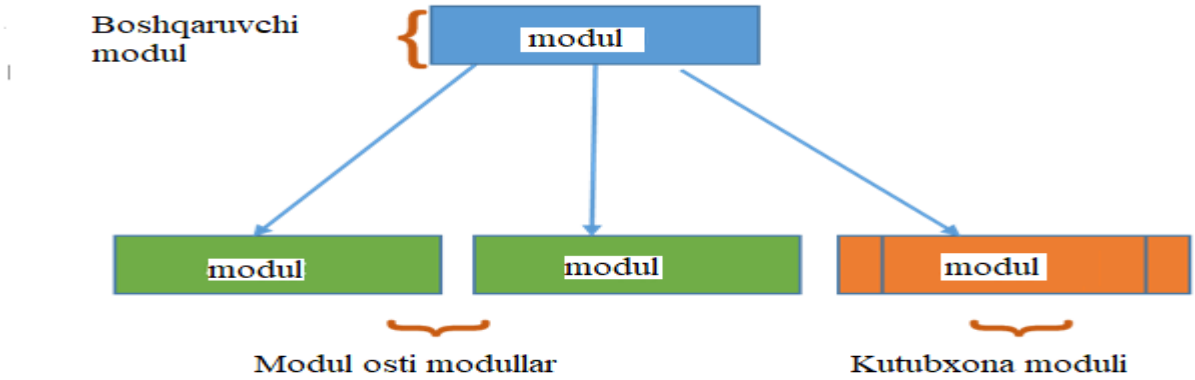
7.4-rasm. 2- darajasi.

[21] Зубкова Т.М. Технология разработки программного обеспечения. Учебное пособие. — Оренбург: Оренбургский государственный университет, 2017.

2 -daraja - bu darajada, DFD 1 -darajadagi ushbu modullar ichida ma’lumot qanday oqishini ko‘rsatadi. Yuqori darajadagi DFD -lar, agar kerakli spetsifikatsiya darajasiga erishilmasa, chuqurroq tushunish darajasiga ega bo‘lgan yanada aniqroq quyi darajadagi DFD -larga aylantirilishi mumkin(7.4-rasm).

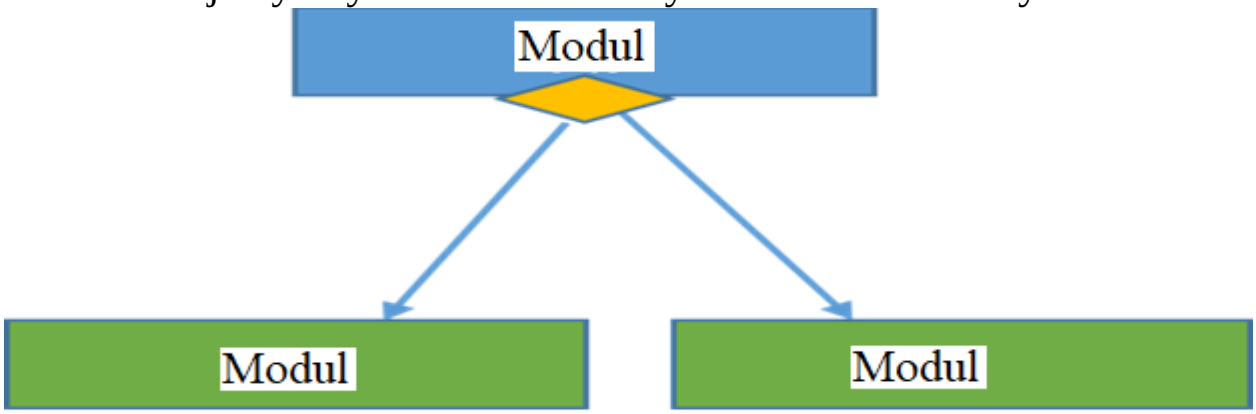
Tuzilmalar diagrammasi

Tuzilmalar diagrammasi - bu ma’lumotlar oqimining diagrammasidan olingan diagramma. Bu tizimni DFDga qaraganda batafsilroq tanishtiradi. U butun tizimni eng past funktsional modullarga ajratadi, tizimning har bir modulining funktsiyalari va kichik funktsiyalarini DFDga qaraganda batafsilroq tasvirlab beradi. Tuzilmalar diagrammasi modullarning ierarxik tuzilishini ifodalaydi. Har bir qatlamda ma’lum bir vazifa bajariladi.



7.5-rasm. Tuzilmalar diagrammasi[22].

Bu yerda tuzilmalar diagrammasini tuzishda ishlatiladigan belgilar: Modul - bu jarayon yoki kichik dastur yoki vazifani ifodalaydi.



7.5-rasm. Biror shart asosida boshqarishni belgilangan modulga o‘tkazish.

[22] Зубкова Т.М. Технология разработки программного обеспечения. Учебное пособие. — Оренбург: Оренбургский государственный университет, 2017.

Boshqarish bo‘linmasi bir nechta submodullarga bo‘linadi. Fayl modullari har qanday moduldan qayta foydalanish mumkin.

Shart – biror shart asosida boshqarishni belgilangan modulga o‘tkaziladi.

7.2. Dasturiy ta’minot obyektining tizimli tahlili

Har qanday obyektni tizim sifatida qarashimiz mumkin. Obyekt uchun dasturiy ta’minotni ishlab chiqrilishi kerak bo‘lsa, bu obyektni tizim sifatida ifodalash tavsiya etiladi. Dasturiy ta’minotni ishlab chiqishdan oldin tizimni tahlil qilish maqsadga muvofiqdir. Tizimni tahlil qilish jarayoniga e’tibor beradigan bo‘lsak, ba’zi tushunchalarga duch kelamiz. “Tizimli yondashuv”, “tizim nazariyasi”, “tizimli tahlil”, “tizimlilik tamoilli” kabi tushunchalar ko‘plab adabiyotlarda qo‘llaniladi. Bu tushunchalarni bir biridan ajratish qiyin, aksariyat hollarda sinonim kabi talqin qilinadi. Bizning fikrimizcha, tizim yaratilishining barcha imkoniyatlari keng ma’noda “tizimlilik” deyiladi. Ushbu termin ikkita asosiy ma’noni anglatadi:

1) Insonga bog’liq bo‘lmagan aniqlilik xususiyati, tizimlilikning obyektivlilik bilan mosligini tashkil qiladi.
2) Insonlar tomonidan to‘plangan xususiyatni o‘zi haqidagi tushunchalarni anglatadi, ya’ni u o‘zida gnoseologik hodisalarni, turli tabiatli tizimlar to‘g’risidagi bilimlarni ifodalaydi.

Fan sohasidagi ko‘p jihatli va keskin yutuqlar, tizimli dunyoqarash va tizimli tahlilning keng qo‘llanilishi asosida kelib chiqqan. Keyingi yillarda ilmiy texnikaviy inqilob texnik yangilanishlar yaratishda so‘zsiz tizimli yondashuv bo‘lib xizmat qildi.Va nihoyat, ishlab chiqarishning muvaffaqiyatlari ham tizimlashtirildi. Qat’iyat bilan shuni aytish mumkinki, XX asr faqatgina atomni kashf qilish yoki kompyuterni ixtirosi bo‘lib qolmadi. Uning asosiy yutug’i bu tizimli dunyoqarashini yaratilishi, so‘ngra atom energiyasidan oqilona foydalanish, kompyuterning yaratilishi va ta’lim, texnika, ishlab chiqarish, siyosat va madaniyat sohalarida minglab yutuqlarga erishildi. Shu yillarda tizimning umumiy nazariyasi ishlab chiqarila boshlandi. Keyinchalik, tizimli bilimlarning ajratilgan amaliy sohasi – sistemotexnika, ya’ni tizimlar to‘g’risidagi muhandislikka yo‘naltirilgan bilimga aylandi. Tizimli yondashuvning asosiy vazifasi butun tizim amal qilish samaradorligini oshirishdan iboratdir.

Tizimli tahlil – obyektlardagi murakkab kuzatilinadigan va tushuniladigan xususiyatlarni va munosabatlarni tadqiqot etish uslubiyoti bo‘lib, ushbu obyektni ma’lum maqsadga yo‘naltirilgan tizim sifatida tasavvur qilib, uning xususiyatlarini, maqsad hamda uni amalga oshiruvchi vositalar orasidagi bog’lan ishlarni o‘rganuvchi ilmiy yo‘nalishdir. Tizimli tahlil uslubiyotini yanada chuqurroq aniqlab olish uchun u foydalangan g’oyalarni ko‘rib chiqamiz:

1-chi g’oya. Murakkab obyektni o‘rganayotganda asosiy e’tiborni uning ichki qismlarining tuzilishiga emas balki, obyektni boshqa tizimlardagi tashqi aloqalariga ajratish lozim. Misollar bilan aniqlaymiz. Firmada biror bir muammo paydo bo‘ldi, masalan, sotuv hajmining kamayib ketishi, daromadning pasayib ketishi va h.k. Muammoning oddiy echishni yo‘li bu muammoni firmaning ichidan qidirishdan iborat: oldindan belgilangan texnologik operatsiyalar tartibining buzilishi, noto‘g’ri boshqaruv va h.k. Ammo omadsizlik firmaning ichida bo‘lmasligi ham mumkin.

Tizimli yondashuv ushbu firmada yuzaga kelgan muammoning echimini ko‘rib chiqarib beradi. Bu holatda bozorni ko’rib chiqish haqiqatga yaqinroq hisoblanadi, ya’ni, iste’molchilar talabini ko‘rib chiqish, raqobatchi – firmalar holatini o‘rganish va h.k. Balkim, bu muvoffaqiyatsizlikning sababi moliyaviy holatlarning barqaror emasligi, mamlakatning noto‘g’ri moliya siyosati va h.k.lar bo‘lishi mumkin. Bu sharoitda firmada yuzaga kelgan muvaffaqiyatsizlikning sababini ichidan izlash qoniqarli natija bermaydi, yoki oxirigacha doimiy ravishda qayta ko‘rib chiqish va yangilashga to‘g’ri keladigan xususiy qaror qabul qilishga olib keladi.

2-chi g’oya. Murakkab obyektni o‘rganilayotganda ustunlik undan chiqariladigan strukturaning maqsadi va funktsiyalariga beriladi, ya’ni tizimli tahlil bu – funktsional yondashuvdir. Bu g’oyani izoxlaymiz. Hayotda ko‘pincha teskari holat bilan to‘qnashishga to‘g’ri keladi, ya’ni obyektning strukturasi mavjud, u qandaydir funktsiyaga ega, lekin shunga qaramay undan kelib chiqadigan natijani bashorat qilish qiyin. Vazifasi oldindan ma’lum bo‘lgan texnik tizimlar haqida gap borganda, bunday yondashuv jiddiy xatolikka olib kelmaydi.

Inson yoki jamiyatdek murakkab tizimlar bilan ish olib borilganda an’anviy yondashuv katta xatoliklarga olib kelishi mumkin. Gap shundaki, bunday tizimlarning vazifasi oldindan ma’lum emas va bunday noaniqliklar ularni boshqarishda qo‘shimcha qiyinchiliklar tug’diradi. Tizimli tahlil boshqacha yondashuvni taklif qiladi, ya’ni bunda maqsad

(funktsiya) mavjud, unga erishish uchun esa qanday struktura kerakligini aniqlash funktsional yondashuv orqali amalga oshiriladi. Bunday yondashuv funktsiyalar qaytarilishi va ularning takrorlanishini istisno etib, optimal echimlarni ishlab chiqish imkonini beradi.

3-chi g'oya. Tizimlar bilan bog'liq bo'lgan muammoni echishda zarur va bo'lishi mumkin bo'lgan, kutilgan va erisha oladigan, samaradorlik va samaradorlik uchun kerakli bo'lgan resurslarni solishtirish kerak. Boshqacha qilib aytganda doimo talab qilinayotgan natijani olish uchun qanday "narx" to'lash kerakligini nazarda tutish kerak. Bu g'oyani izoxlaymiz. Biz turli maqsadlar qo'yamiz va bundan ko'p narsani kutamiz, lekin mavjud resurslarni, ya'ni fizik, intellektual, moddiy, energetik, moliyaviy axborot, vaqt va boshqalarni oldindan baholay olmasak, u holda biz maqsadlarimizni amalga oshira olmaymiz. Buni esdan chiqarish esa bajarib bo'lmaydigan loyihalarga, qaysiki aniq natijani bermaydigan uzoq muddatli dasturlarga olib keladi.

4-chi g'oya. Tizimlarda qaror qabul qilishda ko'rib chiqilayotgan barcha tizimlar uchun echimning natijalarini hisobga olish kerak. Ushbu g'oyani ko'rib chiqamiz. Amaliyotda quydagicha bo'lishi kuzatiladi: har qanday darajada qaror qabul qilishdan osoni yo'qday ko'rinadi. Bunda quydagicha fikr yuritiladi: agar menga qiziq bolmasa, boshqalarning qiziqishlarini nima uchun bilishim kerak? Biroq bunday qiziqishlarni hisobga olinmagan tizimlarda qarorlarini amalga oshirishda ushbu qarorlarga qarshilik ko'rsatish boshlanadi va oqibatida bajarilmaydi, qaror qabul qilgan uchun natija salbiy bolib chiqadi.

Tizimli yondashuvda turli qiziqishlarni hisobga olish va qarorni ishlab chiqishga boshqa tizimlarni jalb qilish nazarda tutiladi. Jalb qilish natijasida katta tizim uchun eng yaxshi qarorni va tashkil qiluvchi tizimlar uchun eng maqbul mumkin bolgan qarorni qabul qilish mumkin boladi. Bunday yondashuvning samaradorligini quyidagi fakt tasdiqlashi mumkin: tizimli yondashuv boshqa rivojlangan mamlakatlarda bo'lgani kabi keng tarqalgan. Yaponiyada qaror qabul qilishda 90% vaqt taaluqli bo'lganlarning barchasi bilan kelishishiga va 10% uni amalga oshirishga sarflanadi.

Tizimli usul aniq va uslubiy vazifalarni, tizimli nazariya esa - tushuntiruvchi va tizimlashtiruvchi vazifalarni tadqiq qiladi. Shu tariqa, tizimlilik aniq faoliyat instrumenti sifatida, hamma borliqning bilish usullari aniq qurollari sifatida talqin qilinadi. Tizimli nazariya tizimlar haqidagi ma'lumotlarni bilim sifatida to'playdi, tartibga soladi, turli tabiiy tizimlarni tushuntirishda foydalanadi.

7.3. Tizimli tahlilning bosqichlari

Tizimli tahlil quyidagi bosqichlardan tashkil topgan:

- masalani qo‘yilishi;
- muammo tadqiqoti;
- dastlabki muhokama (kelishuv);
- tasdiqlash (tajribaviy tekshirish);
- oxirgi muhokama;
- qabul qilingan muhokamani amalga oshirish.

Tizimli tahlilning ba’zi bir bosqichlarini ko‘rib chiqamiz.

Muammo tadqiqoti. Tizimli tadqiqot muammoning dolzarbligini aniqlashdan boshlanadi. Muammo bu echimni talab qiluvchi holat. Odatda muammo haqiqiy holatdan bashorat qilinadigan holatga og’ishda paydo bo‘ladi. Ko‘pincha muammolar xayolan bo‘lib chiqishi mumkin, shuning uchun qimmat tizimli tadqiqotni o‘tkazishni asoslab berish kerak bo‘ladi. Muammoni tahlil qila turib quyidagilarga diqqatni qaratish zarur:

- muammoni topish;
- muammoni aniq shakllantirish;
- muammo strukturasini tahlil qilish;
- muammoni rivojlanishini tahlili;
- tadqiqot olib borilayotgan muammo bilan bog’liq, hamda echimga ega bo‘lishi uchun inobatga olinishi kerak bo‘lgan omillarni aniqlash;
- muammo echilishini imkoniyatini aniqlash.

Muammoni dolzarbligi aniqlangandan so‘ng tadqiqot maqsadi shakllantiriladi. Shunda tadqiqot natijasida biz nimani olishni xohlaymiz degan savol tug’iladi. Masalan, korxona faoliyatini samaradorligini va daromadini 10% oshirish kerak va h.k. Hamma keyingi harakatlar qo‘yilgan maqsadga qarab aniqlanadi. Buni ko‘rib chiqamiz. Shubxasiz, maqsaddan kelib chiqqan holda bitta obyekt turli usullar bilan tavsiflangan bo‘lishi mumkin. Masalan, ekolog daraxtni biosintez elementidek tavsiflaydi, duradgor esa uni taxtachalarga arralab tashlash mumkinligi nuqtai nazaridan qarab chiqadi.

Agar bir guruh ishtirokchilarni yugurish yoki qandaydir bayram qatnashchilari sifatida qarasak, u holda ularning ko‘rsatkichlari (parametrlari) tubdan farq qiladi. Sport bellashuvlari uchun kuch va chidamlilik xarakteristkasi, bayram uchun esa qo‘shiq aytish, o‘ynash qobiliyati muhim. Bir xil insonlar turli ro‘yxatlarda umuman o‘zlariga o‘xshamagan bo‘lishadi.

Dastlabki muhokama. Tizimli tadqiqotning keyingi qadami so‘zalashuv tilidagi obyekt tavsifi shaklida tashkil topgan. Tizimni aniqlash bosqichma – bosqich amalga oshirilishi kerak:

- ekspert holatini aniqlash;
- tadqiqot obyektini va tashqi muhitni aniqlash;
- elementlarni ajratish va aniqlash.

Avvalo obyektni atrof muhitdan cheklab olish zarur. Bu ijtimoiy, iqtisodiy va siyosiy tizimlarni o‘rganishda muhim.

Har bitta tizim o‘ziga xos xususiyatlarga, tashkil etilishiga, maqsadlariga ega bo‘ladi. Biroq barcha tizimlarga ularning fizik tabiatidan qat’iy nazar muayyan umumiy qonuniyatlar, elementlar orasidagi munosabatlar, umumiy boshqaruv qonunlari xos bo‘ladi. Har qanday tabiatga ega bo‘lgan tizimlarni o‘rganishda, ularni boshqarishnig eng yaxshi usullarini qidirishda umumiy yondashuvlar, maxsus uslubiyotlar, tizimlar tuzilmasi va qaror qabul qilishning namunaviy modellarini qo‘llash mumkin bo‘ladi.

Optimal boshqaruvning qidirishning matematik usullari texnik tizimlarda keng qo‘llaniladi. Bugungi kunda ijtimoiy-texnik tizimlarda ana shunday usullarni rivojlantirish dolzarb hisoblanadi. Tizim haqida fikr yuritar ekanmiz, uning asosiy belgilarini quyidagilarga ajratamiz:

1) iyerarxiklik (joylashuv) belgisi - tizim bu elementlar yig’indisi, ularga alohida o‘zlari ham tizim sifatida qaralishi mumkin, boshlang’ich tizimlar umumiy tizimning bir qismidir, ya’ni tizim, tizim ierarxiyasi qismi sifatida ko‘riladi. Masalan avtomobil, shaxarning transport vositalari qismi sifatida qaralishi mumkin va h.k.

2) yaxlitlilikning funksional belgisi: integrativ xususiyatlarning mavjudligi tizim uchun xarakterlidir, ya’ni tizimda mavjud bo‘lgan, ammo uning alohida elementlaridan hech biriga xos bo‘lmagan yoki ularning yigindisi.

3) Mavjudlilik belgisi: mavjud elementlar orasidagi aloqalar tizim uchun xarakterli.

Yuqorida keltirilgan 3 ta belgi bir biri bilan uzviy bog’langan. Bittasini qiymati qolgan ikkitasini qiymatini o‘ziga jalb qiladi. Tizim ko‘p hollarda qismlar yoki elementlar o‘rtasidagi bog’lanishlarning ayrim majmui sifatida belgilanadi va bunday ta’rif tizimning tuzilmaviy tahlilga keyinchalik o‘tish uchun tadqiqot vazifalarini muayyan shakllantirish imkonini beradi. Bunda vazifalar shartiga muvofiq va emperik bilimlarning dastlabki ma’lumotlariga tayangan holda turli tizimlar sifatida bir xil obyektni ko‘rish mumkin.

Dastlab obyekt xususiyatlarning ayrim tizimi kabi namoyon bo‘ladi, ushbu xususiyatlar obyektning butun namoyon bo‘lishdagi tashqi bog’lanishlarni ifodalaydi. Bu erda elementlarning ichki bog’lanishini nazarda tutuvchi obyekt tuzilmasi noma’lum bo‘lganda ham tizimli ko‘rib chiqiladi. Butun xususiyatlar tizimidan tuzilmaga quyidagi shartda o‘tishi mumkin, agar ushbu xususiyatlar tabiati bilan bog’liq bo‘lgan elementlar va ularning barqaror bog’lanishlari topilgan bo‘lsa, ushbu xususiyatlarni tushuntirish imkonini beradi.

Shunday qilib, tizimli yondashuv erkin gipotetik tuzilishlar imkoniyatini ochadi. Tuzilmaviy tadqiqotlar qat’iy qonuniyatlar doirasida ilmiy bilimlarni o‘z ichiga oladi. Klassik tabiatshunoslikda ilmiy tadqiqotning ushbu ikkita turli tiplariga gipoteza va tamoyilllar usullari muvofiq kelgan. Tuzilma tizimdan tashqarida alohida bo‘lmaganidek, tizim o‘z asosida har doim tuzilmaviy bo‘lib qoladi.

Tizimning tuzilmaviy tahlili tizimning muayyan tarkibini aniqlashdan, qismlarni yoki elementlarni mukammal tadqiqot qilishdan, muayyan bog’lanishlarda ularni bir biridan ajratmagan holda ochilishdan boshlanadi. Ushbu munosabatlar ko‘rib chiqilayotgan tizimni keyingi tahlil qilishda tuzilmaviy bog’lanish sifatida namoyon bo‘ladi.

Element tushunchasi tizim tushunchasiga mos kelmaydi. Tuzilmaviy tahlil qism tushunchasidan element tushunchasiga o‘tadi. Tizimning dastlabki qismini aniqlagan, uning tarkibini tahlil qilgan holda, keyin ushbu tarkibini aniqlashtirgan holda tizim elementlarini izlashga o‘tamiz.

7.4. Tizimli tahlilning tamoyillari

Tizimli tahlil uslubiyoti bir qator tamoyillariga asoslanadi. Tizimli tahlilning tamoyillari ostida murakkab tizimlar bilan ishlash tajribasidan kelib chiqqan ma’lum qoidalar majmuasi tushuniladi. Ko‘p hollarda tizimli tahlil tamoyillari qatoriga quyidagilar kiritiladi: yakuniy maqsad tamoyili, o‘lchov tamoyili, barqarorlik tamoyili, birlik tamoyili, o‘zaro bog’liqlik tamoyili, iyerarxiya tamoyili, funksionallik tamoyili, rivojlanish tamoyili, markazlashmaganlik tamoyili. Bularning mohiyatini ko‘rib chiqamiz.

1. Yakuniy maqsad tamoyili. Yakuniy maqsadning ustivorligini belgilaydi. Usbu tamoyil ma’lum qoidalarga ega: tizimli tahlilni amalga oshirish uchun birinchi navbatda tadqiqot maqsadlarini shakllantirish

kerak. Aks holda noaniq belgilangan maqsadlar noto‘g’ri xulosalarga olib kelishi mumkin.

2. O‘lchovlik tamoyili. Qandaydir tizimning faoliyat yuritishi sifati haqida faqat yuqori darajali tizimga muvofiq xulosa chiqarish mumkin. Boshqacha qilib aytganda, tizimni faoliyat yuritishi samaradorligini aniqlash uchun uni yirikroq tizimning bir qismi sifatida tasavvur etish lozim.

3. Barqarorlik tamoyili. Vaqtga bog’liq bo‘lmagan holda va faqat o‘z tavsiflari bilan aniqlanadigan, turli boshlang’ich sharoitlarda va har xil yo‘llar bilan tizim o‘z yakuniy holatiga erishishi mumkinligini belgilaydi.

4. Birlik tamoyili. Bu tizimga bir vaqtda yaxlitlik holatda va alohida qismlarning majmuasi sifatida qarashdir. Ushbu tamoyil tizimni yaxlitligi haqida tasavvurlar saqlangan holda uning bo‘linuvchanligini ko‘zda tutadi.

5. O‘zaro bog’liqlik tamoyili. Tizimning har bir qismini ko‘rib chiqishda barcha tizim elementlari orasida o‘zaro aloqalarini aniqlash ko‘zda tutiladi.

6. Iyerarxiya tamoyili. Iyerarxiya tamoyili (iyerarxiya - quyi sathdan yuqori darajaga o‘tish) murakkab ko‘p sathli tizimlardagi tuzilmaviy munosabatlar turidir, xarakterlanuvchilarning tartiblanganligi, vertikal bo‘yicha alohida sathlarning orasidagi o‘zaro ta’sirini tashkillashtirilganligi. Ierarxik munosabatlar ko‘plab tuzulmaviy xarakterga ega bo‘lgan tizimlarda mavjud.

7. Funksionallik tamoyili. Bu tuzilmani va funksiyani birgalikda ko‘rib chiqishdir. Bunda funksiyaga tuzilmaga nisbatan ustunlik beriladi. Usbu tamoyil ta’kidlaydiki, har qanday tuzilma tizimni va uning alohida qismlarining funksiyalari bilan yaqindan bo‘g’liqdir.

8. Rivojlanish tamoyili. Bu tizim o‘zgaruvchanligini hisobga olishdir. Tizimni rivojlanishga, kengayishga, bilimlarni to‘plashga qobiliyatini aniqlashdir.

9. Markazlashmaganlik tamoyili. Bu murakkab tizimlarda markazlashgan va markazlashmagan boshqaruvning birlashgani. Bu tamoyilning mohiyati shundan iboratki, qo‘yilgan maqsadga erishish uchun markazlashganlik darajasi minimal bo‘lishi kerak.

Tizimli tahlil bilan yaqindan bog’liq bo‘lgan ayrim nazariy yondashuvlarni ko‘rib chiqamiz.

– Tizimlarning «klassik» nazariyasi. Ushbu nazariya klassik matematikadan foydalanadi va quyidagi maqsadlarga ega: umuman tizimlarga yoki ularning muayyan sinflariga (masalan, berk va ochiq

tizimlarga) qo‘llaniladigan tamoyillarni o‘rnatish; ularning tadqiq qilish va tavsiflash uchun vositalarni ishlab chiqish va ushbu vositalarni muayyan hodisalarga nisbatan qo‘llash.

– Graflar nazariyasi. Ko‘plab tizimli muammolar ularning miqdoriy nisbatlariga emas, balki tizimning tuzilmaviy va topologik xususiyatlariga taalluqlidir.

– Yechimlar nazariyasi. Bu matematik nazariya alternativ imkoniyatlar orasidagi tanlash shartlarini o‘rganadi.

– Navbatlar nazariyasi. Ommaviy so‘rovlar sharoitida xizmat ko‘rsatishni optimallashtirish masalalarini ko‘rib chiqadi.

7-bob bo‘yicha xulosalar

Talablar dasturiy ta’minot mijozlarini ehtiyojlarini aks ettiradi. Dasturiy ta’minotning tahlili va konstruktsiyalash talablar spetsifikatsiyasini qo‘shimchaga aylantirishga yordam beradigan barcha tadbirlarni o‘z ichiga oladi. Talab spetsifikatsiyalari dasturiy ta’minotning barcha funktsional va funktsional bo‘lmagan taxminlarini belgilaydi.

Tizimli tahlil – obyektlardagi murakkab kuzatilinadigan va tushuniladigan xususiyatlarni va munosabatlarni tadqiqot etish uslubiyoti bo‘lib, ushbu obyektni ma’lum maqsadga yo‘naltirilgan tizim sifatida tasavvur qilib, uning xususiyatlarini, maqsad hamda uni amalga oshiruvchi vositalar orasidagi bog’lanishlarni o‘rganuvchi ilmiy yo‘nalishdir. Tizimli usul aniq va uslubiy vazifalarni, tizimli nazariya esa -tushuntiruvchi va tizimlashtiruvchi vazifalarni tadqiq qiladi. Shu tariqa, tizimlilik aniq faoliyat instrumenti sifatida, hamma borliqning bilish usullari aniq qurollari sifatida talqin qilinadi.

Tizimli nazariya tizimlar haqidagi ma’lumotlarni bilim sifatida to‘playdi, tartibga soladi, turli tabiiy tizimlarni tushuntirishda foydalanadi. Tizimli tahlilning tamoyillari ostida murakkab tizimlar bilan ishlash tajribasidan kelib chiqqan ma’lum qoidalar majmuasi tushuniladi. Ko‘p hollarda tizimli tahlil tamoyillari qatoriga quyidagilar kiritiladi: yakuniy maqsad tamoyili, o‘lchov tamoyili, barqarorlik tamoyili, birlik tamoyili, o‘zaro bog’liqlik tamoyili, iyerarxiya tamoyili, funksionallik tamoyili, rivojlanish tamoyili, markazlashmaganlik tamoyili. Tizimli tahlilning tamoyillari ostida murakkab tizimlar bilan ishlash tajribasidan kelib chiqqan ma’lum qoidalar majmuasi tushuniladi. Ko‘p hollarda tizimli tahlil tamoyillari qatoriga quyidagilar kiritiladi: yakuniy maqsad tamoyili, o‘lchov tamoyili, barqarorlik tamoyili, birlik tamoyili, o‘zaro

bog’liqlik tamoyili, iyerarxiya tamoyili, funksionallik tamoyili, rivojlanish tamoyili, markazlashmaganlik tamoyili.

7-bob bo‘yicha nazorat savollari

1. Tizimli yondashuv ostida nima tushuniladi?
2. “Tizimli tahlil” tushunchasining tavsifini bering.
2. Murakkab obyektni o‘rganayotganda asosiy e’tibor nimaga beriladi?
3. Funktsional yondashuv ostida nima tushuniladi?
4. Tizimli tahlil qanday bosqichlardan tashkil topgan?
5. Muammoni tahlil qilishda nimalarga e’tibor berish kerak?
6. Muammoli vaziyatlarni tizimli bayon etish uslubiyoti ostida nima tushiladi?
7. Tizimli tahlil uslubiyoti qanday tamoyillariga asoslanadi?
8. Funksionallik tamoyili deganda nimani tushunasiz ?
9. Rivojlanish tamoyili deganda nimani tushunasiz ?
10. Markazlashmaganlik tamoyili deganda nimani tushunasiz ?

8-BOB. DASTURIY TA’MINOTNI KONSTRUKSIYALASHNI BOSHQARISH

8.1. Konstruktsiyalarni boshqarishning asosiy tushunchalari

Konstruktsiyalarni boshqarish ohirgi vaqtlarda investitsion konstruktsiyalarni amalga oshirishda eng yaxshi rejalshtirishning usuli maqomini olgan. Amerikalik baholanishlarga qaraganda, Konstruktsiyalarni boshqarish usullari konstruktsiyaning maqsadlariga erishishni yuqori sifatini va uni amalga oshirishdagi xarajatlarni 10-15 % gacha qisqartiradi. Konstruktsiyalarni boshqarishning asosisy tushuncha va usullarini ko‘rib chiqamiz.

Konstruktsiya - bu vaqtinchalik faoliyat bo‘lib, unikal(takrorlanmagan) maxsulot yoki xizmat yaratishga qaratilgan bo‘ladi. “Vaqtinchalik” deyishning sababi, har bir konstruktsiyaning boshlanish va tugallanish nuqtasi bor, va bu nuqta maqsadga erishilganda yoki ushbu maqsadga erishib bo‘lmaydi degan to‘xtamga kelganda qo‘yiladi.

“Unikal” deyilganda, yaratilayotgan xizmat yoki maxsulot, boshqa analoglardan ancha farq qilinishi tushuniladi.Konstruktsiyalarga misollar :uyni qurish, yangi qurilmani ishlab chiqish, biznes reinjineringi, dasturiy vositalarni yaratish yoki o‘rnatish, rekalama kompaniyasini yuritish, saylovlarni o‘tkazish. Konstruktsiyaning xizmat yoki maxsulot unikalligi konstruktsiyani amalga oshirish vaqtida uning xarakteristikalarini aniqlshga majbur qiladi.

Konstruktsiyalarni boshqarish – konstruktsiyaga qo‘yilayotgan talablarni qondirish uchun bilim, tajriba va usullardan foydalanish va konstruktsiya qatnashchilarini kutish. Ushbu talab vakutishlarni qondirish uchun maqsad, muddat, xarajat, sifat va konstruktsiyaning boshqa xarakterisitikalarining optimal variantini tanlash kerak.

Konstruktsiyalarni boshqarish aniq mantiq bo‘yicha, qaysiki turli sohadagi bilimlar va konstruktsiyalarni boshqarish usullari jamlashtiradi. Birinchi navbatda konstruktsiyaning bir yoki bir nechta maqsadi bo‘lishi kerak. Maqsad deyilganda biz faqatgina oxirgi natijalarni emas, balki ularga erishish uchun qanday yo‘llardan boorish kerakligi ham tushuniladi. Konstruktsiyaning maqsadlariga erishish turli xil yo‘llar orqali amalga oshirilishi mumkin. Ushbu usllarni taqqoslash uchun, qo‘yilgan maqsadlarga erishish kriteriyalari kerak bo‘ladi. Odatda bu kabi kriteriyalarga muddat vamaqsadlarga erishishda sarflangan xarajatlar

tushuniladi. Shu bilan birga rejalashtirilgan maqsadlar vaturli xil variantlarni baholashda asosiy cheklovlar hisoblanadi.

Konstruktsiyalarni boshqarish uchun tayanch vositalar kerak. Konstruktsiyada natijalarga erishish yo‘llari, ishlarni maqsad, sifat, muddat va narxlariga ta’sir qilish uchun qo‘llanilayotgan texnologiyalardan va u yoki bu ishlarni bajarishda resurslarni belgilashdan foydalansh mumkib. Shunday qilib, qo‘llanilayotgan texnologiyalar va resusrlarni konstruktsiyaning asosiy richaglari sifatida qarasak bo‘ladi. Ushbu asosiylardan tashqari yordamchilari ham mavjud va ular asosiylarni boshqarishda ishlatialadi. Shu kabi yordamchi boshqaruv richaglarga misol uchun, kontraktlar- kerakli muddatlarga kerakli resurslarni chaqirish imkonin beradi. Undan tahqari resurslarni boshqarish uchun ishlarning samarali tashkilligi kerak bo‘ladi. Bu konstruktsiyani boshqarish strukturasiga ham, konstruktsiya a’zolarining axborot almashinuvi tashkillashtirilganligiga ham bog’liq.

Konstruktsiyani boshqarishdagi ishlatiladigan axborot yuz foiz ishonarli bo‘lavermaydi. Kiruvchi ma’lumotlarning aniqmasligi muhim va rejalashtirish jarayondia shartnomalarni tuzishda muhim. Noaniqliklarni analiz va hisoblash risklarni aniqlash bag’ishlangan. Har bir konstruktsiya o‘zining amalga oshirilish jarayonida, konstruktsiyaning hayot sikli deb nomlanuvch har xil bosqichlardan o‘tadi. Konstruktsiyani boshqarishda har xil funksiyalarni amalga oshirishda, loyiani boshqarish jarayonlari deb nomlanuvchi xarakatlar lozim bo‘ladi. Boshqarish jarayonlari 6 ta asosiy guruhga bo‘linadi, boshqarishning turli xil funksiyalarini amalga oshiradi:

– Inisiatsiya jarayoni –Konstruktsiyani bajarishni boshlash qarorini qabul qilish.

– Rejalashtirish jarayoni – Konstruktsiyani muvaffaqiyati uchun maqsad va kriteriyalarni aniqlash va Ularga erishish uchun ishlovchi sxemalarni ishlab chiqish.

– Bajarish jarayoni –Rejani amalga oshirishda insonlarga yo‘nalish berish va resurslari boshqarish.

– Analiz jarayonlari –Rejaning tarkibini o‘rganish va muvaffaqiyat kriteriyalari va qo‘yilgan maqsadlar asosida konstruktsiyani boshqarish va To‘g’irlovchi harakatlarni amalga oshirishda kerakli choralarni qo‘llash.

– Boshqaruv jarayoni –Kerakli bo‘lgan to‘g’irlovchi harakatlarni aniqlash, tasdiqlash va qo‘llash

– Tugallash jarayoni –Konstruktsiyani tugallashni formallashtirish va uni tartiblangan hlda ohiriga yetkazish

Konstruktsiyalarni boshqarishning amaliy usullari yordam beradi:

– Maqsadga muvofiq bo‘lgan investitsiyalarni tashkillashtirish;

– Ishlarni moliyalashtirishning optimal sxemasini ishlab chiqarish;

– Ish rejasini tuzish, ishni bajarish muddatlarini o‘rnatish, resurslardan foydalanish, kerakli xarajatlar rejalashtirish;

– Ish faoliyatni optimal boshqarish va Konstruktsiya ishtrikochilarining o‘zaro bog’liq holda ishlashi;

– Sifatni rejalashtirish va boshqarish;

– Konstruktsiyaviy risklarni analiz qilish va boshqarish

– Shartnomalarni optimal rejalashtirish va boshqairsh

– Konstruktsiyalar arxivini olib borish va ularning amaliy tajribasini analiz qilish va kelajakdagi konstruktsiyalarda ulardan foydalanish

Konstruktsiya rejasi bo‘yicha ishlash aniq natijalarni erishishga kerakli bo‘ladigan faoliyatlardan tashkil topgan. Shunday qilib detalizatsiyaning eng quyi darjasida ish asosiy element hisoblanadi va bu uchun qandaydir vaqt sarflanadi, bu esa boshqa ishlarning boshlanishini to‘xtatib turishi mumkin. Ishning tugallanganligi oxrigi maxsulot olinganligini (natijaga erishilganlikni) bildiradi.Ish bazaviy tushuncha hisoblanadi va konstruktsiyani boshqarish tizimlarida ma’lumotlarni tashkillashtirishning asosi hisoblanadi. Amaliyotda ishning yanada detalli bosqichiga yorliq berishda topshiriq iborasi ishlatiladi. Umumiy ma’noda bu ikki ibora ma’nodosh hisoblanadi. Topshiriq iborasi, rejalashtirishning boshqa spesifik kontekstlaridaboshqacha formal ma’noga ega bo‘ladi.

Mantiqiy bog’liqlik - Ishlar o‘rtasidagi bog’liqlik tabiatini ifodalaydi. Konstruktsiyalardagi ko‘pchilik ishlar “Boshlanish-Tugallash” prinsipiga asoslangan bo‘ladi, bunda keyingi ish faqatgina dastlabkisi tugallangandan so‘ng boshlanadi. Ketma-ketlik a’loqalari tarmoq strukturalarini tashkil etadi. Ishlar o‘rtasidagi bog’liqliklar kompleksini odatda konstruktsiyaning manqitiy strukturasi deb atashi, chunki ishlar ketma-ketligini aniqlab beradi.

Tarmoq diagrammasi (tarmoq, graf, PERT diagrammasi) – Konstruktsiya ishlari va ular orasidagi bog’liqliklarning grafik ko‘rinishi. Konstruktsiyalarnirejalashtirish va boshqarishda tarmoq iborasi ostida ishlarnint to‘liq kompleksi va u bilan bog’liq bo‘lgan konstruktsiyalar bog’lanmasi tushiniladi.

Tarmoq diagrammalari tarmoq modelinichiziqlar bilan bog’langan, ishlar o‘rtasidagi bog’lanishlarni ifodalaydigan mos ishlarningko‘pchilik

cho‘qqilar ko‘rinishida tasvirlaydi. Ushbu graf, tarmoq deb nomlanuvchi yoki ketma-ketliklar diagrammasi hozirgi kunda eng ko‘p tarqal vosita hisoblanadi. Boshqa turdagi tarmoq diagrammalari ham mavjud, hodisa-cho‘qqi tarmog’i deb nomlanadi va u amaliyotda kmaroq qo‘llaniladi. Ushbu yo‘nalishda ish ikki hodisa o‘rtasidagi chiziq ko‘rinishida, qaysiki o‘z navbatida ishning boshlanish va oxirini ifodalaydi. PERT-diagrammalariga ushbu turdagi diagrammalari misol bo‘la oladi. Tarmoq diagrammasi blok-sxema bo‘la olmaydi chunki ushbu qurol ish faoliyatini konstruktsiyalashtirishda ishlatiladi. Blok sxemalardanprinsipial farqi shundaki tarmoq diagrammalari ikki odiy ish o‘rtasidagi mantiqiy bog’liqlikni konstruktsiyalashtiradi. U kirish, jarayonlar va chiqishlarni tasvirlamaydi va takrorlanuvchi sikl yoki aylanishlarga yo‘l qo‘ymaydi.

Tarmoqli konstruktsiyalashtirishning usullari - Konstruktsiyaning davomiyligini minimumgacha qisqartirish uchun maqsad qilingan usullar. Amaliy va bir vaqtning o‘zida mustaqil bo‘lga MKP kritik yo‘li va PERT(Programm Evaluation and Review Technique) rejalarini qayta ko‘rib chiqishva taqqoslash usullariga asoslangan.Birinchi usul 1956-yilda “Dyupon” firmasi zavodlarini modernizatsiya qilishda reja-gradiklarini qurishda tashkil topgan. Ikkinchi usul “Lokhid” korporatsiyasi va “Buz, Allend end Gamilton” konsalting firmasi tomonidan “Polaris” raketa tizimini konstruktsiyalashtirishda asos solingan.

Kritik yo‘l –Tarmoqdagi davomiyligi maksimal bo‘lgan yo‘l kritik yo‘l deyiladi. Ushbu yo‘ldagi ishlar ham kritik deb ataladi. Kritik yo‘lning davomiyligi konstruktsiya ustida ishning umumiy davomiyligini izohlab beradi. Umumiy konstruktsiyani bajarish vaqti kritik yo‘llarda yotuvchi ishlarga ketadigan vaqtni qisqartirish hisobiga kamaytirish mumkin. Shu kabi kiritk yo‘lda youtuvchi ishlarga ketadigan vaqtning ortishi bilan konstruktsiya vaqti ham ortib boradi.

Kritik yo‘l konsepsiyasi menedjerning e’tiborini kritik ishlarga qaratishga yordam beradi. Ammo, kritik yo‘lning asosiy ustivorligi kritik yo‘llarda yotmaydigan ishlar ustidan manipulyatsia qilish imkonini beradi.

Kritik yo‘l usuli. Tarmoqning mantiqiy strukturasi aniqlab bergan ishlarni bajarishni kalendar grafigini hisoblash vaKritik yo‘lida yotuvchi ishlarning davomiyligini taqqoslash imkonini beradi.

Vaqtinchalik rezerv yoki vaqt zahirasi –Ishning eng qisqa vaqt ichida tuggalanashi va uni bajarish uchun eng uzoq ketadigan vaqt o‘rtasidagi farq. Vaqt zahirasini boshqaruv ma’nosi shundan iboratki,

kerak bo‘lganida konstruktsiyaning iqtisodiy, texnologik yoki resusr chegaralarini to‘g’irlash, U menedjerga konstruktsiyani yoki u bilan bog’liq topshiriqlarni ma’lum bir vaqtgacha to‘xtatib turish va shuning bilan konstruktsiyani davomiyligiga ta’sir qilmaslikka imkon beradi. Kritik yo‘lda yotuvchi ishlar nolga teng bo‘lgan vaqt rezerviga teng bo‘ladi.

Gant diagrammasi –Gorizontal chiziqli diagramma, unda vaqt bo‘yicha davom etuvchi kesimlarda (boshlanish va tugallash oraliqlarini) konstruktsiyaning ishlari tasvirlanadi.

Ishlarni taqsimlash ishlari –Konstruktsiyaning ishlarini dekompozitsiya ketma-ketligida ierarxik strukturalash. Ishni taqsimlash strukturasi(ITS)ishlarni tashkillashtirishda, tashkilotdagi ishlarning bajaralish strukturasida umumiy ishlarni taqsimlashni ta’minlab beradi. Quyi darajadagi detalizatsiyada elementlar detalizatsiyasiga mos tarmoqli usul orqali ifodalanadigan faoliyat ishlari belgilab olinadi.ITS yaratuvchiga yordam beruvchi ierarxik tarkibda bo‘ladi:

– Komponenta xujjatlarga asoslangan ishni strukturalashtirish;

– Barcha maqsadlar kompleksiga erishish uchun faoliyat yo‘nalishini ta’minlash;

– Konstruktsiya ishlarini bajarishda tizimni ishlab chiqarish ma’suliyati;

– Konstruktsiya bo‘yicha Axborot almashinuvi ва hisobotli tizimni ishlab chiqish.

Tashkilotning strukturaviy sxemasi. Tashkilotning Struktura Sxemasi (TSS) ITS ga o‘xshagan strukturaga ega. ITSning quyi darajadagi har bir elementiga TSSning bir necha elementlari mos tushishi kerak. Shunday qilib, TSS Murakkab tashkilotlardagi qiyint ishlarni bajarishda ma’sullarni aniqlash vositasi hisoblanadi va tizimning xisobot strukturasini ishlab chiqishda asos hisoblanadi

Resurslar–faoliyatning tarkibiy tuzilmasi, qaysiki o‘zida bajaruvchilar, energiya, ashyolar, qurilmalar va h.z.

Resurslarni belgilash va to‘g’irlash. Resurslarni belgilash va taqqoslash menedjerga tarmoq rejasini analiz qilishga yordam beradi, kritik yo‘l usuli orqali qurilgan bo‘ladi, chunkiKonstruktsiyani boshidan oxirigacha bo‘lgan vaqt davomida qaysidir resurslardan foydalanishga imkon beradi. Resusrlarni belgilash turli xil resurlarning qaysi ishga bo‘lgan extiyojini aniqlashdan iborat bo‘ladi. Resusrlani to‘g’irlash usullari o‘zida cheklangan resurlar holatida dasturiy-rejalashtirilgan

evristik algoritmlarni mujassam etadi. Bu usullar menedjerga Konstruktsiyaning haqiqiy ishlar ketma-ketligini tuzishga imkon beradi.

Resusrlar gistogrammasi – Har qanday vaqtd vaqtida konstruktsiyaning u yoki bu resurs ko‘rinishidagi extiyoj gistogrammasi.

Resursli kalendar rejalashtirish – Chegaralangan resurslar holatida ishlarning boshlanish vaqtlarini rejalashtirish. Kalendar rejasida resursli tekshirishni amalga oshirish, konstruktsiyada resurslarning umumiy extiyojiga bo‘lgan funksiyalarni tuzishdan iborat bo‘ladi. Kritik bo‘lmaga ishlarni ularning eng kamida qachon boshlanish(tugallash) vaqtiagacha surish orqali resurslar profil ko‘rinishini o‘zgartirish mumkin va shu bilan birga resusrladan optimal foydalanishni tashkillashtirish mumkin. Konstruktsiyaning resurslarini analiz qilish orqali olingan axborot, menedjer va jamoa ishtirokchilarining e’tiborini resurslarni boshqarish omad kaliti bo‘lgan ishlarga qaratishga yordam beradi.

Konstruktsiyaning amalga oshirilishini analiz qilish – amalga oshirilish tushunchasi turli xil ma’nolarga ega bo‘lishi mumkin: mantiqiy amalga oshirilish, vaqtinchalik analiz, fizik amalga oshirilish, moliyaviy amalga oshirilish.

Oxirgi reja – O‘zida ishningasosiy vaqtinchalik va narxiy parametrlarini muassam etgan ma’lumotlarni tashkil etadi va qaysiki bajarishga topshirilgan. Oxirga rejada odatda ishning hajmi, konstruktsiya topshiriqlarning boshlanish va tugallanish vaqtlari, topshiriq davomiyligi, topshiriq narxini baholash.

8.2. Konsruktsiyalarni boshqarish usullari

Konsruktsiyani boshqarishdagi ishlatiladigan axborot yuz foiz ishonarli bo‘lavermaydi. Kiruvchi ma’lumotlarning aniqmasligi muhim va rejalashtirish jarayondia shartnomalarni tuzishda muhim. Noaniqliklarni analiz va hisoblash risklarni aniqlash bag’ishlangan. Har bir konsruktsiya o‘zining amalga oshirilish jarayonida, konsruktsiyaning hayot sikli deb nomlanuvch har xil bosqichlardan o‘tadi. Konsruktsiyani boshqarishda har xil funksiyalarni amalga oshirishda, loyiani boshqarish jarayonlari deb nomlanuvchi xarakatlar lozim bo‘ladi. Boshqarish jarayonlari 6 ta asosiy guruhga bo‘linadi, boshqarishning turli xil funksiyalarini amalga oshiradi:

– Inisiatsiya jarayoni –Konsruktsiyani bajarishni boshlash qarorini qabul qilish.

– Rejalashtirish jarayoni – Konsruktsiyani muvaffaqiyati uchun

maqsad va kriteriyalarni aniqlash va Ularga erishish uchun ishlovchi sxemalarni ishlab chiqish.

– Bajarish jarayoni –Rejani amalga oshirishda insonlarga yo‘nalish berish va resurslari boshqarish.

– Analiz jarayonlari –Rejaning tarkibini o‘rganish va muvaffaqiyat kriteriyalari va qo‘yilgan maqsadlar asosida konsruktsiyani boshqarish va To‘g’irlovchi harakatlarni amalga oshirishda kerakli choralarni qo‘llash.

– Boshqaruv jarayoni –Kerakli bo‘lgan to‘g’irlovchi harakatlarni aniqlash, tasdiqlash va qo‘llash.

– Tugallash jarayoni –Konsruktsiyani tugallashni formallashtirish va uni tartiblangan hlda ohiriga yetkazish.

Konsruktsiyalarni boshqarishning amaliy usullari yordam beradi:

– Maqsadga muvofiq bo‘lgan investitsiyalarni tashkillashtirish;

– Ishlarni moliyalashtirishning optimal sxemasini ishlab chiqarish;

– Ish rejasini tuzish, ishni bajarish muddatlarini o‘rnatish, resurslardan foydalanish, kerakli xarajatlarni rejalashtirish;

– Ish faoliyatni optimal boshqarish va konsruktsiya ishtrikochilarining o‘zaro bog’liq holda ishlashi.

– Sifatni rejalashtirish va boshqarish.

– Konsruktsiyaviy risklarni analiz qilish va boshqarish.

– Shartnomalarni optimal rejalashtirish va boshqairsh.

– Konsruktsiyalar arxivini olib borish va ularning amaliy tajribasini analiz qilish va kelajakdagi konsruktsiyalarda ulardan foydalanish.

Endilikda konsruktsiyalarni boshqarish jarayonlari, konsruktsiyalarni bajarishda narxli analiz usullari, konsruktsiya risklari analizi usullari, konsruktsiyalarni bohshqarishni dasturiy usullari, konsruktsiyalarni boshqarishni tashkillashtirish. Konsruktsiya rejasi bo‘yicha ishlash aniq natijalarni erishishga kerakli bo‘ladigan faoliyatlardan tashkil topgan. Shunday qilib detalizatsiyaning eng quyi darjasida ish asosiy element hisoblanadi va bu uchun qandaydir vaqt sarflanadi, bu esa boshqa ishlarning boshlanishini to‘xtatib turishi mumkin. Ishning tugallanganligi oxrigi maxsulot olinganligini (natijaga erishilganlikni) bildiradi. Ish bazaviy tushuncha hisoblanadi va konsruktsiyani boshqarish tizimlarida ma’lumotlarni tashkillashtirishning asosi hisoblanadi. Amaliyotda ishning yanada detalli bosqichiga yorliq berishda topshiriq iborasi ishlatiladi. Umumiy ma’noda bu ikki ibora ma’nodosh hisoblanadi. Topshiriq iborasi, rejalashtirishning boshqa spesifik kontekstlarida boshqacha formal ma’noga ega bo‘ladi.

Mantiqiy bog’liqlik - Ishlar o‘rtasidagi bog’liqlik tabiatini ifodalaydi. Konsruktsiyalardagi ko‘pchilik ishlar “Boshlanish-Tugallash” prinsipiga asoslangan bo‘ladi, bunda keyingi ish faqatgina dastlabkisi tugallangandan so‘ng boshlanadi. Ketma-ketlik a’loqalari tarmoq strukturalarini tashkil etadi. Ishlar o‘rtasidagi bog’liqliklar kompleksini odatda konsruktsiyaning manqitiy strukturasi deb atashi, chunki ishlar ketma-ketligini aniqlab beradi. Tarmoq diagrammasi (tarmoq, graf, PERT diagrammasi) – konsruktsiya ishlari va ular orasidagi bog’liqliklarning grafik ko‘rinishi. Konsruktsiyalarni rejalashtirish va boshqarishda tarmoq iborasi ostida ishlarnint to‘liq kompleksi va u bilan bog’liq bo‘lgan konsruktsiyalar bog’lanmasi tushiniladi. Tarmoq diagrammalari tarmoq modelinichiziqlar bilan bog’langan, ishlar o‘rtasidagi bog’lanishlarni ifodalaydigan mos ishlarningko‘pchilik cho‘qqilar ko‘rinishida tasvirlaydi. Ushbu graf, tarmoq deb nomlanuvchi yoki ketma-ketliklar diagrammasi hozirgi kunda eng ko‘p tarqal vosita hisoblanadi. Boshqa turdagi tarmoq diagrammalari ham mavjud, hodisa-cho‘qqi tarmog’i deb nomlanadi va u amaliyotda kmaroq qo‘llaniladi. Ushbu yo‘nalishda ish ikki hodisa o‘rtasidagi chiziq ko‘rinishida, qaysiki o‘z navbatida ishning boshlanish va oxirini ifodalaydi. PERT-diagrammalariga ushbu turdagi diagrammalari misol bo‘la oladi.

Tarmoq diagrammasi blok-sxema bo‘la olmaydi chunki ushbu qurol ish faoliyatini konsruktsiyalashtirishda ishlatiladi. Blok sxemalardanprinsipial farqi shundaki tarmoq diagrammalari ikki odiy ish o‘rtasidagi mantiqiy bog’liqlikni konsruktsiyalashtiradi. U kirish, jarayonlar va chiqishlarni tasvirlamaydi va takrorlanuvchi sikl yoki aylanishlarga yo‘l qo‘ymaydi. Tarmoqli konsruktsiyalashtirishning usullari-konsruktsiyaning davomiyligini minimumgacha qisqartirish uchun maqsad qilingan usullar. Amaliy va bir vaqtning o‘zida mustaqil bo‘lga MKP kritik yo‘li va PERT (Programm Evaluation and Review Technique) rejalarini qayta ko‘rib chiqishva taqqoslash usullariga asoslangan.Birinchi usul 1956-yilda “Dyupon” firmasi zavodlarini modernizatsiya qilishda reja-gradiklarini qurishda tashkil topgan. Ikkinchi usul “Lokhid” korporatsiyasi va “Buz,Allend end Gamilton” konsalting firmasi tomonidan “Polaris” raketa tizimini konsruktsiyalashtirishda asos solingan.

Kritik yo‘l –Tarmoqdagi davomiyligi maksimal bo‘lgan yo‘l kritik yo‘l deyiladi.Ushbu yo‘ldagi ishlar ham kritik deb ataladi. Kritik yo‘lning davomiyligi konsruktsiya ustida ishning umumiy davomiyligini izohlab beradi. Umumiy konsruktsiyani bajarish vaqti kritik yo‘llarda yotuvchi

ishlarga ketadigan vaqtni qisqartirish hisobiga kamaytirish mumkin. Shu kabi kiritk yo‘lda youtuvchi ishlarga ketadigan vaqtning ortishi bilan konsruktsiya vaqti ham ortib boradi. Kritik yo‘l konsepsiyasi menedjerning e’tiborini kritik ishlarga qaratishga yordam beradi. Ammo, kritik yo‘lning asosiy ustivorligi kritik yo‘llarda yotmaydigan ishlar ustidan manipulyatsia qilish imkonini beradi. Kritik yo‘l usuli tarmoqning mantiqiy strukturasi aniqlab bergan ishlarni bajarishni kalendar grafigini hisoblash va kritik yo‘lida yotuvchi ishlarning davomiyligini taqqoslash imkonini beradi.

Konsruktsiyani amalga oshirish

Vaqtinchalik rezerv yoki vaqt zahirasi –Ishning eng qisqa vaqt ichida tuggalanashi va uni bajarish uchun eng uzoq ketadigan vaqt o‘rtasidagi farq. Vaqt zahirasini boshqaruv ma’nosi shundan iboratki, kerak bo‘lganida konsruktsiyaning iqtisodiy, texnologik yoki resusr chegaralarini to‘g’irlash, U menedjerga konsruktsiyani yoki u bilan bog’liq topshiriqlarni ma’lum bir vaqtgacha to‘xtatib turish va shuning bilan konsruktsiyani davomiyligiga ta’sir qilmaslikka imkon beradi.Kritik yo‘lda yotuvchi ishlar nolga teng bo‘lgan vaqt rezerviga teng bo‘ladi.

Gant diagrammasi – Gorizontal chiziqli diagramma, unda vaqt bo‘yicha davom etuvchi kesimlarda (boshlanish va tugallash oraliqlarini) konsruktsiyaning ishlari tasvirlanadi. Vazifalarni taqsimlash ishlari – konsruktsiyaning ishlarini dekompozitsiya ketma-ketligida ierarxik strukturalash. Ishni taqsimlash strukturasi (ITS) ishlarni tashkillashtirishda, tashkilotdagi ishlarning bajaralish strukturasida umumiy ishlarni taqsimlashni ta’minlab beradi. Quyi darajadagi detalizatsiyada elementlar detalizatsiyasiga mos tarmoqli usul orqali ifodalanadigan faoliyat ishlari belgilab olinadi.ITS yaratuvchiga yordam beruvchi ierarxik tarkibda bo‘ladi:

– Komponenta xa xujjatlarga asoslangan ishni strukturalashtirish;

– Barcha maqsadlar kompleksiga erishish uchun faoliyat yo‘nalishini ta’minlash;

– Konsruktsiya ishlarini bajarishda tizimni ishlab chiqarish ma’suliyati;

– Konsruktsiya bo‘yicha Axborot almashinuvi va hisobotli tizimni ishlab chiqish.

Tashkilotning strukturaviy sxemasi. Tashkilotning Struktura Sxemasi (TSS) ITSga o‘xshagan strukturaga ega. ITSning quyi darajadagi har bir elementiga TSSning bir necha elementlari mos tushishi kerak. Shunday qilib, TSS Murakkab tashkilotlardagi qiyin ishlarni bajarishda

ma’sullarni aniqlash vositasi hisoblanadi va tizimning xisobot strukturasini ishlab chiqishda asos hisoblanadi. Resurslar–faoliyatning tarkibiy tuzilmasi, qaysiki o‘zida bajaruvchilar, energiya, ashyolar, qurilmalar. Resurslarni belgilash va taqqoslash menedjerga tarmoq rejasini analiz qilishga yordam beradi, kritik yo‘l usuli orqali qurilgan bo‘ladi, chunkiKonsruktsiyani boshidan oxirigacha bo‘lgan vaqt davomida qaysidir resurslardan foydalanishga imkon beradi. Resusrlarni belgilash turli xil resurlarning qaysi ishga bo‘lgan extiyojini aniqlashdan iborat bo‘ladi. Resusrlani to‘g’irlash usullari o‘zida cheklangan resurlar holatida dasturiy-rejalashtirilgan evristik algoritmlarni mujassam etadi. Bu usullar menedjerga konsruktsiyaning haqiqiy ishlar ketma-ketligini tuzishga imkon beradi. Resusrlar gistogrammasi – Har qanday vaqtda konsruktsiyaning u yoki bu resurs ko‘rinishidagi extiyoj gistogrammasi.

Resursli kalendar rejalashtirish – Chegaralangan resurslar holatida ishlarning boshlanish vaqtlarini rejalashtirish. Kalendar rejasida resursli tekshirishni amalga oshirish, konsruktsiyada resurslarning umumiy extiyojiga bo‘lgan funksiyalarni tuzishdan iborat bo‘ladi. Kritik bo‘lmaga ishlarni ularning eng kamida qachon boshlanish(tugallash) vaqtiagacha surish orqali resurslar profil ko‘rinishini o‘zgartirish mumkin va shu bilan birga resusrladan optimal foydalanishni tashkillashtirish mumkin.

Konsruktsiyaning resurslarini analiz qilish orqali olingan axborot, menedjer va va jamoa ishtirokchilarining e’tiborini resurslarni boshqarish omad kaliti bo‘lgan ishlarga qaratishga yordam beradi. Konsruktsiyaning amalga oshirilishini analiz qilish – amalga oshirilish tushunchasi turli xil ma’nolarga ega bo‘lishi mumkin: mantiqiy amalga oshirilish, vaqtinchalik analiz, fizik amalga oshirilish, moliyaviy amalga oshirilish. Oxirgi reja – O‘zida ishning asosiy vaqtinchalik va narxiy parametrlarini muassam etgan ma’lumotlarni tashkil etadi va qaysiki bajarishga topshirilgan. Oxirga rejada odatda ishning hajmi, konsruktsiya topshiriqlarning boshlanish va tugallanish vaqtlari, topshiriq davomiyligi, topshiriq narxini baholash.

8.3. Dasturiy ta’minotni konstruksiyalashning strategiyalari

Katta konstruksiyalash jarayonlarida alohida tizimosti tizimilar uchun bir qancha ishlab chiqaruvchi guruhlar ma’sul bo‘lishi mumkin. Qoida bo‘yicha, konstruksiyalash guruhi yetkachisi vazifasini bajaradi va o‘zining tizim osti qismini boshqaradi yoki ishlar paketini amalga

oshiradi.

Konstruksiyalashda testlash guruhi quyidagi rollardan tashkil topgan:

– Testlarni loyihalashtiruvchi. Testlash stsenariylarini ishlab chiqish
– Avtomatlashtirilgan tizimlarni ishlab chiqaruvchi
– Testlovchi. Maxsulotni testlash. Natijalarni analiz qilish va xujjatlashtirish.

Ta’minot guruhi a’zolari, qoida bo‘yicha konstruksiyalash jamoasiga kirmaydi. Ular jarayonli faoliyatlaridan chiqmagan holda ishlarni bajarishadi. Ta’minot guruhiga quyidagi konstruksiyalash rollarini kiritish mumkin:

– Texnik yozuvchi;
– Tarjimon;
– Grafik interfeys Dizayneri;
– O‘quv kurslari ishlab chiqaruvchisi, Trener;
– Sharhlash a’zosi;
– Sotuv va marketing;
– Tizim administratori;
– Konstruktor;
– Konstruksiyalash vositalari mutaxassisi;

Konstruksiyaning kattaligidan kelib chiqqan holda, bitta vazifani bir necha kishi bajarishi mumkin. Misol uchun dasturchi, testlash, texnik yozuvchi. Ba’zi rollarni faqat bir kishi bajarishi kerak. Misol uchun konstruksiyalash rahbari, tizim administratori. Bir kishi bir nechta rollarni bajarishi mumkin. Quyidagi moslikliklar ishlatilishi mumkin:

– Konstruksiyalash boshqaruvchisi + Tizim analitigi (+tizim administrator);
– Tizim arxitektori + dasturchi;
– Tizim analitigi + Testlar konstruktori (+ texnik yozuvchi);
– Tizim analitigi + Foydalanuvchi interfeysining konstruktori;
– Konfiguratsiyalarni boshqarishga ma’sul + yigi’sh va integratsiyaga ma’sul shaxs (+ dasturchi).

Quyidagilarni bir-biri bilan bog’lashga iloji boricha yo‘l qo‘yilmasligi kerak :

– Dasturchi + loyiha rahbari;
– Dasturchi + tizim analitigi;
– Dasturchi + foydalanuvchi interfeysini loyihalashtiruvchi;
– Dasturchi + testlovchi.

Konstruksiyalashning kritik vaqtlarida konstruksiyaning dasturchi-menedjeri kelib chiqqan muammolarni hal qilayotganda, konstruksiyalash komandasi to‘liq jamoa bilan uning orqasida turib ushbu jarayonni kuzatib turishganiga ko‘p marta duch kelganmiz. Bu esa konstruksiyani boshqarishning yomon misoli. Dasturchilar dasturlashni yoqtirishadi va bajarishadi. Keling ular shu bilan shug’ullanishsin. Dasturchilarni o‘zlarining ishlari bilan bog’liq bo‘lmagan yumushlar bilan bezovta qilmaslik kerak. Har bir dasturiy maxsulotni konstruksiyalash jarayonida boshqa ko‘plab ishlar bilan shug’ullanadi: Biznes-analiz, ergonomikani loyihalashtirish, grafik dizayn, foydalanuvchi qism xujjatlarini ishlab chiqish. Bu ishlar uchun boshqa tomondan fikrlash kerak bo‘ladi.

Dasturlarni ishlab chiqishda, ushbu topshiriqlar dasturchilarga topshiriladi, bu ishlarni yoqtirishmaydi va qanday bajarilishini bilishmaydi. Bu esa yaxshilikka olib kelmaydi va qimmatga tushadi. Dasturchi o‘zining dasturini birovning ko‘zlari (foydalanuvchi ko‘zlari) bilan ko‘ra olmaydi. Boshqa hechkim foydalanuvchi interfeysi bilan bog’liq texnologik paradigm bilan ishlashni hohlamaydi –dasturchi tizim qanday tuzilganligi haqida bilimga ega bo‘lish kerak. Bu dasturchi uchun tipik holat bo‘lib unga dasturning ishlashi foydalanuvchi uchun qanday ish bajariyayotganidan muhimliroqdir. Shuning uchun konstruksiyalash ishlariga biznes-analitiklar, iqtisodchilar, dizaynerlar, xujjatlashtiruvchilarni jalb etish kerak. Mexnat taqsimoti va mutaxassislashtirish qo‘lbolacha ishlab chiqarishdan undan samaraliroq bo‘lgan ishlab chiqarishga o‘tishga yordam beradi. Professional dasturchilardan yaxshi testlovchilar kelib chiqishadi.

Maqsadga erishish uchun qaratilgan yo‘nalishlar

Konstruksiyalar aniq maqsadlarga erishishga qaratilgan. Albatta mana shu maqsadlar konstruksiyani harakatlantiruvchi kuchi hisoblanadi, va uning rejalashtirishlari va harakatlari ushbu maqsadlarga erishish uchun ishlab chiqiladi. Konstruksiya odatda bir qancha maqsadlarning jamlanmasi hisoblanadi. Misol uchun Kompyuter Dasturiy ta’minoti bilan bog’liq maxsulot konstruksiyasi maqsadi tadbirkorlik boshqaruvini avtomatlashtirish bo‘lishi mumkin. Oraliq maqsadlarga esa ma’lumotlar bazasini ishlab chiqish, Matematik dasturiy ta’minotni ishlab chiqish, tizimni testlashni misol sifatida keltirish mumkin. Ma’lumotlar bazasini ishlab chiqishda yanada pastroq darajadagi maqsadlra ishlatilishi mumkin – ma’lumotalr bazasining mantiqiy strukturasini ishlab chiqish, SUBD

yordamida ma’lumotlar bazisini amalga oshirish.

Bir-biri bilan bog’liq bo‘lgan amallarni boshqarish

Konstruksiyalar tuzilishidan o‘zi murakkab bo‘ladi. Ular bir-biri bilan bog’liq bir qancha amallar ketma-ketligidan tashkil topgan. Alohida holatlarda ushbu bog’liqliklar aniq ko‘rinib turadi, boshqa holatlarda esa nozik tabiat bo‘ladi. Ba’zi bir oraliq topshiriqlar boshqalari bajarilib bo‘lmaguncha bajarilmaydi; ba’zi topshiriqlar faqatgina parallel amalga oshrilishi mumkin va shu kabilar. Agar turli xil topshiriqlarni bajarishning sinxronligi ta’minlanmasa, butun loyiha xavf ostida qolishi mumkin. Ushbu konstruksiya xarakteristikasi ustidan ozroq bosh qotirilsa, shuni anglash mumkinki konstruksiya – bu tizimdir, bir qancha bir-biri bilan bog’liq bo‘lgan tizimdir, shuningdek u dinamik tizimdir va u alohida boshqaruv usullari asosida tuzilgan bo‘lishi shart.

Vaqt bo‘yicha cheklanishlar

Konstruksiyalar aniq vaqt davomiyligida amalga oshiriladi. Ular vaqtinchalikdir. Ularning aniq va unchalik aniq bo‘lmagan boshlanish va tugash nuqtalari mavjud. Konstruksiya uning asosiy maqsadlariga erishilgandan so‘ng tugatilishi mumkin. Konstruksiya bilan ishlashda asosiy kuchlar konstruksiyani vaqtida tugatishga qaratilgan bo‘ladi. Buning uchun topshiriqlarning boshlanish va tugash vaqtlarini ifodalovchi grafiklardan foydalnalidai. Konstruksiya faoliyat tizimi kabi oxirgi natijani olish uchun qancha vaqt kerak bo‘lsa o‘shancha vaqt davomida umr ko‘radi. Loyiha konsepsiyasi, firma yoki tadbirkorlik konsepsiyalariga qarshi bo‘lmasdan balki ular bilan mos keladi. Va aksincha loyiha odatda firmaning asosiy faoliyat turiga aylanadi.

Unikallik

Konstruksiyalash –bir martali va takrorlanmas tadbirlardir. Shu bilan birga, bir konstruksiyaning boshqa konstruksiyadan unikallik darajasi katta farq qilishi mumkin. Agar siz kottedjlar qurilishi bilan shug’ullanib ayni vaqtda 20-chisini qurayotgan bo‘lsangiz, unda sizning konstruksiya unikalligi unchalik katta emas. Ushbu uyning bazaviy elementlari undan oldinroq qurgan 19 siniki bilan o‘xshashdir. Unikallikning asosiy manbalari konkret spesifik ishlab chiqarishlarda ko‘rish mumkin – uyning joylashgan joyida va atrof muhit landshaftida, materiallarning olinish joylari.

Boshqa tomondan, agar unikal qurilma yoki texnologiyani ishlab chiqarayotgan bo‘lsangiz, siz unikal bo‘lgan topshiriq bilan to‘qnash kelgansiz. Siz oldin qilmagan ishlaringizni qilasiz. Va uzoq vaqtli tajribangiz sizga ushbu holatda imkoniyatlarni beradi, konstruksiyalashni bajarishda nimalarni kutsa bo‘ladi, u tavakkalchilik va aniqmasliklarga to‘la.

Konstruksiyalashni boshqarish

Lermantning mashhur qonunida aytilishicha: “Hohlagan texnik muammoni, yetarlicha pul va vaqt bilan hal qilish mumkin”, Lermant kuzatishi bo‘yicha: “Sizga hech qachon yoki pul yoki vaqt yetishmaydi”. Lerman kuzatishi natijalari bo‘yicha kelib chiqqan muammolarni hal qilishda loyiha usuli kelib chiqqan. Ushbu metodikaning turli hil faoliyat sohalarida ishlatilishi uning samaradorliligidan habar beradi. Agar siz menedjerdan uning loyihasining asosiy vazifasi nimadan iborat ekanligi so‘rasangiz u: “Ishlarning bajarilishi taminlash” deb javob beradi. Agar tajribaliroq menedjerga ushbu savolni beradigan bo‘lsak, biz ya’nada to‘liqroq javob olishimiz mumkin: “Ishlarning texnik topshiriq asosida va vaqtida bajarilishini ta’minlash” deb javob beradi. Asosan ushbu uch holat: vaqt, byudjet va ish sifati loyiha va konstruksiyalash boshqaruvchisining doimiy nazoratida bo‘ladi. Ularni shuningdek loyiha va konstruksiyalashga yuklatiladigan asosiy cheklovlar deb tariflash mumkin. Loyiha va konstruksiyalash boshqaruvi deb, maksimal imkoniyatlardan foydalaib, cheklangan mablag’lar va vaqt ichida hamda shuningdek, oxirgi natijalarning sifati e’tiborga olinib boshqarish tushuniladi.

O‘ttiz yildan ortiqdirki, konstruksiyalashni boshqarish texnologiyasini qo‘llashda, konstruksiyalar boshqaruvchilariga yordam sifatida bir qancha usul va vositalar ishlab chiqilgan. Uch asosiy cheklovlardan eng qiyiti berilgan loyiha natijalarini nazorat qilishdir. Muammo shundan iboratki, topshiriqni nazorat qilish va shakllantirish juda qiyin. Ushbu kabi muammolarni yechishda ishlarning sifatini boshqarish usulidan foydalaniladi.

Loyiha va konstruksiyalashni boshqarish usullari tarmoqni rejalashtirishda, AQSHda 50-yillarning oxirida ishlab chiqilgan.1956-yilda “Dyupon” firmasidan Uniyac mashinasining hisoblash imkoniyatlarini o‘rganayotib, u “Remington Rend” firmasi kapital qurilishni rejalashtirish bo‘limidan D.Kelli bilan birlashadi. Ular “Dyupon” firmasi zavodlarini rivojlantirish ishlari grafiklarini EHM vositasida amalga oshirishmoqchi bo‘lgan. Avval u Uolker-Kenni usuli

deb nom oldi va keyinchalik, u Kritik Yo‘l Metodi - KYP deb nom oldi. Parallel va mustqali ravishda AQSH ning dengiz kuchlarida PERT dasturlarni baholash va analiz qilish tizimi ishlab chiqildi. Ushbu usul “Loxid” korporatsiyasi va “Buz Aleend end Gemilton” konsalting firmalari tomonidan “Polaris” raketa tizimini ishlab chiqish loyihasida ishlatilgan, qaysiki 3800 ta asosiy podryadchik va 60ming dan ortiq amallardan tashkil topgan. PERT metodining qo‘llanilishi dastur rahbariyatiga, aniq vaqt paytida nima ish qilish kerak, va uni kim qilish kerak va shuningdek alohida jaryonlarning vaqtida bajarilishini bashorat qilishga yordam berdi. Dastur yo‘riqnomasi shunchalik omadli bo‘ldiki, loyihani rejalashtirilganidan 2 yil oldin tugatishga muvaffaqiyat bo‘lishdi. Yirik ishlab chiqaruvchi korporatsiyalar shu kabi boshqaruv metodlarini ishlab chiqarishni modernizatsiyalash va yangi maxsulotlarni ishlab chiqrishda harbiylar bilan bir vaqtda qo‘llashdi.

Rejalashtirish metodikasini qo‘llash keng qamrovda qurilishda ishlatildi. Misol uchun, Nyufaullenddagi Cherchil daryosidagi gidroelektr stansiyasi qurilishi. Loyihaning qurilishi 950 mln dollarni tashkil etdi. Gidro elektrostansiyasi 1967-yildan 1976-yigacha qurilgan Ushbu loyiha 100 dan ortiq qurilish kontraktlarini muajassam etgan, shuni takidlash kerakki ularning ba’zilari 76mln dollarni tashkil etgan. 1974-yilda loyiha rejadan 18 oy oldinlab ketgan va xarajatlar chizig’idan chiqib ketmagan. Loyiha buyurtmachisi Cherchill Falls Labrador Corp bo‘lib, loyihani ishlab chiqish va boshqarishga Acress Canadian Betchel firmasini yollagan.

Birinchi bo‘lib yirik kompaniyalar dasturiy ta’minotni o‘zlarining loyihalirini qo‘llab-quvvatlash uchun ishlab chiqarishgan, ammo keyinchalik loyihalarni boshqarish tizimlari dasturiy ta’minotlar bozorida paydo bo‘ldi. Rejalashtirish asosidagi tizimlar Katta kuchli kompyuterlar va mini-EHM tarmoqlari uchun yartilgan.

Ushbu klassdagi asosiy ko‘rsatkichlar yuqori miqdordagi kuchlanganlik va murakkab tarmoq rejalashtirishlar konstruksiyalarni yetarli detallarda tasvirlay olishlik qobiliyati bo‘yicha baholangan. Ushbu tizimlar Tarmoq rejalashtirish bilan yaxshi tanish bo‘lgan va spesifik terminlarni yaxshi tushungan, katta loyihalarni amalga oshira oladigan juda professional menedjerlar uchun mo‘ljallangan. Qoida bo‘yicha, konstruksiyalarni ishlab chiqish va konstruksiyalashni boshqarish bo‘yicha konsultatisyalar konsalting firmalari tomonidan amalga oshiriladi. Ayni vaqtda AQSHda konstruksiyalarni boshqarish tizimlar hayot faoliyatining ko‘p qismida ishlatish odati paydo bo‘lgan.

Shuningdek rejalashtiriladigan konstruksiyalarning juda ko‘pchiligi unchalik kata bo‘lmagan konstruksiyalardan tashkil topgan. Har kunlik InfoWorld tomonidan o‘rganilib chiqilgan ma’lumotlarga ko‘ra, aholining ellik foizi 500-1000 dan tashkil topgan rejalarni qo‘llab quvvatlaydigan tizimlar va faqatgina 20% foydalanuvchilar ishlari soni 1000 tadan ortiqligini o‘rganib chiqishgan. Resurslarga keladigan bo‘lsak, 38% foydalanuvchilar 50-100 turdagi resurslarni loyiha chegarasidaboshqarish imkoniyati bo‘lar ekan, faqatgina 28% foydalanuvchilargina 100 dan ortiq resurslarni boshqarish talab etiladi. Tabiiyki, konstruksiya menedjmenti foydalanuvchilari ortib borishi bilan, boshqarish usullarining turlari ham ko‘payib boradi. G’arb kompyuter jurnallari konstruksiyalarni boshqarish tizimlarini boshqarish haqida maqolarlni doimiy chop etishmoqda, va shuningdek Boshqaruv yo‘nalishlarda turli xil rejalshtirish muammolarini hal qilish va tarmoq rejalashtirishni analiz qilish bo‘yicha maqolalar chop etishmoqda.

Konstruksiyaning hayot davri

Har bir konstruksiya o‘zining rivojlanishida aniq bir bosqichlardan o‘tadi. Konstruksiyaning hayot sikli faoliyat turi va ishlarni tashkilashtirish usullariga bog’liq. Ammo, har bir loyihada boshlang’ich bosqichini, amalga oshirish bosqichi va konstruksiyalash bo‘yicha ishlarni tugatish bosqichlarini ajratib ko‘rsatsa bo‘ladi. Bu ko‘rinib turibdiki, konstruksiyaning hayot sikli tushunchasi menedjer uchun eng muhim tushunchalardan biridir, chunki joriy bosqich menedjerning faoliyati, foydalanalidagian usullar va vositaviy konstruksiyalarni aniqlab beradi.

Konstruktsiya boshqaruvchilari konstruksiyaning yashash siklini etaplarga turli xil bo‘lishadi. Misol uchun, dasturiy ta’minot ishlab chiqarish konstruktsiyalarida, information tizimga, talablarni tashkillashtirish, tizimni konstruktsiyalash, kodlash, testlash, espluatatsion qo‘llab-quvvatlash kabi etaplarga bo‘lishadi. Ammo eng ommaviylariga 4 etapga bo‘lish hisoblanadi: Konstruktsiya formulirovkasi, rejalashtirish, amalga oshirish va tugallash.

Konstruktsiya formulirovkasi tushunchasi ostida konstruktsiyani tanlash dunksiyasi yotadi. Konstruktsiyalar ehtiyojlarning kelib chiqishini qondirishga qartilgan maqsadlar orqali kelib chiqadi. Ammo, kamchilik sharoitlarida barcha extiyojlarni qondirib bo‘lmaydi. Shunda tanlashga to‘g’ri keladi. Ba’zi bir konstruktsiyalar tanlanadi, ba’zilari esa yo‘q. Qarorlar resurslardan kelib chiqib qabul qilinadi, birinchi navbatda

iqtisodiy imkoniyatlar nazarga olinadi, konstruktsiyalarning samaradorligi taqqosalanadi. Amalga oshirishda konstruktsiyalarni tanalash muhimki, qanchaki konstruktsiya yirik bo‘lsa, chunki yirik konstruktsiyalar yo‘nalishi kelajakka qartilgan bo‘ladi (ba’zida yillar) va joriy mavjud iqtisodiy mexnat imkoniyatlari nazarda turiladi.

Asosiy ko‘rsatkichlar sifatida bu yerda investitsiyalarning qiymati belgilab beriladi. Boshqacha qilib aytganda “A” konstruktsiya tanlab turib “B” konstruktsiyani tanlamasdan tashkilot “B” konstruktsiya keltirishi mumkin bo‘lgan foydadan voz kechadi. Konstruktsiyalarni nisbatan taqqoslashda ushbu bosqichda konstruktsiya analizi usullari qaysiki o‘zida iqtisodiy, moliyaviy, kommersiali, tashkiliy, ekologik va boshqa turdagi risklar analizini mujassam etgan. Konstruktsiyalarni ushbu bosqichda boshqarishni rejalashtirish cheklangan miqdorda ishlatiladi.

Rejalashtirish. Rejalashtish konstruktsiyani amalga oshirish davomida u yoki bu ko‘rinishda amalga oshiriladi. Konstruktsiyaning siklik hayotining boshida odatda ofitsial bo‘lmagan boshlang’ich rejasi – konstruktsiyani amalga oshirish mumkin bo‘lganda kerak bo‘ladigan resurslari belgilab olinadi. Konstruktsiyani tanlashdagi qarorlar sezirarli darajada joriy konstruktsiyani baholashga asoslangan bo‘ladi. Konstruktsiyaning kalit nuqtalari belgilab olinadi, topshiriqlar belgilanadiva ular o‘rtasidagi bog’liqliklar aniqlanadi. Aynan ushbu bosqichda konstruktsiyalarni boshqarish tizimlaridan foydalaniladi: ierarxik ishlar strukturasi resurslari, Ganttning diagrammalari va tarmoq grafiklari, resurslarni joylash gistogrammalari.

Konstruktsiyaning rejasi o‘zgarishsiz qolmaydi, chunki konstruktsiyani amalga oshirish vaqtidagi holatdan kelib chiqqan holda korrektirovka amalga oshiriladi.

Amalga Oshirish. Formal reja tasdiqlanganidan so‘ng, menedjerga uni amalga oshirish ma’suliyati yuklanadi. Konstruktsiyani amalga oshirish vaqtida boshliq-rahbarlar ish jarayonini doimiy boshqarib borishlari kerak. Boshqaruv deganda ish jarayonidagi ma’lumotlarni yig’ib rejadagi bilan solishtirish tushuniladi. Afsuski, konstruktsiyalarni boshqarishda rejalar va faktik ko‘rsatkichlar orasida odatda farq bo‘ladi. Shuning uchun menedjerning vazifasi chetlashishning bajarilgan ishga ta’sirini o‘rganib chiqish va kerakli boshqaruv amallarini bajrishdan iborat. Misol uchun, belgilangan vaqtdan konstruktsiya ortda qolib ketayotgan bo‘lsa, kritik vazifalarga ketadigan resurslarni oshirib rejadagiga yetib olish kerak.

Tugallash. Ertami, kechmi konstruktsiyalar o‘z nihoyasiga yetadi. Konstruktsiya uning oldiga qo‘yilgan maqsadlarga erishilgandan so‘ng tugallanadi. Bazi vaqtlari konstruktsiyaning tugashi to‘satdan va vaqtidan oldin bo‘ladi, ushbu hollarda konstruktsiyani grafikdan oldin tugallash holati kuzatilishi mumkin. Qanday bo‘lishidan qat’iy nazar konstruktsiya tugalanayotganda konstruktsiya rahbari ba’zi bir tadbirlarni amalga oshirishi kerak. Ushbu majburiyatlarning maqsadi konstruktsiyadan kelib chiqadi. Agar konstruktsiyada qurilmalardan foydalanilgan bo‘lsa, ularning inventarizatsiyasini amalga oshirib, foydalanishga topshirish kerak. Erta tugagan konstruktsiyalarda olingan natijalar buyurtmachining yoki kontraktdagi talablarga javob berishini tekshirish kerak.

8-bob bo‘yicha xulosalar

Konstruktsiya - bu vaqtinchalik faoliyat bo‘lib, unikal (takrorlanmagan) maxsulot yoki xizmat yaratishga qaratilga bo‘ladi. Konstruktsiyalarni boshqarish – konstruktsiyaga qo‘yilayotgan talablarni qondirish uchun bilim, tajriba va usullardan foydalanish va konstruktsiya qatnashchilarini kutish. Konstruktsiya rejasi bo‘yicha ishlash aniq natijalarni erishishga kerakli bo‘ladigan faoliyatlardan tashkil topgan.

Konstruktsiyaning amalga oshirilishini analiz qilish – amalga oshirilish tushunchasi turli xil ma’nolarga ega bo‘lishi mumkin: mantiqiy amalga oshirilish, vaqtinchalik analiz, fizik amalga oshirilish, moliyaviy amalga oshirilish. Har bir konsruktsiya o‘zining amalga oshirilish jarayonida, konsruktsiyaning hayot sikli deb nomlanuvch har xil bosqichlardan o‘tadi. Katta konstruksiyalash jarayonlarida alohida tizimosti tizimilar uchun bir qancha ishlab chiqaruvchi guruhlar ma’sul bo‘lishi mumkin. Qoida bo‘yicha, konstruksiyalash guruhi yetkachisi vazifasini bajaradi va o‘zining tizim osti qismini boshqaradi yoki ishlar paketini amalga oshiradi. Konstruksiyalar aniq maqsadlarga erishishga qaratilgan. Albatta mana shu maqsadlar konstruksiyani harakatlantiruvchi kuchi hisoblanadi, va uning rejalashtirishlari va harakatlari ushbu maqsadlarga erishish uchun ishlab chiqiladi.

Konstruksiyalar tuzilishidan o‘zi murakkab bo‘ladi. Ular bir-biri bilan bog’liq bir qancha amallar ketma-ketligidan tashkil topgan. Konstruksiya uning asosiy maqsadlariga erishilgandan so‘ng tugatilishi mumkin. Konstruksiya bilan ishlashda asosiy kuchlar konstruksiyani vaqtida tugatishga qaratilgan bo‘ladi. Har bir konstruksiya o‘zining rivojlanishida aniq bir bosqichlardan o‘tadi. Konstruksiyaning hayot sikli faoliyat turi va ishlarni tashkilashtirish usullariga bog’liq. Ammo, har bir

loyihada boshlang’ich bosqichini, amalga oshirish bosqichi va konstruksiyalash bo‘yicha ishlarni tugatish bosqichlarini ajratib ko‘rsatsa bo‘ladi. Bu ko‘rinib turibdiki, konstruksiyaning hayot sikli tushunchasi menedjer uchun eng muhim tushunchalardan biridir, chunki joriy bosqich menedjerning faoliyati, foydalanalidagian usullar va vositaviy konstruksiyalarni aniqlab beradi.

Konstruktsiyani tanlashdagi qarorlar sezirarli darajada joriy konstruktsiyani baholashga asoslangan bo‘ladi. Konstruktsiyaning kalit nuqtalari belgilab olinadi, topshiriqlar belgilanadiva ular o‘rtasidagi bog’liqliklar aniqlanadi.

8-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minot uchun talablar deganda nimani tushunasiz?
2. Foydalanuvchi talablari ma’nosini aytib bering.
3. Dasturiy ta’minot talablari iborasi nimani anglatadi?
4. Ma’lumotlar diagrammalarining turlarini aytib bering.
5. Axborot oqimining diagrammasi deganda nimani tushunasiz?
6. DFD komponentlarini sanab bering.
7. DFD darajalarini aytib bering.
8. Tuzilmalar diagrammasiga ta’rif bering.
9. Dasturiy ta’minotni ishlab chiqishda tizimli yondashuvning asosiy vazifasi nimalardan iborat.
10. Tizimli tahlil bosqichlarini sanab bering.

9-BOB. DASTURIY TA’MINOTNI MODELLASHTIRISH

9.1. Dasturiy ta’minotni tahlil qilish usullari

Biror bir dasturiy ta’minot konstruksiyalashda, matematik ta’minotni tashkil qiluvchi, matematik modelni yaratish metod va algoritmlarni tanlash uchun bir qancha talablarni e’tiborga olish kerak bo‘ladi. Asosiy talablarga universallik, algoritmik puxtalik, aniqlik, kompyuter vaqtiga bo‘lgan sarflarni chegaralash, xotira hajmiga kam foydalanish kabilar kiradi. Matematik ta’minotning universalligi deyilganda, bu ta’minotdan bir qancha konstruksiyalash obyektlarini konstruksiyalashda foydalanish imkoniyati borligi tunushiladi. Masalan: Dasturiy ta’minotni sintez qilishda foydalaniladigan metod va modellar, berilgan baza elementlarini oldindan ko‘rsatilgan cheklanishlar asosida, hohlagan obyektni yaratishni ta’minlay olishi kerak.

Dasturiy tizimlarda matematik ta’minotning asosiy xususiyatlaridan biri ularning yuqori universallikka ega ekanligi hisoblanadi. Yuqori, universallikka ega bo‘lishi esa bunday dasturiy ta’minotlardan bir qancha konstruksiyalash obyektlari uchun foydalanish imkonini beradi. Matematik ta’minotning komponentalarining muhum xususiyatlaridan biri bu ta’minotdan foydalanilganda olinadigan natijaning to‘g’ri bo‘lishligi hisoblandi va shunday xoldagina bu ta’minot algoritmik puxta deyiladi. Algoritmik puxtalik asosan olingan natija qanchalik to‘g’riligi ehtimolligiga qarab baholanadi. Agar bu ehtimollik birga teng bo‘lsa, yoki unga yaqinlashib borsa, u holda bu ta’minotni yaratish uchun foydalanilgan metod algoritmik puxta metod deyiladi. Dasturiy ta’minotda algoritmik puxta metod bo‘lmagan metodlardan foydalanish maqsadga muvofiq bo‘lmaydi.

Algoritmik puxtalik bilan, sharoitga moslashtirish uzviy bog’langandir. Sharoitga moslashtirilmagan hollarda, berilganlardagi ozgina xatoliklar, natijada katta xatoliklarni yuzaga keltirish mumkin. Konstruksiyalashning har bir bosqichida o‘zining oraliq natijalari va har bir bosqichdagi xatolik kelib chiqish manbaalari mavjud. Shu sababli yomon sharoitga moslashtirish natijasida hatoliklar juda kattalashib ketishi mumkin. Ko‘pgina matematik ta’minot komponentlari uchun aniqlik muhim hossalardan biri hisoblanadi. Aniqlik – bu olgan natija bilan haqiqiy olinishi kerak bo‘lgan natijaning mos tushishiligi hisoblanadi. Algoritmik puxta metodlar har xil aniqlikdagi natijalarni berishi mumkin.

Unisersal model va metodlarda hisoblash ishlari ko‘p bo‘lib, konstruksiyalash masalasining hajmi kattalashib ketishi mumkin, shuning uchun ko‘pgina dasturiy ta’minot proseduralarida T—kompyuter vaqtini, sarf qilish konstruksiyalash harajatlarini va konstruksiyalashga ketadigan vaqtni oshib ketishiga olib keladi. Shuning uchun T ni minimallash konstruksiyalash jarayonidagi muhim talablardan biridir. Yuqori universallikka ega, aniq natijali va kamkompyuter vaqt talab qiluvchi matematik ta’minotni yaratish doimo bir – biriga qarama – qarshi bo‘lib, keladi. Shuning uchun matematik ta’minotning komponentlari ba’zi bir masalalarni hal qilish uchun sifatli bo‘lsa, boshqa masal uchun kam sifatli bo‘lishi mumkin. Shuning uchun dasturiy ta’minotlarda model va metodlarning har xillarini o‘zida mujassamlatirilgan kutubxonalar tashkil qilish kerak.

Kompyuter vaqtini tejash talabidan keyingi talab, xotira hajmini tejash talabidir. Bu talab matematik ta’minotning qanchalik iqtisodlashtirilganligii ko‘rsatadi va tuzilgan dastur uzunligi bilan harakterlanadi. Hozirgi davrdagi operativ xotira hajmini sezilarli darajada katta bo‘lishligiga qaramy, uni tejash aktual masala bo‘lib qolmoqda. Kompyuterlarning multidasturli ishlash rejimida, katta xotira hajmini talab qiluvchi masalalar pastroq ustunlikka ega bo‘lishi va natijada bu masalalarni qayta ishlash vaqti oishb ketishi kuzatiladi. Bunday murakkablikni yechish uchun tashqi xotiralardan foydalanish maqsadga muvofiq bo‘ladi. Bu esa operativ va tashqi xotiralarning o‘zaro ma’lumotlarni almashlashlari uchun, T- vaqtni oshib ketishiga olib keladi. Dasturiy ta’minotni matematik ta’minotlarini rivojlanishiga, foydalanilaniladigan metod va modellarning iqtisodkorligini oshirishga intilish sezilarli ta’sir ko‘rsatadi.

Matematik modellarni hosil qilish metodlarini ikki guruhga ajratish mumkin. Birinchi guruh metodlariga, turli iyerarxik darajalarda elementlar matematik modellarini va tizimlar makromodellarini hosil qilish metodlari kiradi. Bunday metodlarning xarakterli xususiyatlari bo‘lib, ularda formal bo‘lmagan (evristik) usul va proseduralardan foydalanish hisoblanadi. Modelning matematik munosabatlari ko‘rinishi tanlash prosedurasi formallashtirilmagan hisoblansa, keyingi bosqich model parametralri sonli qiymatlarini aniqlash formalashtirilgan bo‘lishi mumkin.

Birinchi guruh matematik modellarni hosil qilishda nazariy va eksperimental metodlar mavjud. Nazariy metodlar modelda tasvirlanadigan jarayonlarni fizik qonuniyatlardan foydalanib yaratishga

asoslangan. Modelni hosil qilish, modellashtirilayotgan obyektni maxsus xususiyatlarini tas’virlovchi qator yengilliklar qabul qilish bilan olib boriladi. Hosil qilinayotgan modellar asosini, yechimi fazoviy o‘zgaruvchilarning o‘zaro bog’liqligi hisoblangan tenglamalar tizimi tashkil qiladi. Bu modellar algoritmik modellar hisoblanib, ular o‘zgaruvchilarning nisbatan keng diapazonlaridan o‘zgarishida ham to‘g’ri hisoblanadi. Eksperimantal metodlar obyekt modelini uning parametrlarining o‘zaro bog’liqligi va obyektninig fazoviy o‘zgaruvchilarini eksperementlar yo‘li bilan hosil qilishga asoslangan. Eksperementlar yaratilgan obyektlarning o‘zida yoki ularning fizik modellarida (maketlar, stendlar) yoki to‘la matematik modellarida o‘tkazilib makromodel hosil qilinadi. Eksperemental ma’lumotlarni matematik modelga o‘zgartirish jarayonida, ularni modelga o‘zgartirish uchun eksperementlarni rejalashtirish metodlaridan foydalaniladi.

Eksperimental metodlar, inersiyaga ega bo‘lmagan, o‘zgaruvchilari o‘rtasida nisbatan tekis bog’liqlikka ega obyektlarni modellashtirishga qulay hisoblanadi. Ko‘pincha bu metodlarni foydalanish natijasi bo‘lib, xususiy xarakterga ega bo‘lgan faktorli modellar hisoblanadi. Element modellarni hosil qilishda formal bo‘lmagan metodlardan foydalnish, ko‘pgina konstruksiyalash proseduralarining avtomatlashtirish darajasi kamaytiradi deb o‘ylash to‘g’ri bo‘lmaydi. Chunki aniq mo‘ljallangan tizimlardagi element turlari soni, elementlar sonidan ancha kam bo‘lib, bu element turlari ko‘pgina konstruksiyalanilayotgantizimlarda takroran foydalaniladi.

Bundan tashqari har bir element turi uchun aniqligi, iqtisodliligi, unversalligi ko‘rsatkichlari bilan farqlanadigan bir qancha modellar yaratish imkoniyati ham tug’iladi. Dasturiy ta’minotlarda hamma unifisirlashtirilgan element tiplari modellari elementlar modellari kutubxonasiga kiritib qo‘yiladi. Dasturiy ta’minotlardan foydalanishda har doim yechish kerak bo‘ladigan masala bo‘lib, tizim modellarini yaratish va anliz qilish hisoblanadi. Ikkinchi guruh metodlariga, elementlarining berilgan modellaridan tizimning to‘la matematik modellini hosil qilish metodlari kiradi. Bu metod bilan tizim modelini hosil qilish metodalir kiradi. Bu metod bilan tizim modelini hosil qilish prosedurasi to‘lalagicha formallashtirilgan bo‘lishi mumkin. Ikkinchi guruh metodlari hisoblanadi. Bunday metodlarga misol qilib, elektrotexnikadagi zanjir potensiallari farqi usullarini, mexanikada o‘rin almashish, avtomatik boshqarish tizimlarida funksional modellashtirish kabilarni keltirishimiz mumkin.

Ikkinchi guruh metodlari asosini quyidagi ikki qarashdan biri tashkil qiladi. Birinchi qarash elementlarda tashqi ta’sirlarning tarqalishi bir yo‘nalishliya’ni kirishdan – chiqishga qarab bo‘lishligiga yo‘l qo‘yishga asoslangan. Boshqacha qilib aytganimizda, elnmentning holatini o‘zgarishida yuklanish, manba elementiga ta’sirlarni to‘g’ri tarqatuvchi, bog’lovchi shoxlar orqali uzatilmaydi. Bu qarash hisoblash qurilmalarini mantiqiy sxemalarini hosil qilishda, avtomatik boshqarish tizimini modellashtirishda va turli ommaviy xizmat ko‘rsatuvchi ko‘rsatuvchi tizimlarda ya’ni metadarajada matematik modellar tashkil qilishda keng tarqalgan.

Ikkinchi qarashi birinchi qarashdagi qabul qilingan yo‘l qo‘yishlar bilan bog’lamagan ya’ni modellashtirilayotgan obyekt elementlarida tashqi bog’lanishlarni kirish va chiqish bog’lanishlariga ajratish shart emas, undan tashqari modellarining bir yo‘nalishiga bo‘lgan cheklanish olib tashlanadi. Bu qarash qarash ishlatish ancha murakkab hisoblanadi. Ularning invariantligi, fizik tizimlarda analoglar borligiga asoslangan, shuning uchun bu metodlar to‘g’ri analogiyaga asoslangan modellashtirish metodlari deyiladi. Funksional modellarni hosil qilish metodikasi. Konstruksiyalashning har bir iyerarxik darajasi uchun o‘zining baza elementlari to‘plami xarakterlidir.

Dasturiy ta’minotning matematik modellarini hosil qilish prosedurasi to‘la formallashtiriladigan bo‘lishi mumkin, chunki ular elementning tuzilishi va ishlashi to‘g’risidagi qabul qilingan u yoki bu formal bo‘lmagan qarashlar va yo‘l qo‘yishlarga asoslangandir. Berilgan baza elementlari modellari va elementlarni bog’lovchi sxemalar asosida tizimning to‘la matematik modelni hosil qilish metodi formallashtirilgan bo‘lishi mumkin. Makromodellardan konstruksiyalash masalalarini yiriklashib ketishini kamaytirish maqsadida foydalaniladi. Ular baza elementlariga nisbatan murakkabroq bo‘lgan tizim fragmentlari uchun yaratiladi. Iqtisod qilish ko‘rsatkichi bo‘yicha makromodellar, to‘la modellar fragmentiga nisbatan juda oddiy.

Makromodellarning iqtisodliliga, ularda adekvantlik soxasiga qarashli soddalashtirilgan tasavvurlarni qabul qilinishi hisobiga erishiladi. Tizim matematik modellarini va dasturiy ta’minot matematik modelini hosil qilish uchun bir xil metodlardan foydalaniladi. Dasturiy ta’minot matematik modellarini hosil qilish usullarida ko‘p o‘xshashliklar mavjud, lekin ba’zi bir farqlar ham mavjuddir. Dasturiy ta’minot matematik modellarini hosil qilishdagi umumiy metodika quyidagi proseduralardan tashkil topadi:

1. Modelda ko‘rsatilishi kerak bo‘lgan modellashtirilayotgan obyekt xususiyatlari aniqlanadi. Shu o‘rinda u obyektlarning chiqish parametrlari tartibi va q-tashqi parametrlari tartibi va obyekt haqida fikrlar aniqlanadi;

2. Model strukturasi tanlanadi. Ko‘pincha bu struktura injenerlar qabul qila olishlari uchun qulay bo‘lgan sxema ko‘rinishidagi modellarni matematik munosabatli modellarga o‘zgartirishning aniq qoidasi o‘rnatilgan bo‘lishi kerak

3. Identifikasiyalash masalasi yechiladi, ya’ni modelning berilgan strukturasi uchun model parametrlarining x-sonli qiymatlari hisoblanadi;

4. Tanlangan test masalalari bo‘yicha modeldagi xatoliklar aniqlanadi. Agar bu xatoliklar Ye ruxsat etilgan qiymatidan oshib ketsa, 2 va 3 punktlar, model strukturasini qaytadan tanlash uchun takrorlanadi. Agar Ye ruxsat etilgan qiymatdan kam yoki unga teng bo‘lsa, 5-punktlar o‘tiladi;

5. q va q qiymatlar aniqlanadi. Bu metodika bo‘yicha 3 va 5 punkt proseduralari formallashtiriladi. Analiz qilish metodini tanlash. Analiz qilish metodini tanlashda, foydalanilayotgan matematik modellar xususiyatlarini albatta e’tiborga olish kerak.

Matematik modellashtirishga nisbatan e’tiborli xususiyatlariga quyidagilar kiradi:

1. Yuqori aniqlikdagi natijalar olishni ta’minlovchi matematik modellardan foydalanishga intilish, zamonaviy qurilmalar murakkabligi, ulardan ishlatiladigan elementlar sonini ko‘payib ketishi natijasidagi model o‘lchamining kattalashib ketishi, dasturiy ta’minot larda ishlatiladigan modellarning o‘lchamlariga bo‘lgan chegaralanishlar, kompyuterlar tezkorligi va operativ hotirasiga bo‘lgan chegaralanishga bog’liqdir. Bu chegaralanishlarni T-kompyuter vaqtini va P-xotirani iqtisod qilish bo‘yicha analiz qilish metodlaridan foydalanish hisobiga anchayumshmtish imkoniyati mavjuddir. Shuning uchun T va P ko‘rsatkichlar dasturiy ta’minotlar uchun analiz qilish metodlarini tanlashda muhim o‘rinni egallaydi.

2. Matrisaning nollashtiriluvchanligi. dasturiy ta’minotlar konstruksiyalashtirilayotgan obyektlar matematik modellarida, kuchli nollashtirilgan matrisalar, ya’ni ko‘p miqdorda nolli elementlarga ega matrisalar ishtrok etadi. Nollashtirilganlik konstruksiyalashtirilayotgan obyektning har bir elementi boshqa elementlar bilan bevosita kam bog’linishga ega ekanligi bilan asoslanadi. S matrisaning nollashtirilganligi N matrisadagi elementlar umumiy soniga nisbatan olinadigan nollar soni bilan baholanadi. R-to‘lqinliklar esa N matrisadagi

elementlarga, nol bo‘lmagan elementlar munosabatini anglatadi, ya’ni R=1-S.

3. Modelning yomon moslashuvchanligi - bu modeldagi dastlabki ma’lumotlardagi ozgina xatoliklar, natijasidagi sezilarli xatoliklarni keltirib chiqarish mumkin bo‘lmagan modellarda namoyon bo‘ladi. Bunday modellardan foydalaniladigan interasion va ko‘p qadamli hisoblash proseduralarida hisoblashning mos kelmasligi va qat’iy bo‘lmasligini paydo bo‘lish ehtimolligi oshib ketadi. Dasturiy ta’minotlarni analiz qilish masalalarida yomon moslashuvchan modellar ko‘plab uchrab turadi. Shuning uchun dasturiy ta’minotlarni analiz qilish uchun, keyingi qadamlarda yoki iterasiyalarda xatoliklarni ko‘paytirib yubormaydigan, hisoblash jarayonlarini kechishini bir maromda ushlab turuvchi metodlardan foydalanish maqsadga muvofiq bo‘ladi. Dasturiy ta’minotlarda analiz qilish metod va algoritmlariga T va P ko‘rsatkichlari bo‘yicha yuqori darajadagi iqtisodlik, universallik va aniqlik talablari qo‘yiladi. Optimallash metodlari. Chiziqsiz dasturlash metodlari sinflar haqidagi ma’lumotlar quyidagi 9.1 - jadvalda keltirilgan:

9.1 – jadval

Chiziqsiz dasturlash metodlari sinflari

Sinflarga bo‘lish	Metodlar guruxi	Guruxlar xususiyati
Foydalanish x bo‘yicha F (x) maqsadli funksiya darajasi bo‘yicha	Nolinchi tartibli Birinchi tartibli Ikkinchi tartibli	Darajalardan foydalanmaydi. Birinchi tartibli darajalardan foydalaniladi ikkinchi tartibli darajalardan foydalaniladi
Izlanayotgan ekstremumlar xarakteri bo‘yicha	Global izlash Lokal izlash	Global ekstremum aniqlanadi Lokal ekstremum aniqlanadi
Chegaralashlar borligi bo‘yicha	Shartsiz optimalash Shartli optimallash	Chegaralanishlar bo‘lmaydi Chegaralanishlar borligida izlash
n-boshqaruvchi parametrlar soni	Bir o‘lchovli izlash Ko‘p o‘lchovli izlash	n=1 n>1

Yechilayotgan masalalar xarakteriga optimallash metod va algoritmlarning samaradorligi ko‘rsatkichlarining bog’liqligi, dasturiy ta’minotlari parametrlari optimallashning asosiy metodlari sifatida

chiziqsiz dasturlashning bitta yoki ikkita metodini asosiy metodlari deb tanlab olish imkoniyatini bermaydi. Samaradorlikning asosiy ko‘rsatkichlari bo‘lib quyidagilar hsioblanadi:

– izlash uchun ketadigan vaqt – berilgan dastlabki nuqtalaridan ekstremal nuqta atrofini topish jarayonida tizim tomonida matematik modelga qilinadigan murojaattlar soni:

– algoritmik puxtalik – berilgan masala uchun ekstremal nuqta atrofini topishga ketadigan chegeralangan izlash vaqt ehtimlligi;

– aniqlik – E – X* vektorlarning gurux masalalari uchun o‘rtacha mezoni, bu yerda E –ekstremal nuqta, X* - izlanish trayektoriyasining oxirgi nuqtasi.

Quyidagi optimallash metodlari mavjud:

– chiziqsiz dastulrash metodlari;
– maksimumni izlash algoritmlari;
– bir o‘lchovli minimallash metodlari;
– chiziqli dasturlashda simpleks metodi;
– diskret dasturlash metodlari;
– graflar haqida asosiy ma’lumotlar;

Konstruksiyalashtirilayotgan dasturiy tizimlar, konstruksiyalar, jarayonlar strukturasini izlhlashda graflar nazariyasi matematik apparatidan keng foydalaniladi. Bu apparatlar to‘plamlar nazariyasi va matematikaga asoslanadi. Graflar nazariyasidan foydalanish dasturiy ta’minotlarni konstruktiv xususiyatlarini ko‘rinarli qilib izohlash imkoniyatini beradi, undan tashqari turli algoritmlarni kompyuterlardan ishlatishda axborotlarni o‘zgartirishga qulay hisoblanadi.

9.2. Dasturiy ta’minotni modellashtirish vazifalari

Dasturiy ta’minotni modellashtirilayotganda, dasturiy ta’minot elementlari va ular orasidagi munosabatlar tavsiflanadi. UML ko‘p xollarda modellashtirishning obektga yo‘naltirilgan tili sifatida qo‘llaniladi, shuning uchun bunday yondashuvda tarkib topgan dasturiy ta’minot tarkibiy qisimlarining asosi bo‘lib sinflar va ular orasidagi munosabatlar hisoblanadi. Dasturiy konstruksiyalarni modellashtirish vazifasilari:

– Dastur bajarilishi vaqtida obektlar orasidagi aloqalar tuzilmasi;
– Ma’lumotlarni saqlash tuzilmasi;
– Dasturiy kod tuzilmasi;

– Ilovadagi komponentlar tuzilmasi;

– O‘zaro ta’sirlashuvchi qismlardan iborat murakkab obektlar tuzilmasi;

– Loyihadagi artefaktlar tuzilmasi;

– Foydalaniladigan hisoblash resurslari tuzilmasi.

– Dastur bajarilishi vaqtidagi obektlar o‘rtasidagi aloqalar tuzilmasi.

Obektga yo‘naltirilgan dasturlash paradigmasida dastur bajarilishi jarayoni dastur obektlari bir-biri bilan, xabarlar almashingan xolda o‘zaro ta’sirlashuvidan iborat. Xabarlashuv eng ko‘p tarqalgan tipi bo‘lib bir sinf obyekt metodini boshqa sinf obyekti metodidan chaqirish hisoblanadi. UMLda aloqalar tuzilmasini modellashtirish uchun sinflar diagrammasidagi uyushmalar munosabatlaridan foydalaniladi.

Ma’lumotlarni saqlash tuzulmasining dasturiy ta’minoti kompyuter xotirasudagi ma’lumotlarga ishlov beradi. Obektga yo‘naltirilgan dasturlash paradigmasida dastur bajarilishi vaqtida ma’lumotlarni saqlash uchun sinflar atributilari mo‘ljallangn. Biroq ish yuritishni avtomatlashtirish uchun mo‘ljallangan ilovalar katta qismi shunday tuzulganki, faqat ma’lum belgilangan ma’lumotlar(xammasi emas) kompyuter xotirasida nafaqta ilova seansi vaqtida, balki doimiy, ya’ni seanslar orasida saqlanishi lozim.

Dasturiy ta’minot konstruktsiyalarini modellashtirish masalasi dasturiy ilovalar uchun birinchi darajali hisoblanadi. Biroq bu masalani yechishning “moxiyat-aloqa” singari ishonchli metodlari mavjud. Bu metodlar(belgilashlar aniqligigacha) qutiblar karraligi ko‘rsatilgan uyushma shaklida UML da ham qo‘llaniladi.

Dasturiy ta’minot konstruktsiyalari kattaligi bo‘yicha juda sezilarli farqlashishi sir emas – katta va kichik dasturiy ta’minot konstruktsiyalari bo‘ladi. Kichik dasturiy ta’minot konstruktsiyalari uchun kod tuzilmasi deyarli axamiyatga ega emas, kattalari uchun esa aksincha deyarli hal qiluvchi axamiyatga ega. UML dasturlash tili bo‘lmaganligi uchun, model kod tuzilmasini bevosita aniqlamaydi, biroq tuzilma modeli bilvosita usul bilan kod tuzulmasiga sezilarli ta’sir ko‘rsatadi. Aksariyat instrumentlar bir yoki bir necha odatda obektga yo‘naltirilgan dasturlash tillari uchun kodning yarim avtomatik generetsiyani ta’minlaydi. Ko‘p xollarda model sinflari maqsadli til(yoki unga ekvivalent konstruksiyalar) sinflariga translatsiya qilinadi. Bundan tashqari, ko‘p instrumentlar modelda paketlar tuzilmasini hisobga oladi va uni maqsadli dasturiy ta’minot konstruktsiyalari mos sinf usti tuzilmalariga translatsiya qiladi. Shunday qilib, agar kod avtomatik generatsiyasi vositasi xarakatga tushirilsa, u

xolda modeldagi sinflar va paketlar tuzilmasi ilova kodi tuzilmasini deyarli to‘liq modellashtiradi.

Bir komponentga ega ilova modellashtirilishi shart bo‘lmaga trivial (oddiy) komponentlar tuzilmasiga ega. Lekin ko‘p zamonaviy ilovalar loyixalashtirish bosqichida ko‘p komponetlar(xatto ular taqsimlanmagan bo‘lsada) o‘zaro aloqasini namoyish etadi. Komponentali tuzilma ikki soxa tavsifini ifodalaydi: birinchidan, sinflarning komponentlar bo‘ylab qanday taqsimlanganligi, ikkinchidan, komponentlar qay tarzda (qanday interfeyslar orqali) bir – biri bilan o‘zaro ta’sirlashadi. Bu ikkala soxalar UML komponentlari diagrammalari yordamida modellashtiriladi.

O‘zaro ta’sirlashuvchi qismlardan iborat murakkab obektlar tuzilmasini modellashtirish uchun UML – tasniflovchi ichki tuzilmasi diagrammasi yangi vositasi qo‘llaniladi. Berilgan diagramma sinf va komponentlar ichki tuzilmasini tavsiflsh uchun foydalaniladi. Ko‘p qismlar o‘zaro ta’sirlashuvini xam tavsiflashga yo‘l qo‘yadigan yana bir moxiyat xam mavjuddir. Bu mohiyat koomperatsiya deb yuritiladi va ma’lum konteksdagi o‘zaro tasirlashuvni tasvirlash ucun xizmat qiladi. Ichki tuzilma nuqtai nazardan kooperatsiyaning sinf va komponentdan asosiy farqi shundaki, kooperatsiya o‘z qisimlarining egasi hisoblanmaydi va koomperatsiya qismlarining bog’lovchilari asotsasiya ko‘rinishdagi yaqqol ifodaga ega bo‘lmasliklari mumkun. Biroq sinf va komponentlar singari kooperatsiyalarda bajarish vaqtida funksiyalashadigan nusxalar bo‘lishi mumkun.

Dasturiy ta’minot konstruktsiyalaridagi artefaktlar tuzilmasining eng oddiy ilovalari bir artefakt – dastur bajariladigan kodidan tashkil topadi. Real ilovaning ko‘pchiligi o‘nlab, yuzlab va minglab turli komponentkarni o‘z tarkibiga hisoblaydi: bajariladigan ikkilik fayllari, resurs fayllari, birga yuruvchi turli hujjatlar, ma’lumot beruvchi fayllar, ma’lumotli fayllar va x.k. Katta ilova uchun nafaqat barcha artefaktlar aniq va to‘liq ro‘yhatga ega bo‘lish, balki tizim aniq nusxasiga aynan qaysilarikirishini ko‘rsatish ham mumkun. Gap shundaki, katta ilovalar uchun liyihada bir rtefaktning turli versiyalari mavjud. Turli tipdagi artefaktlarni tavsiflash uchun standart sterotiplar ko‘zda tutilgan UML komponentlar va joylashtirish diagrammalri bilan to‘liq tarzda modellashtiriladi. Deskriptlarning eng muhum tipi klassifikatorlar (tasniflovchilar) hisoblanadi. Tasniflovchi (classifier) – bu, bir tipli obektlar to‘plami deskriptorlari. Tasniflovchining asosiy va xarakteristik tasniflovchi (bevosita yoki bilvosita) nusxalarga ega bo‘lishi mumkun. UML da tasniflovchilar:

– Amal qiluvchi shaxs (actor); Foydalanish variant (use case).
– Artefakt (artifact); ma’lumotlar tipi (data type);
– Assotsiatsiya – uyushma (assostion); uyushma sinfi (association class);
– Interfeys (interfase); sinf (class);
– Kooperatsiya (collaboration); komponent (component);
– Tugun (node).

Tasniflovchining yetti eng muhim xossalarini tavsiflaymiz:

– Birinchidan, tasniflovchilar nomlarga ega. Nom model elementini indentifikatsiya qilish uchun hizmat qiladi va shuning uchun berilgan nomlar muxitida unikal(noyob) bo‘lishi kerak.

– Ikkinchidan, avval aytilganidek, tasniflovchi nusxalargabega bo‘lishi mumkun. Nushalar bevosita va bilvosita bo‘ladi. Agar qaysidir obekt bevosita A tasniflovchisi konstruktori yotdamida yaralgan bo‘lsa, u xolda bu obekt tasniflovchining bevosita yoki to‘g’ri nusxa (dipect inctance)si deb yuritiladi. Agar A tasniflovchi B tasniflovchi uyushmasi hisoblansa yoki huddi shunday, B tasniflovchi barcha nusxalari A tasniflovchi bilvosita nusxalari hisoblanadi. Berilgan xossa tranzit hisoblanadi: agar A tasniflovchi B tasniflovchi umumlashmasi hisoblansa va B tasniflovchi C tasniflovchi umumlashmasi bo‘lsa, C tasniflovchi barcha nusxalari shuningdek A ning bilvosita nusxalari hisoblanadi.

– Uchunchidan, tasniflovchi mavxum yoki aniq bo‘lishi mumkun. Mavxum (abstract) tasniflovchi bevosita nusxalarga ega bo‘la olmaydi va bu holda uning nomi bilan ajratiladi. Aniq (concrete) tasniflovchi bevosita nusxalarga ega bo‘la oladi va bu xolda uning nomi to‘g’ri shrift bilan yoziladi. Mavxum tasniflovchi – bu shunday obektlar to‘plami deskriptoriki, unda elementlar bevosita tavsifi bo‘lmaydi, lekin ushbu tavsiflovchi boshqa tasniflovchilar bilan umumlashma munosabati bilan bog’liq va ular nusxalari to‘plamlari birlashmasi berilgan mavxum tasniflovchi nusxalari to‘plami hisoblanadi. Boshqa so‘z bilan aytganda, to‘plam bevosita emas, kichik guruhlar yig’indisi orqali aniqlanadi. Masalan, interfeys bo‘lg’usi mavhum sinf bevosita nushalarga ega bo‘la olmaydi, biroq uni realizatsiya qiluvchi sinf yoki interfeys tasniflovchi hisoblanadi.

– To‘rtinchidan, tasniflovchi ko‘rinishga ega. Ko‘rinish (visibility) bir tasniflovchi tashkil etuvchi boshqa tasniflovchida foydalana olish mumkunligin aniqlaydi. Agar ma’lum konteksda nimadir mumkun bo‘las va qandaydir tarzda foydalanish mumkun bo‘lsa, u holda u ko‘rinarli (bu

konteksda) hisoblanadi. Agar u ko‘rinarli bo‘lmasa u xolda undan foydalanib bo‘lmaydi. Ko‘rinish to‘rt ma’nodan biriga ega bo‘lishi mumkun:

- Ochiq (+ belgisi yoki public kalit so‘zi bilan belgilanadi);
- Ximoyalangan (# belgisi yoki protected kalit so‘zi bilan belgilanadi);
- Yopiq (- belgisi yoki private kalit so‘zi bilan belgilanadi);
- Paket (~ belgisi yoki package kalit so‘zi bilan belgilanadi).

– Beshinchidan, tasniflovchi tashkil etuvchilari xarakat soxasiga ega. Xarakat soxasi (scope) nusxalarda tasniflovchi tashkil etuvchisi o‘zini qanday nomoyon etishini aniqlaydi, ya/ni tashkil etuvchi o‘z qiymatlarining nushalariga ega yoki bir ma;noni qo‘shma tarzda foydalaniladi. Xarakat soxasi mumkun bo‘lgan ikki qiymatga ega:

– Nusxa (instance) – xechqanday maxsus belgilanmaydi, chunki default shakli olinadi;

– Tasniflovchi (classifier) – tasniflovchi tashkil etuvchisi tavsifi tagiga chizib qo‘yiladi.

Agar tashkil etuvchi xarakat soxasi nusxa hisoblansa, u holda tasniflovchi nushasi o‘z tashkil etuvchi qiymatiga ega bo‘ladi. Bu har bir obekt – sinf nushasi – boshqa obektlar berilgan atributi, shu sinf nushalari qiymatlaridan mustaqil tarzda o‘zgara oladigan atribut o‘z hususiy qiymatiga ega. Agar tashkil etuvchining xarakat soxasi tasniflovchi hisoblansa, u xolda tasniflovchi barcha nusxalari birgalikda tashkil etuvchining bir qiymatidan foydalaniladi. Masalan, konstruktr xarakat soxasi, odatda tasniflovchi(sinf) bo‘ladi, chunki u ushbu sinf barcha nusxalari uchun umumiy amaliyot(protsedura) hisoblanadi.

Modellar talablar injiniringi jarayonida tizim uchun talablarni hosil qilishda foydalaniladi. Siz mavjud tizimlarning modelini va ishlab chiqarilayotgan tizimning modelini tuzishingiz mumkin.

1.Mavjud tizim modellari talablar injiringi mobaynida foydalaniladi. Ular mavjud tizimning nima ish bajarishini aniqlashtiradi va tizimning kuchli va kuchsiz tomonlarini muhokama qilishga asos bo‘ladi. Bu yangi tizim uchun talablar ishlab chiqishga olib keladi.

2. Yangi tizim modellari talablar injiniringi davomida yordam berish uchun ishlatiladi. Injinerlar dizayn bo‘yicha takliflarni muhokama qilishda modellardan foydalanadi. Tizim modelining eng muhim tomoni shundaki unda tizim haqidagi batafsil ma’lumotlar tashlab ketiladi. Model o‘rganilayotgan tizimning mavhum ko‘rinishidir. Siz tizimni turli xil ko‘rinishlarini ko‘rsatish uchun turli xil modellarni ishlab chiqishingiz

mumkin. Masalan:

1.Tashqi ko‘rinish, tizimning konteksti yoki muhitini modellashtirish.

2.O‘zaro munosabatlar ko‘rinishi, tizim bilan muhit yoki tizim komponentalari o‘rtasidagi o‘zaro munosabatni modellashtirish.

3.Strukturaviy ko‘rinish, tizim tomonidan ishlov berilayotgan ma’lumotlar strukturasi yoki tizim tashkilotini modellashtirish.

4.Xatti harakatlar ko‘rinishi, tizimning dinamik xatti harakatlari va hodisalarga Qanday javob berishini modellashtirish.

Tizimning turli xil modellarini yaratish uchun UML bir nechta diagrammalarga ega.

1.Faoliyat diagrammalari, jarayondagi faoliyatlarni ko‘rsatadi.

2.Foydalanish holati diagrammari, tizim va uning muhiti o‘rtasidagi munosabatni ko‘rsatadi.

3.Ketma-ketlik diagrammalari, shaxs va tizim va tizim komponentalari orasidagi munosabatlarni ko‘rsatadi.

4.Sinf diagrammalari, tizimdagi obyektlar sinflari va ularning o‘zaro munosabatini ko‘rsatadi

5. Holat diagrammari, tizimning ichki va tashqi hodisalarga ta’sirini ko‘rsatadi.

UML(Unified Modeling Language) - birlashgan modellashtirish tili dasturiy ta’minot tizimlarini modellashda 13 ta turli xil diagramma turlaridan foydalanadi.

UML dasturiy ta’minot tizimlarini modelini yaratishda standart yondashuv deb qabul qilingan.

Konteks modellar

Konteks modellar tizimning tezkor kontekstini ko‘rsatishda foydalaniladi. Arxitekturaviy modellar tizim va uning boshqa tizimlar bilan munosabatini ko‘rsatadi.

Tizim chegaralari

Tizim chegaralari nima, tizim ichida va nima tizim tashqarisidaligini ko‘rsatadi. Ular ishlab chiqarilayotgan tizimda foydalanilayotgan yoki bog’liq bo‘lgan boshqa tizimlarni ko‘rsatadi. Konteks modellar muhitdagi ishlab chiqarilayotgan tizimni emas balki muhitdagi boshqa tizimlarni ko‘rsatadi.

Jarayon modellar ishlab chiqarilayotgan modellarni ko‘rsatadi. UML diagrammalar jarayon modellarda foydalaniladi.

9.3. UML da modellashtirish

Yagona modellashtirish tili (UML) dasturiy ta’minot tizimlarini, shuningdek, biznes modellarini va boshqa dasturiy ta’minot bo‘lmagan tizimlarni belgilash, vizuallashtirish, qurish va hujjatlashtirish uchun tildir. UML - bu katta va murakkab tizimlarni modellashtirish uchun ilgari muvaffaqiyatli qo‘llanilgan muhandislik texnikasining birikmasidir.

UML yaratuvchilari uni dasturiy ta’minot tizimlarini, biznes tizimlarini va turli xarakterdagi boshqa tizimlarni aniqlash, taqdim etish, loyihalash va hujjatlashtirish uchun til sifatida taqdim etadilar. UML notatsiya va metamodelni belgilaydi. Belgilash - bu modellarda qo‘llaniladigan grafik ob’ektlar to‘plami; bu modellashtirish tilining sintaksisidir. Birlashtirilgan modellashtirish tili (UML):

- ob’ektga yo‘naltirilgan (OO) dasturlash tillariga bog’liq emas;
- foydalanilgan loyihani ishlab chiqish metodologiyasiga bog’liq emas;
- har qanday OO dasturlash tilini qo‘llab-quvvatlashi mumkin.

UML ochiq manba va asosiy yadroga kengaytirilishi mumkin. UMLda siz ko‘pincha bir-biridan juda farq qiluvchi turli mavzulardagi sinflar, ob’ektlar va komponentlarni mazmunli tasvirlashingiz mumkin.

UML diagrammasi

Tizim dizayneri ixtiyorida Rational Rose quyidagi diagramma turlarini taqdim etadi, ularning ketma-ket yaratilishi butun loyihalashtirilgan tizim va uning alohida komponentlari haqida to‘liq tasavvurga ega bo‘lishga imkon beradi:

- Ish diagrammasidan foydalaning
- Joylashtirish diagrammasi (topologiya diagrammasi);
- Davlat diagrammasi;
- O‘zaro ta’sir diagrammasi Faoliyat diagrammasi
- Ketma-ketlik diagrammasi
- Hamkorlik diagrammasi
- Sinf diagrammasi
- Komponent diagrammasi
- Xulq-atvor sxemalari
- Faoliyat diagrammasi
- Amalga oshirish sxemalari

Ushbu diagrammalarning har biri tizim modelining boshqa ko‘rinishini konkretlashtiradi. Bunday holda, foydalanish diagrammasi boshqa barcha diagrammalarni qurish uchun boshlang’ich nuqta bo‘lgan

tizimning kontseptual modelini ifodalaydi. Sinf diagrammasi - bu tizimning strukturaviy dizaynining statik tomonlarini aks ettiruvchi mantiqiy model va mantiqiy modelning navlari bo‘lgan xatti-harakatlar diagrammasi uning ishlashining dinamik tomonlarini aks ettiradi. Amalga oshirish diagrammasi tizimning tarkibiy qismlarini ifodalash va uning jismoniy modeliga murojaat qilish uchun ishlatiladi.

Yuqoridagi diagrammalardan ba’zilari ikki yoki undan ortiq kichik turlarni ko‘rsatish uchun ishlatiladi. Mustaqil tasvirlar sifatida quyidagi diagrammalardan foydalaniladi: foydalanish holatlari, sinflar, holatlar, faoliyatlar, ketma-ketlik, hamkorlik, komponentlar va joylashtirish. UML diagrammalari uchun uchta turdagi vizual belgilar mavjud bo‘lib, ular tarkibidagi ma’lumotlar jihatidan muhim:

- ulanishlar, tekislikda turli xil chiziqlar bilan ifodalangan;
- matn individual geometrik shakllar chegaralarida joylashgan;
- grafik belgilar diagrammalarning vizuallari yaqinida chizilgan.

Diagrammalarni grafik ko‘rsatishda quyidagi qoidalarga rioya qilish tavsiya etiladi:

- har bir diagramma simulyatsiya qilingan domenning ba’zi bir qismining to‘liq tasviri bo‘lishi kerak;
- diagrammada ko‘rsatilgan model ob’ektlari bir xil kontseptual darajada bo‘lishi kerak;
- ob’ektlar haqidagi barcha ma’lumotlar diagrammada aniq ko‘rsatilishi kerak;
- diagrammalar qarama-qarshi ma’lumotlarni o‘z ichiga olmaydi;
- diagrammalar matnli ma’lumotlar bilan ortiqcha yuklanmasligi kerak;
- uning barcha elementlarini to‘g’ri talqin qilish uchun har bir diagramma o‘zini o‘zi etarli bo‘lishi kerak;
- muayyan tizimni tavsiflash uchun zarur bo‘lgan diagramma turlarining soni qat’iy belgilanmagan va ishlab chiquvchi tomonidan belgilanadi;
- tizim modellari faqat UML yozuvida belgilangan elementlarni o‘z ichiga olishi kerak.

UMLdagi ob’ektlar

UML to‘rt turdagi ob’ektlarni belgilaydi: strukturaviy, xulq-atvor, guruhlash va izohlash. Ob’ektlar tilning asosiy ob’ektga yo‘naltirilgan elementlari bo‘lib, ular yordamida modellar yaratiladi.

Strukturaviy tuzilmalar UML modellaridagi otlar. Odatda, ular tizimning kontseptual yoki jismoniy elementlariga mos keladigan modelning statik qismlarini ifodalaydi. Strukturaviy ob'ektlarga sinf, interfeys, hamkorlik, foydalanish holati, komponent, tugun, aktyor misol bo'ladi. Xulq-atvor ob'ektlari UML modelining dinamik komponentlaridir. Bu modelning vaqt va makondagi xatti-harakatlarini tavsiflovchi fe'llardir. Xulq-atvorning ikkita asosiy turi mavjud:

– o'zaro ta'sir - bu xulq-atvor bo'lib, uning mohiyati aniq maqsadga erishish uchun muayyan kontekstdagi ob'ektlar o'rtasida xabarlar almashinuvidir;

– avtomat - turli hodisalarga javoban ob'ekt yoki o'zaro ta'sir o'tadigan holatlar ketma-ketligini aniqlaydigan xatti-harakatlar algoritmi.

Ob'ektlarni guruhlash UML modelining tashkiliy qismlari hisoblanadi. Bu modelni parchalash mumkin bo'lgan bloklardir. Bunday asosiy ob'ektning bitta nusxasi mavjud - bu paket. Paketlar elementlarni guruhlarga ajratish uchun universal mexanizmdir. Strukturaviy, xulq-atvor va boshqa guruhlash ob'ektlari paketga joylashtirilishi mumkin. Dastur ishlayotgan vaqtda mavjud bo'lgan komponentlardan farqli o'laroq, paketlar faqat kontseptualdir, ya'ni ular faqat ishlab chiqish jarayonida mavjud bo'ladi.

Annotatsiya ob'ektlari UML modelining tushuntirish qismlari: qo'shimcha tavsif, tushuntirish yoki modelning istalgan elementiga izohlar uchun izohlar. Annotatsiya elementlarining faqat bitta asosiy turi mavjud, annotatsiya. Eslatma norasmiy yoki rasmiy matnda ifodalangan diagrammalarga sharhlar yoki cheklovlar berish uchun ishlatiladi.

UMLdagi munosabatlar

UMLda quyidagi aloqa turlari aniqlanadi: qaramlik, assotsiatsiya, umumlashtirish va amalga oshirish. Ushbu munosabatlar UML-dagi asosiy yopishtiruvchi konstruksiyalar bo'lib, shuningdek, ob'ektlar modellarni yaratishda qanday foydalaniladi.

Tobelik ikki shaxs o'rtasidagi semantik munosabat bo'lib, ulardan birining o'zgarishi, mustaqil, ikkinchisining semantikasiga ta'sir qilishi mumkin, bog'liq.

Uyushma- ob'ektlar orasidagi semantik yoki mantiqiy bog'lanishlar to'plamini tavsiflovchi tizimli munosabatlar.

Umumlashtirish ixtisoslashgan element ob'ekti (avlod) umumiy element ob'ekti (ajdodi) o'rniga almashtirilishi mumkin bo'lgan munosabatdir. Shu bilan birga, ob'ektga yo'naltirilgan dasturlash

tamoyillariga muvofiq, bola ota-onasining (ota-onaning) tuzilishi va xatti-harakatlarini meros qilib oladi.

Amalga oshirish klassifikatorlar o‘rtasidagi semantik munosabat bo‘lib, unda bir klassifikator majburiyatni belgilaydi, ikkinchisi esa uning bajarilishini kafolatlaydi. Amalga oshirish munosabatlari ikki holatda yuzaga keladi:

– interfeyslar va ularni amalga oshiradigan sinflar yoki komponentlar o‘rtasida;

– foydalanish holatlari va ularni amalga oshiradigan hamkorlik o‘rtasida.

Umumiy UML mexanizmlari

UMLda tizimning aniq tavsifi uchun umumiy mexanizmlar qo‘llaniladi:

– spetsifikatsiyalar;
– qo‘shimchalar (bezaklar);
– bo‘linmalar (umumiy bo‘linmalar);
– kengaytirish mexanizmlari.

UML nafaqat grafik tildir. O‘z yozuvining har bir grafik elementi mavjud spetsifikatsiya tegishli til konstruktsiyasining matnli tasvirini o‘z ichiga oladi. Masalan, sinf belgisi o‘zining atributlari, operatsiyalari va xatti-harakatlarini tavsiflovchi spetsifikatsiyaga ega, garchi vizual ravishda diagrammada piktogramma ko‘pincha ushbu ma’lumotlarning faqat kichik qismini aks ettiradi. Bundan tashqari, model ushbu sinfning boshqa ko‘rinishini o‘z ichiga olishi mumkin, bu uning butunlay boshqa tomonlarini aks ettiradi, ammo shunga qaramay, spetsifikatsiyaga mos keladi. Shunday qilib, grafik UML yozuvi tizimni vizualizatsiya qilish uchun ishlatiladi va spetsifikatsiyalar yordamida uning tafsilotlarini tavsiflaydi. Deyarli har bir UML elementi o‘zining eng muhim xususiyatlarini vizual tasvirini ta’minlaydigan noyob grafik tasvirga ega.

Ob’ektning "sinf" belgisi uning nomi, atributlari va operatsiyalarini o‘z ichiga oladi. Sinf spetsifikatsiyasi atributlar va operatsiyalarning ko‘rinishi, sharhlar yoki sinfning mavhum ekanligini ko‘rsatish kabi boshqa tafsilotlarni o‘z ichiga olishi mumkin. Ushbu tafsilotlarning aksariyati grafik yoki matn sifatida ko‘rsatilishi mumkin. qo‘shimchalar sinfni ifodalovchi standart to‘rtburchakka. Ob’ektga yo‘naltirilgan tizimlarni modellashtirishda ma’lum bir narsa mavjud. bo‘linish vakili bo‘lgan shaxslar. Birinchidan, sinflar va ob’ektlarga bo‘linish mavjud. Sinf mavhumlikdir, ob’ekt esa bu abstraktsiyaning konkret timsolidir. Shu

munosabat bilan deyarli barcha til konstruktsiyalari sinf / ob’ekt ikkiligi bilan tavsiflanadi. Shunday qilib, foydalanish holatlari va foydalanish holatlari, komponentlar va komponent namunalari, tugunlar va tugun misollari mavjud.

Grafik tasvirda ob’ekt uchun sinf uchun bo‘lgani kabi bir xil belgidan foydalanish va nomning tagiga chizish odatiy holdir. Ikkinchidan, interfeysga va uni amalga oshirishga bo‘linish mavjud. Interfeys majburiyatlarni e’lon qiladi va amalga oshirish ushbu majburiyatlarning aniq timsolini ifodalaydi va e’lon qilingan semantikaga qat’iy rioya qilinishini ta’minlaydi. Shu sababli, deyarli barcha UML konstruksiyalari interfeys/amalga oshirish dualligi bilan tavsiflanadi. Masalan, foydalanish holatlari kooperativlar tomonidan, operatsiyalar esa usullar bo‘yicha amalga oshiriladi. UML ochiq tildir, ya’ni u boshqariladigan kengaytmalarga domen modellarining o‘ziga xos xususiyatlarini aks ettirish imkonini beradi. UML kengaytma mexanizmlariga quyidagilar kiradi:

– stereotiplar (stereotiplar) - UML lug’atini kengaytirish, mavjud til elementlariga asoslanib, ma’lum bir muammoni hal qilishga qaratilgan yangilarini yaratishga imkon beradi;

– teglangan qiymat - element spetsifikatsiyasiga qo‘shimcha ma’lumotlarni kiritish imkonini beruvchi asosiy UML konstruktsiyalarining xususiyatlarini kengaytiradi;

– cheklovlar - UML konstruksiyalarining semantikasini kengaytiradi, bu sizga yangi yaratish va mavjud qoidalarni bekor qilish imkonini beradi.

– Ushbu uchta tilni kengaytirish mexanizmlari birgalikda uni loyiha ehtiyojlariga yoki ishlab chiqish texnologiyasining o‘ziga xos xususiyatlariga muvofiq o‘zgartirishga imkon beradi.

9-bob bo‘yicha xulosalar

Matematik ta’minotning universalligi deyilganda, bu ta’minotdan bir qancha konstruksiyalash obyektlarini konstruksiyalashda foydalanish imkoniyati borligi tunushiladi. Dasturiy ta’minotda algoritmik puxta metod bo‘lmagan metodlardan foydalanish maqsadga muvofiq bo‘lmaydi. Konstruksiyalashning har bir bosqichida o‘zining oraliq natijalari va har bir bosqichdagi xatolik kelib chiqish manbaalari mavjud. Shu sababli yomon sharoitga moslashtirish natijasida hatoliklar juda kattalashib ketishi mumkin. Kompyuter vaqtini tejash talabidan keyingi talab, xotira hajmini tejash talabidir. Bu talab matematik ta’minotning

qanchalik iqtisodlashtirilganligii ko‘rsatadi va tuzilgan dastur uzunligi bilan harakterlanadi.

Matematik modellarni hosil qilish metodlarini ikki guruhga ajratish mumkin. Birinchi guruh metodlariga, turli iyerarxik darajalarda elementlar matematik modellarini va tizimlar makromodellarini hosil qilish metodlari kiradi. Dasturiy ta’minotning matematik modellarini hosil qilish prosedurasi to‘la formallashtiriladigan bo‘lishi mumkin, chunki ular elementning tuzilishi va ishlashi to‘g’risidagi qabul qilingan u yoki bu formal bo‘lmagan qarashlar va yo‘l qo‘yishlarga asoslangandir. Dasturiy ta’minot matematik modellarini hosil qilish usullarida ko‘p o‘xshashliklar mavjud, lekin ba’zi bir farqlar ham mavjuddir. Dasturiy ta’minotni modellashtirilayotganda, dasturiy ta’minot elementlari va ular orasidagi munosabatlar tavsiflanadi. Dasturiy ta’minot konstruktsiyalarini modellashtirish masalasi dasturiy ilovalar uchun birinchi darajali hisoblanadi. Biroq bu masalani yechishning “moxiyat-aloqa” singari ishonchli metodlari mavjud.Yagona modellashtirish tili (UML) dasturiy ta’minot tizimlarini, shuningdek, biznes modellarini va boshqa dasturiy ta’minot bo‘lmagan tizimlarni belgilash, vizuallashtirish, qurish va hujjatlashtirish uchun tildir. UML - bu katta va murakkab tizimlarni modellashtirish uchun ilgari muvaffaqiyatli qo‘llanilgan muhandislik texnikasining birikmasidir. UML to‘rt turdagi ob’ektlarni belgilaydi: strukturaviy, xulq-atvor, guruhlash va izohlash.

9-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotni modellashtirish deganda nimani tushunasiz?
2. Matematik ta’minotning universalligi deyilganda nimani tushunasiz?
3. Eksperimental metodlar deganda nimani tushunasiz?
4. Dasturiy ta’minot matematik modellarini hosil qilishdagi umumiy metodika qanday jarayonlardan iborat?
5. Dasturiy ta’minotni modellashtirish vazifalari nimalardan iborat.
6. Yagona modellashtirish tili (UML) haqida nimalar bilasiz.
7. UML diagrammalari haqida nimalar bilasiz.
8. UMLda necha turdagi ob’ektlar mavjud.
9. UMLdagi munosabatlarni aytib bering.
10. UML diagrammalari uchun necha turdagi vizual belgilar mavjud.

10-BOB. DASTURIY TA’MINOTNING ARXITEKTURASI

10.1. Dasturiy ta’minot arxitekturasi tushunchasi

Dasturiy ta’minot arxitekturasi (англ. *software architecture*) dasturiy ta’minot tizimini tashkil etish bo‘yicha muhim qarorlar to‘plamidir. Arxitektura quyidagilarni o‘z ichiga oladi:

– tizim yordamida tuziladigan strukturaviy elementlar va ularning interfeyslarini tanlash, shuningdek, strukturaviy elementlarning hamkorligi doirasidagi xatti-harakatlari;

– tuzilma va xatti-harakatlarning tanlangan elementlarini tobora kattaroq tizimlarga ulash;

– butun tashkilotni boshqaradigan arxitektura uslubi - barcha elementlar, ularning interfeyslari, hamkorligi va aloqasi

Dasturiy ta’minot arxitekturasini (SW) hujjatlashtirish ishlab chiquvchilar o‘rtasidagi aloqa jarayonini soddalashtiradi, qabul qilingan loyiha qarorlarini qayd etish va tizimni ishlatuvchi xodimlariga ular haqida ma’lumot berish, komponentlar va loyiha shablonlarini boshqalarda qayta ishlatish imkonini beradi. “Dasturiy ta’minot arxitekturasi”ning umumiy qabul qilingan ta’rifi yo‘q. Shunday qilib, Software Engineering Institute veb-saytida ushbu kontseptsiyaning 150 dan ortiq ta’riflari berilgan.

Komyuter ilmlari sohasi yaratilganidan beri dasturiy tizimlarning murakkabligi bilan bog’liq muammolarga duch keldi. Ilgari murakkablik muammolari ishlab chiquvchilar tomonidan ma’lumotlar tuzilmalarini to‘g’ri tanlash, algoritmlarni ishlab chiqish va vakolatlarni ajratish kontseptsiyasini qo‘llash orqali hal qilingan. "Dasturiy ta’minot arxitekturasi" atamasi dasturiy ta’minotni ishlab chiqish sanoati uchun nisbatan yangi bo‘lsa-da, bu sohaning asosiy tamoyillari 1980-yillarning o‘rtalaridan boshlab dasturiy ta’minotni ishlab chiqish pionerlari tomonidan qo‘llanilgan. Tizimning dasturiy ta’minot arxitekturasini tushunish va tushuntirishga birinchi urinishlar noaniqliklarga to‘la edi va tashkilotchilikning yetishmasligidan aziyat chekdi, ko‘pincha chiziqlar bilan bog’langan blok sxemalarning diagrammasi sifatida ifodalangan. 1990-yillarda ushbu fanning asosiy jihatlarini aniqlash va tizimlashtirishga harakat qilindi. Bu vaqt ichida loyihalash shablonlari to‘plami, loyihalash uslublari, eng yaxshi amaliyotlar, tavsiflash tillari va rasmiy mantiqning dastlabki to‘plami ishlab chiqilgan.

Dasturiy ta’minot arxitekturasi sohasining asosiy g’oyasi - bu mavhumlik va vakolatlarni ajratish orqali tizimning murakkabligini kamaytirish g’oyasi. Bugungi kunga qadar "dasturiy ta’minot arxitekturasi" atamasining aniq ta’rifi bo‘yicha kelishuv mavjud emas. Tizimni ishlab chiqishning "to‘g’ri" yo‘li haqida aniq qoidalarga ega bo‘lmagan hozir vaqtda dasturiy ta’minot arxitekturasi loyihalash rivojlanayotgan soha sifatida hali ham fan va san’at aralashmasidir. "San’at" jihati shundaki, har qanday tijorat tizimi dastur yoki topshiriqni nazarda tutadi.

Dasturiy ta’minot arxitekturasi foydalanuvchisi nuqtai nazaridan, dasturiy ta’minot arxitekturasi har bir foydalanuvchining ixtisosligi bilan bog’liq muammolarni ko‘chirish va hal qilish yo‘nalishini ta’minlaydi, masalan, manfaatdor tomon, dasturiy ta’minot ishlab chiquvchisi, dasturiy ta’minotni qo‘llab-quvvatlash jamoasi, dasturiy ta’minotni ta’minlovchi, dasturiy ta’minotni joylashtirish bo‘yicha mutaxassis, dasturiy ta’minotni testdan o‘tkazuvchi mutaxasis, shuningdek, oxirgi foydalanuvchilar. Shu ma’noda, dasturiy ta’minot arxitekturasi aslida tizimga turli nuqtai nazarlarni birlashtiradi. Ushbu bir nechta turli nuqtai nazarlarni dasturiy ta’minot arxitekturasida birlashtirish mumkinligi dasturiy ta’minotni ishlab chiqish bosqichidan oldin ham dasturiy ta’minot arxitekturasini yaratish zarurati va maqsadga muvofiqligi foydasiga dalildir.

Dasturiy ta’minot arxitekturasi kontseptsiya sifatida 1968 yilda Edsger Deykstra va 1970-yillarning boshlarida Devid Parnassusning tadqiqot ishlaridan boshlangan. Bu olimlar dasturiy ta’minot tizimining tuzilishi muhimligini va to‘g’ri tuzilmani yaratish muhimligini ta’kidladilar. Ushbu sohani o‘rganish 1990-yillarning boshidan arxitektura uslublari (naqshlar), arxitektura tavsifi tillari, arxitektura hujjatlari va rasmiy usullar bo‘yicha tadqiqot ishlari bilan mashhur bo‘ldi. Dasturiy ta’minot arxitekturasining fan sifatida rivojlanishida ilmiy-tadqiqot muassasalari muhim rol o‘ynaydi. Karnegi Mellon universitetidan Meri Shou va Devid Garlan "Dasturiy ta’minot arxitekturasi: 1996 yilda yangi fan bo‘yicha istiqbollar" nomli kitob yozdilar, unda ular komponentlar, bog’lovchilar, uslublar va boshqalar kabi dasturiy ta’minot arxitekturasi tushunchalarini ilgari surdilar. Kaliforniya universitetida Irvine dasturiy ta’minot tadqiqot instituti birinchi navbatda arxitektura uslublari, arxitektura tavsifi tillari va dinamik arxitekturalarni o‘rgandilar.

Birinchi dasturiy ta’minot arxitektura standarti IEEE 1471: ANSI/IEEE 1471-2000: Asosan dasturiy ta’minot tizimlarini tavsiflash bo‘yicha ko‘rsatmalar. U 2007 yilda ISO ISO/IEC 42010:2007 nomi bilan qabul qilingan. Arxitektura tavsifi tillari (ADLS) dasturiy ta’minot arxitekturasini tavsiflash uchun ishlatiladi. AADL (SAE standarti), Rayt (Karnegi Mellon universitetida ishlab chiqilgan), Acme (Karnegi Mellon universitetida ishlab chiqilgan), xADL (UCIda ishlab chiqilgan), Darvin (London Imperial kollejida ishlab chiqilgan), DAOP-ADL (Malaga universitetida ishlab chiqilgan) va ByADL (L’Aquila universiteti, Italiya) kabi turli tashkilotlar tomonidan bir nechta turli ADLS ishlab chiqilgan. Ushbu tillarning barchasi uchun umumiy elementlar komponent, bog’lovchi va konfiguratsiya tushunchalaridir. Bundan tashqari, ixtisoslashgan tillardan tashqari, arxitekturani tavsiflash uchun UML birlashtirilgan modellash tili ko‘pincha ishlatiladi.

Dasturiy ta’minot arxitekturasi odatda bino qurilishidagi har xil turdagi chizmalarga o‘xshash bir nechta ko‘rinishlarni o‘z ichiga oladi. ANSI/IEEE 1471-2000 tomonidan belgilangan ontologiyada ko‘rinishlar ma’lum bir manfaatdor tomonlar to‘plami nuqtai nazaridan arxitekturani tavsiflash uchun misollardir. Arxitektura ko‘rinishi 2 komponentdan iborat:

– Elementlar
– Elementlar orasidagi aloqalar

Arxitektura ko‘rinishlarini 3 asosiy turga bo‘lish mumkin:

1. Modulli ko‘rinishlar (inglizcha - *module views*) - tizimni turli xil dasturiy bloklar strukturasi sifatida ko‘rsatish.

2. Komponentlar-va-konnektorlar (inglizcha - *component-and-connector views*) - tizimni parallel ishlaydigan elementlarning (komponentlarning) tuzilishi va ularning o‘zaro ta’sirini (bo‘lovchilar) ko‘rsatish.

3. Joylashtirish *(*inglizcha - *allocation views)* - tizim elementlarini tashqi muhitda joylashtirishni ko‘rsatadi.

Modulli ko‘rinishlarga misollar:

– Dekompozitsiya (*ing. decomposition view*) – “submoduldir” munosabati kontekstidagi modullardan iborat.

– Foydalanish (*ing. uses view*) – “foydalanish” munosabati kontekstidagi modullardan iborat (ya’ni bir modul boshqa modul xizmatlaridan foydalanadi)

– Qatlamli ko‘rinish (ing. *layered view*) - funksionallik bilan bog’liq modullar guruhlarga (darajalarga) birlashtirilgan tuzilmani ko‘rsatadi.

– Sinflar turi/umumlashmalar (ing. c*lass/generalization view*) – “merosdan olingan” va “nasol” munosabati orqali bog’langan sinflardan iborat.

Komponent va konnektor turlariga misollar:

– Jarayon ko‘rinishi (inglizcha - *process view*) - aloqa, sinxronizatsiya va/yoki istisno operatsiyalari bilan bog’langan jarayonlardan iborat;

– Parallel ko‘rinish (inglizcha - *concurrency view*) - komponentlar va konnektorlardan iborat bo‘lib, ulagichlar "mantiqiy oqimlar";

– Umumiy ma’lumotlar ko‘rinishi (inglizcha - *shared-data (repository) view*) — doimiy ma’lumotlarni yaratuvchi, saqlaydigan va qabul qiluvchi komponentlar va ulagichlardan iborat;

– Mijoz-server ko‘rinishi (inglizcha - *client-server view*)- o‘zaro ta’sir qiluvchi mijozlar va serverlardan, shuningdek ular orasidagi konnektorlardan iborat (masalan, protokollar va umumiy xabarlar).

Joylashish turlariga misollar:

– Deployment (ing. deployment view) - dasturiy ta’minot elementlari, ularni jismoniy muhitda joylashtirish va aloqa elementlaridan iborat.

– Amalga oshirish (ing. *implementation view*) - dastur elementlari va ularning turli muhitdagi fayl tuzilmalariga mos kelishi (ishlab chiqish, integratsiya va boshqalar) dan iborat.

– Ish topshirig’i ko‘rinishi (ing. *work assignment view*)- modullardan va ularning har birini amalga oshirish uchun kim mas’ul ekanligi tavsifidan iborat.

Dasturiy ta’minot arxitekturasini tavsiflash uchun bir nechta tillar ishlab chiqilgan bo‘lsa-da, hozircha qaysi qarashlar to‘plamini ma’lumotnoma sifatida qabul qilish kerakligi haqida kelishuv mavjud emas. UML tili "dasturiy ta’minot tizimlarini (va nafaqat) modellashtirish uchun" standart sifatida yaratilgan. Loyihalashtirilgan tizimni turli sifat atributlari bilan qondirish uchun turli konstruktorlik shablonlari qo‘llaniladi. Har bir shablonning o‘z maqsadi va kamchiliklari bor. Arxitektura shablonlariga misollar:

– Ko‘p qatlamli shablon (Layered pattern). Tizim diagrammada bir-biridan yuqorida ko‘rsatilgan darajalarga bo‘linadi. Har bir daraja faqat uning ostidagi 1-darajani chaqirishi mumkin. Shunday qilib, har bir darajaning rivojlanishi nisbatan mustaqil ravishda amalga oshirilishi mumkin, bu tizimning rivojlanish darajasini oshiradi. Ushbu yondashuvning kamchiliklari tizimning murakkabligi va ish sur’atining pasayishi hisoblanadi.

– Broker shabloni (Broker pattern). Tizimda ko‘p sonli modullar mavjud bo‘lganda, ularning bir-biri bilan bevosita o‘zaro ta’siri juda murakkablashadi. Muammoni hal qilish uchun vositachi (masalan, ma’lumotlar shinasi) joriy qilinadi, u orqali modullar bir-biri bilan aloqa qiladi. Shunday qilib, tizim modullarining o‘zaro ishlashi ortadi. Barcha kamchiliklar vositachining mavjudligidan kelib chiqadi: u ish faoliyatini pasaytiradi, unga kirish imkoni bo‘lmasligi butun tizimga kirish imkoni bo‘lmasligini keltirib chiqarishi mumkin, u tashqi hujumlar uchun qulay ob’ekt bo‘lishi mumkin.

– “Model - ko‘rinish - nazoratchi” shabloni (Model – View - Controller pattern). Interfeysga qo‘yiladigan talablar tez-tez o‘zgarganligi sababli, ma’lumotlar bilan to‘g’ri munosabatda bo‘lish (o‘qish, saqlash) uchun uni tez-tez o‘zgartirish zarurati tug’iladi. Buning uchun Model-View-Controller (MVC) shablonida interfeys ma’lumotlardan ajratiladi. Bu sizga interfeyslarni o‘zgartirish, shuningdek, ularning turli versiyalarini yaratish imkonini beradi. MVC-da tizim quyidagilarga bo‘linadi:

a) Ma’lumotlarni saqlash modeli
b) Ma’lumotlar qismini ko‘rsatadigan va foydalanuvchi bilan o‘zaro aloqada bo‘lgan ko‘rinish
c) Ko‘rinishlar va model o‘rtasida vositachi bo‘lgan nazoratchi

Biroq, MVC kontseptsiyasi ham o‘zining kamchiliklariga ega. Xususan, o‘zaro ta’sirning murakkabligi tufayli tizimning tezligi pasayadi.

– Mijoz-server shablon (Client-Server pattern). Agar ko‘p sonli iste’molchilar tomonidan cheklangan foydalanishni talab qiladigan cheklangan miqdordagi resurslar mavjud bo‘lsa, u holda mijoz-server arxitekturasini amalga oshirish qulay. Ushbu yondashuv tizimning kengaytirilishi va kirish imkoniyati mavjudligini oshiradi. Ammo shu bilan birga, server tizimda to‘siq bo‘lib qolishi mumkin, agar u mavjud bo‘lmasa, butun tizim ishlamay qoladi.

10.2. Dastur arxitekturasining mezonlari

Kichkina, ammo real va o‘sib borayotgan loyihani yozish vazifasini o‘z zimmamizga olib, dasturning nafaqat yaxshi ishlashi, balki yaxshi tashkil etilganligi qanchalik muhimligini o‘zimizning ishimizda ko‘rdik. Yaxshi o‘ylangan arxitektura faqat yirik loyihalar uchun kerakligiga

ishonmang (faqat yirik loyihalar uchun arxitekturaning mavjud bo‘lmasligi uning yo‘q bo‘lib ketishiga olib keladi).

Murakkablik dastur hajmiga qaraganda tezroq o‘sadi. Agar siz buni oldindan hisobga olmasangiz, uni boshqarishni to‘xtatadigan bir lahza juda tez keladi. To‘g’ri arxitektura ko‘p kuch, vaqt va pulni tejaydi. Ko‘pincha bu sizning loyihangiz saqlanib qoladimi yoki yo‘qligini aniqlaydi. Hatto oddiy "taburetkani qurish" bo‘lsa ham, boshida uni loyihalash juda foydali.

Yaxshi arxitektura uchun mezonlar

Umuman olganda, "dasturiy ta’minot arxitekturasi" umumiy qabul qilingan atama yo‘q. Biroq, amaliyotga kelganda, ko‘pchilik ishlab chiquvchilar uchun qaysi kod yaxshi va nima yomon ekanligi allaqachon aniq. Yaxshi arxitektura, birinchi navbatda, dasturni ishlab chiqish va unga xizmat ko‘rsatish jarayonini sodda va samaraliroq qiladigan foydali arxitekturadir. Yaxshi arxitekturali dasturni kengaytirish va o‘zgartirish, shuningdek, sinab ko‘rish, hatolarni to‘g’rilash va tushunish osonroq. Ya’ni, aslida juda oqilona va universal mezonlar ro‘yxatini shakllantirish mumkin:

Tizim samaradorligi. Dastur, albatta, eng avvalo, o‘z oldiga qo‘yilgan vazifalarni hal qilishi va o‘z vazifalarini har xil sharoitlarda yaxshi bajarishi kerak. Bunga ishonchlilik, xavfsizlik, ishlash, ortib borayotgan yukni engish qobiliyati (miqyoslilik) va boshqalar kabi xususiyatlar kiradi.

Tizimning moslashuvchanligi. Har qanday dastur vaqt o‘tishi bilan o‘zgarishi kerak - talablar o‘zgaradi, yangilari qo‘shiladi. Mavjud funksionallikka o‘zgartirishlar kiritish qanchalik tez va qulay bo‘lsa, u sabab bo‘ladigan muammolar va xatolar qanchalik kam bo‘lsa, tizim shunchalik moslashuvchan va raqobatbardosh bo‘ladi. Shuning uchun, ishlab chiqish jarayonida, keyinroq uni qanday o‘zgartirishingiz kerakligi nuqtai nazaridan olingan natijalarni baholashga harakat qilinadi. O‘zingizdan so‘rang: "Agar joriy arxitektura qarori noto‘g’ri bo‘lib chiqsa nima bo‘ladi?", "Bu holatda qancha kod o‘zgartiriladi?". Tizimning bir qismini o‘zgartirish uning boshqa qismlariga ta’sir qilmasligi kerak. Mumkin bo‘lgan hollarda, arxitektura qarorlari o‘zgaruvchan bo‘lishi kerak va me’moriy xatolarning oqibatlari oqilona cheklangan bo‘lishi kerak. "Yaxshi arxitektura sizga asosiy qarorlarni keyinga qo‘yishga imkon beradi" (Bob Martin) va xatolarning "xarajatlarini" minimallashtiradi.

Tizimning kengaytirilishi. Tizimga uning asosiy tuzilishini buzmasdan yangi ob’ektlar va funktsiyalarni qo‘shish imkoniyati. Dastlabki bosqichda tizimga faqat asosiy va eng zarur funksionallikni kiritish mantiqan to‘g’ri keladi (Yagni printsipi - you ain’t gonna need it — bu sizga kerak bo‘lmaydi, “Sizga kerak emas”) Lekin shu bilan birga, arxitektura kerak bo‘lganda qo‘shimcha funksiyalarni osongina oshirish imkonini berishi kerak. Shunday qilib, eng mumkin bo‘lgan o‘zgarishlarni kiritish uchun eng kam harakat talab etiladi.

Tizim arxitekturasining moslashuvchan va kengaytiriladigan (ya’ni o‘zgarish va evolyutsiyaga qodir) bo‘lishi talabi shunchalik muhimki, u hatto alohida tamoyil (printsip) sifatida shakllantirilgan - Ochiq-yopiq tamoyil – (Open-Closed Principle - beshta SOLID tamoyilining ikkinchisi): Dastur ob’ektlari (sinflar, modullar, funktsiyalar va boshqalar) kengaytirish uchun ochiq, lekin o‘zgartirish uchun yopiq bo‘lishi kerak. Boshqacha qilib aytganda: tizimning mavjud qismlarini o‘zgartirmasdan/qayta yozmasdan tizimni kengaytirish/o‘zgartirish imkoniyati bo‘lishi kerak.

Bu shuni anglatadiki, dastur shunday ishlab chiqilishi kerakki, uning xatti-harakatini o‘zgartirish va yangi funksiyalarni qo‘shish mavjud kodni o‘zgartirmasdan, yangi kod (kengaytmalar) yozish orqali amalga oshiriladi. Bunday holda, yangi talablarning paydo bo‘lishi mavjud mantiqni o‘zgartirishga olib kelmaydi, lekin birinchi navbatda uni kengaytirish orqali amalga oshirilishi mumkin. Aynan shu tamoyil "plugin arxitekturasi" (Plugin Architecture) ning asosi hisoblanadi.

Rivojlanish jarayonining kengayishi. Loyihaga yangi odamlarni qo‘shish orqali rivojlanish vaqtini qisqartirish qobiliyati. Arxitektura dastur ustida bir vaqtning o‘zida ko‘p odamlar ishlashi uchun rivojlanish jarayonini parallellashtirishga imkon berishi kerak.

Sinovga yaroqlilik. Tekshirish osonroq bo‘lgan kod kamroq xatolarni o‘z ichiga oladi va ishonchliroq bo‘ladi. Ammo testlar kod sifatini yaxshilashdan ko‘proq narsani amalga oshiradi. Ko‘pgina ishlab chiquvchilar "yaxshi sinovdan o‘tish" talabi avtomatik ravishda yaxshi loyihaga olib keladigan yetakchi kuch va shu bilan birga uning sifatini baholashning eng muhim mezonlaridan biri degan xulosaga kelishadi: Yaxshi sinf dizaynining "lakmus qog’ozi" sifatida "sinovga layoqatliligi" tamoyilidan foydalanish kerak. Agar siz bir qator test kodini yozmasangiz ham, bu savolga 90% javob berish, loyihadagi hamma narsa qanchalik “yaxshi” yoki “yomon” ekanligini tushunishga yordam beradi.

Testlarga asoslangan dasturlarni ishlab chiqishning butun metodologiyasi mavjud bo‘lib, u Test-Driven Development (TDD) deb ataladi.

Qayta foydalanish imkoniyati. Tizimni uning qismlari boshqa tizimlarda qayta ishlatilishi mumkin bo‘lishi uchun loyihalash maqsadga muvofiqdir.

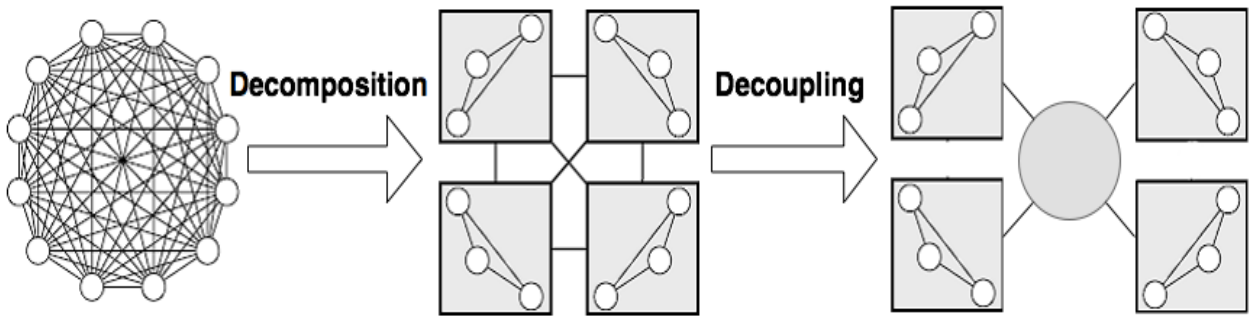
Yaxshi tuzilgan, o‘qiladigan va tushunarli kod. Xizmat ko‘rsatish qobiliyati. Qoidaga ko‘ra, dasturda ko‘p odamlar ishlaydi - ba’zilari ketadi, yangilari keladi. Dasturni yozgandan so‘ng, qoida tariqasida, dasturni ishlab chiqishda ishtirok etmagan odamlarga hamrohlik qilish kerak. Shuning uchun, yaxshi arxitektura uchun yangi odamlar tizimni nisbatan oson va tez tushunishi kerak. Loyiha yaxshi tuzilgan bo‘lishi kerak, takrorlashni o‘z ichiga olmagan, yaxshi shakllangan kodga va afzalroq hujjatlarga ega bo‘lishi kerak. Va iloji bo‘lsa, tizimdagi dasturchilarga tanish bo‘lgan standart, umumiy qabul qilingan yechimlardan foydalanish yaxshiroqdir. Tizim qanchalik ekzotik bo‘lsa, boshqalar uchun tushunish shunchalik qiyin bo‘ladi (*Principle of least astonishment* - Eng kam hayratlanish tamoyili. Odatda foydalanuvchi interfeysiga nisbatan qo‘llaniladi, lekin kod yozish uchun ham qo‘llaniladi). **Yomon loyiha mezonlari**:

1. O‘zgartirish qiyin, chunki har qanday o‘zgarish tizimning juda ko‘p boshqa qismlariga ta’sir qiladi. (**Rigidity** - qattiqlik).
2. O‘zgartirishlar kiritilganda, tizimning boshqa qismlari to‘satdan buziladi. (**Fragility** - mo‘rtlik).
3. Kodni boshqa ilovada qayta ishlatish qiyin, chunki uni joriy ilovadan “ajratish” juda qiyin. (**Immobility** - harakatsizlik).

10.3. Modulli arxitektura

Mezonlarning xilma-xilligiga qaramasdan, yirik tizimlarni ishlab chiqishda asosiy vazifa murakkablikni kamaytirish vazifasidir. Va murakkablikni kamaytirish uchun qismlarga bo‘linishdan boshqa hech narsa ixtiro qilinmagan. Buni ba’zan "bo‘laklarga ajrat va hukmronlik qil" (divide et impera) tamoyili deb ham atashadi, lekin mohiyatiga ko‘ra bu ierarxik parchalanishdir. Murakkab tizimni har biri o‘z navbatida kichikroq qismlardan qurilgan oz sonli oddiy quyi tizimlardan qurish kerak. Bunda eng kichik qismlar to‘g’ridan-to‘g’ri tushunish va yaratish uchun yetarlicha sodda bo‘lgunga qadar jarayon davom ettiriladi(10.1-rasm). Buning yaxshi tomoni shundaki, bu yechim nafaqat ma’lum, balki

universaldir. Murakkablikni kamaytirishdan tashqari, u bir vaqtning o‘zida muhim qismlarni takrorlash orqali tizimning moslashuvchanligini, yaxshi kengayish imkoniyatini va mustahkamligini oshiradi.



10.1-rasm. Modulli arxitektura[23].

Shunga ko‘ra, dastur arxitekturasini qurish, uning strukturasini yaratish haqida gap ketganda, bu asosan dasturni quyi tizimlarga (funktsional modullar, xizmatlar, qatlamlar, kichik dasturlar) parchalanishi va ularning bir-biri bilan va tashqi dunyo bilan o‘zaro ta’sirini tashkil etishni anglatadi. Bundan tashqari, quyi tizimlar qanchalik mustaqil bo‘lsa, ularning har birini ma’lum bir vaqtning o‘zida alohida rivojlantirishga e’tibor qaratish va shu bilan birga barcha boshqa qismlar haqida qayg’urmaslik shunchalik xavfsizroq bo‘ladi. Bunday holda, "spagetti kodi" dan dastur aniq belgilangan va oddiy qoidalarga muvofiq bir-biri bilan o‘zaro ta’sir qiluvchi modullar/kichik dasturlar to‘plamidan iborat bo‘lgan konstruktsiyaga aylanadi, bu aslida uning murakkabligini boshqarishga imkon beradi, shuningdek uni amalga oshiradi. Odatda yaxshi arxitektura kontseptsiyasi bilan bog’liq bo‘lgan barcha imtiyozlarni olish mumkin:

– **Masshtablilik (Scalability)** tizimni kengaytirish va yangi modullarni qo‘shish orqali uning ish faoliyatini oshirish imkoniyati.

– **Ta’mirlash imkoniyati (Maintainability)** bitta modulni o‘zgartirish boshqa modullarni o‘zgartirishni talab qilmaydi.

– **Modullarni almashtirish imkoniyati** (**Swappability**) modulni boshqasi bilan almashtirish oson.

– **Sinov imkoniyati (Unit Testing)** modul barcha boshqalardan uzilishi va sinovdan o‘tkazilishi / ta’mirlanishi mumkin.

– **Qayta foydalanish imkoniyati (Reusability)** modul boshqa dasturlarda va boshqa muhitlarda qayta ishlatilishi mumkin.

[23] С.В. Назаров. Архитектура и проектирование программных систем: Монография - М.: НИЦ Инфра-М, 2013.

– **Texnik xizmat koʻrsatish (Maintenance)** modullarga boʻlingan dasturni tushunish va saqlash osonroq.

Aytish mumkinki, murakkab masalani oddiy qismlarga ajratish barcha loyihalash texnikasining maqsadi hisoblanadi. Va "arxitektura" atamasi, aksariyat hollarda, oddiygina bunday boʻlinish natijasini, shuningdek, "qabul qilingandan keyin oʻzgartirish qiyin boʻlgan baʼzi konstruktiv qarorlarni" anglatadi (Martin Fowler "Korxona dasturiy taʼminot ilovalari arxitekturasi"). Shuning uchun u yoki bu shakldagi koʻpgina taʼriflar quyidagilarga toʻgʻri keladi:

– "Arxitektura tizimning asosiy tarkibiy qismlarini va ularning oʻzaro taʼsirini aniqlaydi. Shuningdek, u fundamental deb talqin qilinadigan va kelajakda oʻzgartirilmaydigan qarorlarni tanlashdir."

– “Arxitektura - bu tizimni tashkil etish, uning tarkibiy qismlari, ularning bir-biri va atrof-muhit bilan aloqasi. Tizim - bu muayyan funktsiyani bajarish uchun birlashtirilgan komponentlar toʻplamidir."

Yaxshi arxitektura, birinchi navbatda, modulli blokli arxitekturadir. Yaxshi arxitekturani olish uchun siz tizimni qanday qilib toʻgʻri dekompozitsiya qilishni (parchalashni) bilishingiz kerak. Shunday qilib, tushunish kerak - qanday dekompozitsiya qilish "toʻgʻri" deb hisoblanadi va uni qanday amalga oshirish yaxshiroq?

10-bob boʻyicha xulosalar

Dasturiy taʼminot arxitekturasini (SW) hujjatlashtirish ishlab chiquvchilar oʻrtasidagi aloqa jarayonini soddalashtiradi, qabul qilingan loyiha qarorlarini qayd etish va tizimni ishlatuvchi xodimlariga ular haqida maʼlumot berish, komponentlar va loyiha shablonlarini boshqalarda qayta ishlatish imkonini beradi. Dasturiy taʼminot arxitekturasi sohasining asosiy gʻoyasi - bu mavhumlik va vakolatlarni ajratish orqali tizimning murakkabligini kamaytirish gʻoyasi. Bugungi kunga qadar "dasturiy taʼminot arxitekturasi" atamasining aniq taʼrifi boʻyicha kelishuv mavjud emas. Dasturiy taʼminot arxitekturasini tavsiflash uchun bir nechta tillar ishlab chiqilgan boʻlsa-da, hozircha qaysi qarashlar toʻplamini maʼlumotnoma sifatida qabul qilish kerakligi haqida kelishuv mavjud emas.

Murakkablik dastur hajmiga qaraganda tezroq oʻsadi. Agar siz buni oldindan hisobga olmasangiz, uni boshqarishni toʻxtatadigan bir lahza juda tez keladi. Toʻgʻri arxitektura koʻp kuch, vaqt va pulni tejaydi. Umuman olganda, "dasturiy taʼminot arxitekturasi" umumiy qabul qilingan atama yoʻq. Biroq, amaliyotga kelganda, koʻpchilik ishlab

chiquvchilar uchun qaysi kod yaxshi va nima yomon ekanligi allaqachon aniq. Yaxshi arxitektura, birinchi navbatda, dasturni ishlab chiqish va unga xizmat ko‘rsatish jarayonini sodda va samaraliroq qiladigan foydali arxitekturadir. Yaxshi arxitekturali dasturni kengaytirish va o‘zgartirish, shuningdek, sinab ko‘rish, hatolarni to‘g’rilash va tushunish osonroq.

Qoidaga ko‘ra, dasturda ko‘p odamlar ishlaydi - ba’zilari ketadi, yangilari keladi. Dasturni yozgandan so‘ng, qoida tariqasida, dasturni ishlab chiqishda ishtirok etmagan odamlarga hamrohlik qilish kerak. Shuning uchun, yaxshi arxitektura uchun yangi odamlar tizimni nisbatan oson va tez tushunishi kerak. Loyiha yaxshi tuzilgan bo‘lishi kerak, takrorlashni o‘z ichiga olmagan, yaxshi shakllangan kodga va afzalroq hujjatlarga ega bo‘lishi kerak. Va iloji bo‘lsa, tizimdagi dasturchilarga tanish bo‘lgan standart, umumiy qabul qilingan yechimlardan foydalanish yaxshiroqdir. Shunday qilib, yaxshi arxitektura, birinchi navbatda, modulli yoki blokli arxitekturadir. Yaxshi arxitekturani olish uchun siz tizimni qanday qilib to‘g’ri dekompozitsiya qilishni (parchalashni) bilishingiz kerak.

10-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minot arxitekturasi deganda nimani tushunasiz.
2. Arxitektura ko‘rinishi necha komponentdan iborat?
3. Arxitektura ko‘rinishlarini necha asosiy turga bo‘linadi?
4. Modulli ko‘rinishlarga misollar keltiring.
5. Joylashish turlariga misollar keltiring.
6. Arxitektura shablonlariga misollar keltiring.
7. Dastur arxitekturasining mezonlari
8. Modulli arxitektura deganda nimani tushunasiz.
9. Masshtablilik (Scalability) deganda nimani tushunasiz.
10. Modullarni almashtirish imkoniyati (Swappability)

11-BOB. DASTURLARNI KONSTRUKSIYALASHNING AMALIY JIHATLARI

11.1. Dasturiy ta’minot o‘lchovlari

Vaqti-vaqti bilan qiziqish paydo bo‘ladigan va yo‘qoladigan dasturlash mavzularidan biri bu dasturiy kod ko‘rsatkichlari masalasidir. Katta dasturiy ta’minot muhitida vaqti-vaqti bilan turli o‘lchovlarni hisoblash mexanizmlari paydo bo‘ladi. Mavzuga bo‘lgan qiziqish shundan iboratki, ular bilan nima qilish kerakligini hali aniqlanmagan.

Ya’ni, ba’zi bir instrumentlar ba’zi o‘lchovlarni yaxshi hisoblash imkonini beradigan bo‘lsa ham, keyin nima qilish kerakligi ko‘pincha noaniq bo‘lib qoldmoqda. Albatta, ko‘rsatkichlar kod sifatini nazorat qilish, dasturchilarning "ishlashi" (tirnoq belgilarida) va loyihani ishlab chiqish tezligi.

Umuman olganda, o‘lchovlardan foydalanish loyiha menejerlari va korxonalarga ishlab chiqilgan yoki hatto ishlab chiqilayotgan loyihaning murakkabligini o‘rganish, ish hajmini, ishlab chiqilgan dastur uslubini va har bir ishlab chiquvchining ma’lum bir yechimni amalga oshirish uchun sarflagan sa’y-harakatlarini baholash imkonini beradi. Biroq, o‘lchovlar faqat maslahat xarakterida bo‘lib xizmat qilishi mumkin, ularni to‘liq qo‘llab bo‘lmaydi, chunki dasturchilar dasturiy ta’minotni ishlab chiqishda o‘z dasturi uchun u yoki bu o‘lchovni minimallashtirish yoki maksimal darajada oshirishga harakat qilib, dastur samaradorligini pasaytirishgacha bo‘lgan usullarga murojaat qilishlari mumkin. Bundan tashqari, agar, masalan, dasturchi oz sonli kod satrlarini yozgan bo‘lsa yoki oz sonli tarkibiy o‘zgarishlarni amalga oshirgan bo‘lsa, bu uning hech narsa qilmaganligini anglatmaydi, lekin bu dastur nuqsoni juda qiyin bo‘lganligini anglatishi mumkin. Biroq, oxirgi muammoni qisman murakkablik ko‘rsatkichlari yordamida hal qilish mumkin, chunki murakkabroq dasturda xatoni topish qiyinroq.

Dasturiy ta’minot ko‘rsatkichi - bu dasturiy ta’minotning ba’zi xususiyatlari yoki uning texnik xususiyatlarining raqamli qiymatini olish imkonini beruvchi o‘lchov. Miqdoriy usullar boshqa sohalarda yaxshi ishlaganligi sababli, ko‘plab kompyuter olimlari va amaliyotchilar ushbu yondashuvni dasturiy ta’minotni ishlab chiqishga o‘tkazishga harakat qilishdi. Tom DeMarko aytganidek, "siz o‘lchay olmaydigan narsani nazorat qila olmaysiz". Amaldagi o‘lchovlar to‘plamiga quyidagilar kiradi:

– o‘sish tartibi (asimptotik tahlil va O-notatsiya nuqtai nazaridan algoritmlarni tahlil qilishni anglatadi),

– kod satrlari soni,

– siklomatik murakkablik,

– funktsiya nuqtalarini tahlil qilish,

– kodning 1000 satridagi xatolar soni,

– test orqali kodni qamrab olish darajasi,

– talablarni qoplash ,

– sinflar va interfeyslar soni,

– Robert Sesil Martin tomonidan ishlab chiqilgan dasturiy paket o‘lchovlari,

– bog’liqlik.

Avvalo, dasturlarning dastlabki kodining miqdoriy xususiyatlarini hisobga olish kerak (ularning soddaligini hisobga olgan holda). Eng elementar ko‘rsatkich bu kod satrlari soni (SLOC). Ushbu ko‘rsatkich dastlab loyihaning mehnat xarajatlarini baholash uchun ishlab chiqilgan. Biroq, bir xil funktsiyani bir nechta satrlarga bo‘lish yoki bitta satrda yozish mumkinligi sababli, bir qatorda bir nechta buyruqlar yozilishi mumkin bo‘lgan tillar paydo bo‘lishi bilan metrikani deyarli qo‘llamaydi. Shuning uchun kodning mantiqiy va jismoniy qatorlari o‘rtasida farqlanadi. Kodning mantiqiy qatorlari dastur ko‘rsatmalari sonidir. Ta’rifning ushbu versiyasi ham o‘zining kamchiliklariga ega, chunki u ishlatiladigan dasturlash tili va dasturlash uslubiga juda bog’liq. Miqdoriy xarakteristikalarga SLOCdan tashqari qo‘shimcha ravishda quyidagilar ham kiradi[24]:

– bo‘sh qatorlar soni;

– izohlar soni;

– izohlar foizi (izohlarni o‘z ichiga olgan satrlar sonining umumiy satrlar soniga nisbati, foizda ifodalangan);

– funksiyalar (sinflar, fayllar) uchun satrlarning o‘rtacha soni;

– funktsiyalar (sinflar, fayllar) uchun manba kodini o‘z ichiga olgan o‘rtacha qatorlar soni;

– modullar uchun o‘rtacha qatorlar soni.

Ba’zan qo‘shimcha sifatida dasturning uslubini baholash farqlanadi (F). Bu dasturni n ta teng bo‘laklarga bo‘lish va Fi = SIGN (Ncomm.i / Ni - 0,1) formulasi bo‘yicha har bir fragment uchun ballni hisoblashdan iborat, bu yerda Ncomm.i - i-chi fragmentdagi sharhlar soni, Ni - i-chi

[24] https://habr.com/ru/company/intel/blog/106082/

fragmentdagi satr kodlarining umumiy soni. Keyin butun dastur bo‘yicha umumiy ball quyidagicha aniqlanadi:

$$F = SUM(Fi).$$

Shuningdek, dastur kodidagi ba’zi ko‘rsatkichlarni hisoblashga asoslangan o‘lchamlar guruhiga Halsted o‘lchamlari kiradi. Ushbu o‘lchamlar quyidagilarga asoslanadi:

n1 — dasturning takrorlanmas operatorlari soni, jumladan ajratuvchi belgilar, protsedura nomlari va operatsiya belgilari (operatorlar lug’ati);

n2 — dasturning takrorlanmas operandlari soni (operandlar lug’ati);

N1 — dasturdagi operatorlarning umumiy soni;

N2 — dasturdagi operandlarning umumiy soni;

n1’ — dasturning takrorlanmas operatorlarning nazariy soni;

n2’ — dasturning takrorlanmas operandlarning nazariy soni.

Kiritilgan belgilarni hisobga olgan holda, biz quyidagilarni aniqlashimiz mumkin:

n=n1+n2 — dastur lug’ati;

N=N1+N2 — dastur uzunligi;

n’=n1’+n2’ — bu dasturning nazariy lug’ati;

N’= n1*log2(n1) + n2*log2(n2) - dasturning nazariy uzunligi (stilistik jihatdan to‘g’ri bo‘lgan dasturlar uchun N ning N’ dan chetlanishi 10% dan oshmaydi);

V=N*log2n — dastur hajmi;

V’=N’*log2n’ - dasturning nazariy hajmi, bu yerda n* - dasturning nazariy lug’ati;

L=V’/V — dasturlash sifat darajasi, ideal dastur uchun L=1;

L’= (2 n2)/ (n1*N2) — nazariy parametrlarni hisobga olmagan holda faqat real dastur parametrlariga asoslangan dasturlash sifat darajasi;

EC=V/(L’)2 — dasturni tushunishning murakkabligi;

D=1/ L’ - dasturni kodlashning murakkabligi;

y’ = V/ D2 — ifodaning til darajasi;

I=V/D — dasturning axborot mazmuni, bu xususiyat dasturni yaratish uchun aqliy xarajatlarni aniqlash imkonini beradi;

E=N’ * log2(n/L) — dasturni ishlab chiqishda zarur bo‘lgan intellektual harakatni baholash, dasturni yozishda talab qilinadigan elementar qarorlar sonini tavsiflash.

Halsted o‘lchovlaridan foydalanganda, bir xil funktsiyani turli xil qatorlar va operatoralr yoirdamida yozish imkoniyati bilan bog’liq kamchiliklar qisman qoplanadi. Dasturiy ta’minotning miqdoriy

o‘lchovlarining yana bir turi Jilba o‘lchovlaridir. Ular dasturning shartli operatorlar yoki davriy operatorlari bilan to‘ldirilganligiga asoslangan dasturiy ta’minotning murakkabligini ko‘rsatadi. Ushbu ko‘rsatkich, soddaligiga qaramay, dasturni yozish va tushunishning murakkabligini juda yaxshi aks ettiradi va shartli va davriy operatorlarni joylashtirishning maksimal darajasi kabi ko‘rsatkichni qo‘shganda, ushbu ko‘rsatkichning samaradorligi sezilarli darajada oshadi.

11.2. Dasturiy ta’minot oqimining murakkabligi o‘lchovlari

O‘lchovlarning navbatdagi katta sinfi dasturlarning miqdoriy ko‘rsatkichlarga emas, balki dasturning boshqarish grafini tahlil qilishga asoslangan bo‘lib, Dasturiy ta’minotrni boshqarish oqimining murakkablik olchovlari deb ataladi. Biron dastur taqdim etilgan bo‘lsin. Ushbu dastur uchun faqat bitta kirish va bitta chiqishni o‘z ichiga olgan yo‘naltirilgan graf tuziladi. Grafning uchlari dastur kodining faqat ketma-ket hisob-kitoblar mavjud bo‘lgan bo‘limlari bilan bog’liq bo‘lib, bu yerda tarmoqlanish va davriy operatorlari mavjud emas va yoylar blokdan blokga o‘tish va dasturni bajarish shoxlari bilan korrelyatsiya qilinadi. Bu grafni tuzish sharti shundan iboratki, har bir cho‘qqiga boshlang’ich cho‘qqidan, yakuniy cho‘qqiga esa boshqa har qanday cho‘qqidan chiqish mumkin bo‘ladi.

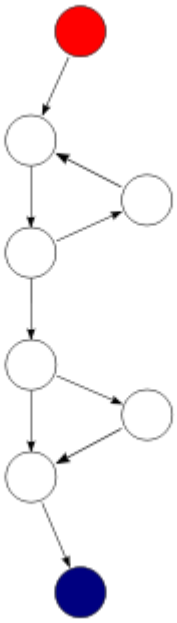
Olingan graf tahliliga asoslangan eng keng tarqalgan baholash bu dasturning siklomatik murakkabligi (Makkeybning siklomatik raqami). U V(G)=e - n + 2p sifatida aniqlanadi, bu yerda e - yoylar soni, n - cho‘qqilar soni, p - bog’langan komponentlar soni. Grafning bog’langan komponentlari sonini grafni kuchli bog’langanga aylantirish uchun qo‘shilishi kerak bo‘lgan yoylar soni deb hisoblash mumkin. Grafning istalgan ikkita uchi o‘zaro bir-biriga yetib borishi mumkin bo‘lsa, u kuchli bog‘langan deb ataladi. To‘g’ri dasturlarning graflari, ya’ni kirish nuqtasi va "osilgan" kirish va chiqish nuqtalaridan yetib bo‘lmaydigan segmentlarga ega bo‘lmagan graflar uchun, odatda, dasturning oxirini yoy bilan belgilovchi cho‘qqini yopish orqali kuchli bog’langan grafik olinadi. ushbu dasturga kirish nuqtasini belgilovchi cho‘qqi. Aslida, V (G) kuchli bog’langan grafdagi chiziqli mustaqil konturlar sonini aniqlaydi. Shunday qilib, yaxshi yozilgan dasturlarda p=1 va shuning uchun siklomatik murakkablikni hisoblash formulasi quyidagicha bo‘ladi:

$$V(G)=e — n + 2.$$

Oddiy dasturning boshqaruv oqimi grafi. Dastur qizil tugunda bajarilishini boshlaydi, so‘ngra tsiklga kiradi (qizil tugunning ostidagi uchta tugundan iborat guruh). Tsikldan chiqishda shartli operator mavjud (tsikl ostidagi guruh) va nihoyat dastur ko‘k tugunda chiqadi. Ushbu grafda 9 ta qirralar, 8 ta tugunlar va 1 ta ulangan komponent mavjud, shuning uchun dasturning siklomatik murakkabligi

9 - 8 + 2 * 1 = 3 ga teng.

Dasturning siklomatik murakkabligi manba kodi chiziqli mustaqil soni yo‘llar uning ichida - "chiziqli mustaqil" degani, har bir yo‘l boshqa yo‘llarning birida bo‘lmagan kamida bitta chekkaga ega. Masalan, agar manba kodida "yo‘q" bo‘lsa boshqaruv oqimlaridagi operatorlar (shartli yoki qaror nuqtalari), murakkabligi 1 ga teng bo‘ladi, chunki kod orqali faqat bitta yo‘l bo‘ladi. Agar kod bitta shartli IF operatoriga ega bo‘lsa, kod orqali ikkita yo‘l bo‘lishi kerak edi: IF ifoda TRUE ga, boshqasi FALSE ga teng bo‘lsa murakkablik 2 ga teng bo‘ladi. Ikkita bir shartli IF yoki ikki shartli bitta IF operatori mavjud bo‘lsa murakkablik 3 ga teng bo‘ladi.



11.1-rasm. Oddiy dasturning boshqaruv oqimi grafi[25].

Afsuski, bu baholash davriy va shartli konstruktsiyalarni ajrata olmaydi. Ushbu yondashuvning yana bir muhim kamchiligi shundaki, bir xil graflar bilan ifodalangan dasturlar murakkabligi bo‘yicha butunlay boshqacha predikatlarga ega bo‘lishi mumkin (predikat kamida bitta o‘zgaruvchini o‘z ichiga olgan mantiqiy ifodadir). Ushbu kamchilikni

[25] Б.В. Черников. Управление качеством программного обеспечения: Учебник - М.: ИД ФОРУМ: ИНФРА-М, 2012.

tuzatish uchun G. Myers yangi usulni ishlab chiqdi. Baho sifatida u intervalni (bu baho oraliq deb ham ataladi) [V(G),V(G)+h] olishni taklif qildi, bunda oddiy predikatlar uchun h nolga teng, n-joyli predikatlar uchun h=n -1 ga teng. Bu usul har xil murakkablikdagi predikatlarni farqlash imkonini beradi, lekin amalda u deyarli ishlatilmaydi.

McCabe usulining yana bir modifikatsiyasi Hansen usulidir. Bu holda dastur murakkabligi o‘lchovi tsiklomatik murakkablik, operatorlar soni juftligi sifatida ifodalanadi. Ushbu usulning afzalligi uning dasturiy ta’minot tuzilishiga nisbatan sezgirligidir. Chenning topologik o‘lchovi dasturning murakkabligini dastur grafi tomonidan tashkil etilgan hududlar orasidagi chegara kesishishlar soni bo‘yicha ifodalaydi. Ushbu yondashuv faqat boshqaruv konstruktsiyalarini ketma-ket ulash imkonini beruvchi strukturali dasturlarga nisbatan qo‘llaniladi. Strukturaviy bo‘lmagan dasturlar uchun Chen o‘lchovi shartli va shartsiz o‘tish operatorlariga bog’liq bo‘ladi. Bunday holda o‘lchovning yuqori va pastki chegaralarini belgilash mumkin. Yuqori chegarasi m + 1, bu yerda m - o‘tish operatorlari o‘zaro ichma-ich joylashtirilgan mantiqiy operatorlar soni. Pastki chegarasi 2 ga teng. Dasturning boshqaruv grafi faqat bitta bog’langan komponentga ega bo‘lsa, Chen o‘lchovi McCabe siklomatik o‘lchoviga mos tushadi.

Dasturning boshqaruv grafini tahlil qilishni davom ettirsak, o‘lchovlarning yana bir kichik guruhini ajratib ko‘rsatish mumkin - Xarrison va Meigel o‘lchovlari. Ushbu usullar ichma-ich joylashish darajasini va dasturning davomiyligini hisobga oladi. Har bir cho‘qqi u tasvirlagan operatorga mos ravishda o‘ziga xos murakkablik bilan belgilanadi. Cho‘qqining ushbu boshlang’ich murakkabligini har qanday usulda, shu jumladan Halsted o‘lchovlari yordamida hisoblash mumkin. Har bir predikat cho‘qqi uchun undan chiqadigan yoylarning uchlari bo‘lgan cho‘qqilar tomonidan hosil qilingan subgrafni belgilaymiz. Shuningdek, har bir bunday cho‘qqidan (pastki chiziqning pastki chegarasi) erishish mumkin bo‘lgan cho‘qqilarni va qandaydir pastki chegarada predikat cho‘qqidan chiqadigan yo‘llarda yotgan cho‘qqilarni ajratib ko‘rsatamiz. Ushbu subgraf predikat tugunining ta’sir doirasi deb ataladi.

Predikat tugunining keltirilgan murakkabligi uning ta’sir doirasiga kiruvchi tugunlarning boshlang’ich yoki keltirilgan murakkabliklari yig’indisi, shuningdek, predikat tugunining o‘zining birlamchi murakkabligi. Dasturning funktsional o‘lchovi (SCOPE) - bu boshqarish grafining barcha cho‘qqilarining keltirilgan murakkabliklarining

yig’indisidir. Funktsional munosabat (SCORT) - bu boshqaruv grafidagi cho‘qqilar sonining uning funktsional murakkabligiga nisbatidir, bunda uchlari sonidan ohirgi tugunllar chiqarib tashlanadi. SCORT bir xil siklomatik raqamga ega graflar uchun turli qiymatlarni qabul qilishi mumkin.

Pivovarskiy metrikasi siklomatik murakkablik o‘lchovining yana bir modifikatsiyasidir. U nafaqat ketma-ket va ichki boshqaruv konstruksiyalari, balki strukturalashtirilgan va strukturalashtirilmagan dasturlar orasidagi farqlarni kuzatish imkonini beradi. U N(G) = v *(G) + sum(Pi) munosabati bilan ifodalanadi, bu yerda v*(G) ifodasi V(G) bilan bir xil tarzda hisoblangan modifikatsiya qilingan siklomatik murakkablikdir, lekin bir farq bilan: n chiqishga ega CASE operatori “n-1” operatorlar to‘plami kabi emas, balki bitta mantiqiy operator sifatida qabul qilinadi. P_i - i-nchi predikat tugunining ichma-ich joylashish chuqurligi. Predikat tugunlarining chuqurligini hisoblash uchun "ta’sir doiralari" soni qo‘llaniladi. Ichma-ich joylashish chuqurligi deganda ko‘rib chiqilayotgan cho‘qqi sferasida to‘liq joylashgan yoki u bilan kesishgan predikatlarning barcha "ta’sir doiralari" soni tushuniladi. Ichma-ich joylashish chuqurligi predikatlarning o‘zlari emas, balki "ta’sir doiralari" ning ichma-ich joylashishi tufayli ortadi. Pivovarskiy o‘lchovi ketma-ket tuzilgan dasturlardan ichma-ich joylashgan dasturlarga o‘tishda va keyinchalik strukturalashtirilmagan dasturlarga o‘tish jarayonida ortib boradi, bu uning ushbu guruhning boshqa ko‘plab o‘lchovlaridan katta afzalligi borligini bildiradi.

Vudvord o‘lchovi - boshqaruv grafi yoylarining kesishishlari soni. Bunday holatlar yaxshi strukturalashtirilgan dasturda yuzaga kelmasligi kerakligi sababli, bu o‘lchov asosan zaif strukturalashtirilgan tillarda qo‘llaniladi. Kesishish nuqtasi boshqaruv ketma-ket operatorlar bo‘lgan ikkita yuqori chegarasidan tashqariga chiqqanda sodir bo‘ladi.

Chegaraviy qiymat usuli ham dasturning bosqaruv grafini tahlil qilishga asoslanadi. Ushbu usulni aniqlash uchun bir nechta qo‘shimcha tushunchalarni kiritish kerak. Yagona boshlang’ch va yagona oxirgi cho‘qqiga ega bo‘lgan dasturning boshqaruv grafi G bo‘lsin. Bu grafda yoylarning kiruvchi cho‘qqilari soni cho‘qqining manfiy darajasi, cho‘qqidan chiquvchi yoylar soni esa cho‘qqining musbat darajasi deyiladi. U holda graf cho‘qqilari to‘plamini ikki guruhga bo‘lish mumkin: musbat darajasi <=1 bo‘lgan cho‘qqilar; musbat darajasi >=2 bo‘lgan cho‘qqilar. Birinchi guruhning cho‘qqilari qabul qiluvchi cho‘qqilar, ikkinchi guruhning cho‘qqilari esa tanlash cho‘qqilari deb

ataladi. 0 ga teng keltirilgan murakkablikka ega bo‘lgan yakuniy cho‘qqidan tashqari har bir qabul qiluvchi cho‘qqi 1 ga teng keltirilgan murakkablikka ega. G dagi barcha cho‘qqilarning keltirilgan murakkabliklari qiymatlari dasturning mutlaq chegara murakkabligini hosil qilish uchun qo‘shiladi. Shundan so‘ng, dasturning nisbiy chegara murakkabligi aniqlanadi:

$$S_0=1-(v-1)/S_a,$$

Bu yerda S_0 - dasturning nisbiy chegara murakkabligi, S_a - dasturning mutlaq chegara murakkabligi, v - dastur grafining umumiy cho‘qqilari soni.

Boshqarish grafida mumkin bo‘lgan yo‘llar soni orqali ifodalangan Schneidewind o‘lchovi mavjud.

11.3. Ma’lumotlar oqimining murakkabligi o‘lchovlari

O‘lchovlarning keyingi sinfi - bu ma’lumotlar oqimining murakkabligi o‘lchovlaridir. Chapin o‘lchovi: usulning mohiyati kirish / chiqish ro‘yxatidagi o‘zgaruvchilardan foydalanishni tahlil qilish orqali yagona dastur modulining axborot quvvatini baholashdan iborat.

Kirish - chiqish ro‘yxatini tashkil etuvchi o‘zgaruvchilarning butun to‘plami 4 funktsional guruhga bo‘lingan:

1. P - hisob-kitoblar va chiqishni ta’minlash uchun kirish o‘zgaruvchilari
2. M – modifikatsiya qilingan yoki dastur ichida yaratilgan o‘zgaruvchilar,
3. C - dastur modulining ishlashini boshqarishda ishtirok etuvchi o‘zgaruvchilar (boshqaruvchi o‘zgaruvchilar),
4. T - dasturda ishlatilmaydigan ("parazit") o‘zgaruvchilar.

Har bir o‘zgaruvchi bir vaqtning o‘zida bir nechta funktsiyalarni bajarishi mumkinligi sababli, uni har bir tegishli funktsional guruhda hisobga olish kerak.

Chapin ko‘rsatkichi:

$$Q = a1*P + a2*M + a3*C + a4*T,$$

bu yerda ***a1, a2, a3, a4*** vazn koeffitsientlari.

Vaznlik koeffitsientlari har bir funktsional guruh dasturining murakkabligiga turli xil ta’sir ko‘rsatish uchun ishlatiladi. O‘lchov muallifining so‘zlariga ko‘ra, C funktsional guruhi eng yuqori 3 ga teng vaznga ega, chunki u dasturni boshqarish oqimiga ta’sir qiladi. Qolgan guruhlarning vazn koeffitsientlari quyidagicha taqsimlanadi:

$a1=1$, $a2=2$, $a4=0,5$.

T guruhining vazn koeffitsienti 0 ga teng emas, chunki "parazit" o‘zgaruvchilar dasturning ma’lumotlar oqimining murakkabligini oshirmaydi, lekin ba’zida uni tushunishni qiyinlashtiradi. Og’irlik koeffitsientlarini hisobga olgan holda:

$$Q = P + 2M + 3C + 0,5T.$$

Span o‘lchovi har bir dastur bo‘limi oralig’ida ma’lumotlarga kirishni mahalliylashtirishga asoslanadi. Span - dastur matnidagi berilgan identifikatorni o‘z ichiga olgan birinchi va oxirgi takrorlanish oralig’idagi bayonotlar (komandalar) soni. Shuning uchun n marta paydo bo‘lgan identifikator n-1 spanga ega. Katta qiymatga ega spanda testlash va dasturning xatolarini aniqlashni qiyinlashadi.

Ma’lumotlar oqimining murakkabligini hisobga oladigan yana bir o‘lchov dasturning murakkabligini global o‘zgaruvchilarga murojaatlar bilan bog’laydigan o‘lchovdir. “Modul-global o‘zgaruvchilar” juftligi (p,r) sifatida belgilanadi, bu yerda “p” - global o‘zgaruvchi “r” ga kirish huquqiga ega bo‘lgan modul. Dasturda “r” o‘zgaruvchiga haqiqiy murojaat mavjudligiga qarab, ikki turdagi "modul - global o‘zgaruvchi" juftlari hosil bo‘ladi: haqiqiy va mumkin bo‘lgan. “r” ga “p” yordamida mumkin bo‘lgan murojaat shuni ko‘rsatadiki, “r” ning mavjudlik sohasi “p” ni o‘z ichiga oladi. Bu xususiyat Aup bilan belgilanadi va Up modullari global o‘zgaruvchilarga murojaat qilish uchun necha marta haqiqatda ruxsat olganliklarini va Pup raqami ular necha marta ruxsat olish mumkinligini bildiradi. Haqiqiy murojaatlar sonining mumkin bo‘lgan murojaatlar soniga nisbati quyidagi ifoda bilan belgilanadi:

Rup = Aup/Pup.

Ushbu formula ixtiyoriy modulni ixtiyoriy global o‘zgaruvchiga havola qilishning taxminiy ehtimolini ko‘rsatadi. Shubhasiz, bu ehtimollik qanchalik yuqori bo‘lsa, har qanday o‘zgaruvchining "ruxsatsiz" o‘zgarishi ehtimoli shunchalik yuqori bo‘ladi, bu dasturni o‘zgartirish bilan bog’liq ishni sezilarli darajada murakkablashtirishi mumkin. Axborot oqimlari kontseptsiyasi asosida Kafur o‘lchovi yaratilgan. Ushbu o‘lchovdan foydalanish uchun lokal va global oqim tushunchalari kiritiladi: A dan B ga ma’lumotlarning lokal oqimi mavjud, agar:

1. A moduli B modulini chaqiradi (to‘g’ri lokal oqim).
2. B moduli A modulini chaqiradi va A moduli B modulida ishlatiladigan qiymatni B ga qaytaradi (bilvosita lokal oqim).

3. C moduli A, B modullarini chaqiradi va A modulining bajarilishi natijasini B ga uzatadi.

Bundan keyin global axborot oqimi tushunchasi berilishi kerak: global ma’lumotlar strukturasi D orqali A dan B ga global axborot oqimi mavjud, agar A moduli D ga ma’lumot joylashtirsa va B moduli D ma’lumotlaridan foydalansa. Ushbu tushunchalarga asoslanib, I - protseduraning axborot murakkabligi qiymati kiritiladi:

$$I = length * (fan\_in * fan\_out)^2.$$

Bu yerda:

– length - protsedura matnining murakkabligi (hajm o‘lchamlaridan biri orqali o‘lchanadi, masalan, Halsted, McCabe, LOC va boshqalar);

– fan_in - protseduraga kiradigan lokal oqimlar soni va protsedura ma’lumot oladigan ma’lumotlar tuzilmalari soni;

– fan_out - protseduradan chiqadigan lokal oqimlar soni, shuningdek protsedura tomonidan yangilanadigan ma’lumotlar tuzilmalari soni.

Modulning axborot murakkabligini uni tashkil etuvchi protseduralarning axborot murakkabligi yig’indisi sifatida aniqlash mumkin. Keyingi qadam, ba’zi ma’lumotlar strukturasiga nisbatan modulning axborot murakkabligini ko‘rib chiqishdir. Ma’lumotlar tuzilimasiga nisbatan modul murakkabligining axborot o‘lchovi:

$$\mathbf{J = W * R + W * RW + RW *R + RW * (RW — 1)}$$

Bu yerda:

W - faqat ma’lumotlar strukturasini yangilaydigan protseduralar soni;

R - faqat ma’lumotlar strukturasidan ma’lumotlarni o‘qish;

RW - ma’lumotlar strukturasidagi ma’lumotlarni o‘qish va yangilash.

Ushbu guruhning yana bir o‘lchami Oviedo o‘lchovidir. Uning mohiyati shundan iboratki, dastur chiziqli kesishmaydigan bo‘limlarga - dasturning boshqaruv grafini tashkil etuvchi operatorlar yoylariga bo‘linadi. Metrika muallifi quyidagi farazlardan kelib chiqadi: dasturchi o‘zgaruvchining aniqlovchi va foydalanuvchi munosabatlar orasidagi bog‘lanishni yoylar orasidagi bog‘lanishga qaraganda osonroq topa oladi; har bir yoyda turli xil kirishlarni aniqlovchi elementlar soni har bir yoyda o‘zgaruvchan elementlardan foydalanishning umumiy sonidan muhimroqdir.

11-bob bo‘yicha xulosalar

Umuman olganda, o‘lchovlardan foydalanish loyiha menejerlari va korxonalarga ishlab chiqilgan yoki hatto ishlab chiqilayotgan loyihaning murakkabligini o‘rganish, ish hajmini, ishlab chiqilgan dastur uslubini va

har bir ishlab chiquvchining ma’lum bir yechimni amalga oshirish uchun sarflagan sa’y-harakatlarini baholash imkonini beradi. Biroq, o‘lchovlar faqat maslahat xarakterida bo‘lib xizmat qilishi mumkin, ularni to‘liq qo‘llab bo‘lmaydi, chunki dasturchilar dasturiy ta’minotni ishlab chiqishda o‘z dasturi uchun u yoki bu o‘lchovni minimallashtirish yoki maksimal darajada oshirishga harakat qilib, dastur samaradorligini pasaytirishgacha bo‘lgan usullarga murojaat qilishlari mumkin.

Dasturiy ta’minot ko‘rsatkichi - bu dasturiy ta’minotning ba’zi xususiyatlari yoki uning texnik xususiyatlarining raqamli qiymatini olish imkonini beruvchi o‘lchov. Miqdoriy usullar boshqa sohalarda yaxshi ishlaganligi sababli, ko‘plab kompyuter olimlari va amaliyotchilar ushbu yondashuvni dasturiy ta’minotni ishlab chiqishga o‘tkazishga harakat qilishdi.

Dasturlarning dastlabki kodining miqdoriy xususiyatlarini hisobga olish kerak. Eng elementar ko‘rsatkich bu kod satrlari soni (SLOC). Ushbu ko‘rsatkich dastlab loyihaning mehnat xarajatlarini baholash uchun ishlab chiqilgan. Kodning mantiqiy qatorlari dastur ko‘rsatmalari sonidir. Ta’rifning ushbu versiyasi ham o‘zining kamchiliklariga ega, chunki u ishlatiladigan dasturlash tili va dasturlash uslubiga juda bog’liq.

O‘lchovlarning katta sinfi dasturlarning miqdoriy ko‘rsatkichlarga emas, balki dasturning boshqarish grafini tahlil qilishga asoslangan bo‘lib, Dasturiy ta’minotrni boshqarish oqimining murakkablik olchovlari deb ataladi.

Chenning topologik o‘lchovi dasturning murakkabligini dastur grafi tomonidan tashkil etilgan hududlar orasidagi chegara kesishishlar soni bo‘yicha ifodalaydi. Ushbu yondashuv faqat boshqaruv konstruktsiyalarini ketma-ket ulash imkonini beruvchi strukturali dasturlarga nisbatan qo‘llaniladi. Strukturaviy bo‘lmagan dasturlar uchun Chen o‘lchovi shartli va shartsiz o‘tish operatorlariga bog’liq bo‘ladi.

Chegaraviy qiymat usuli ham dasturning bosqaruv grafini tahlil qilishga asoslanadi. Ushbu usulni aniqlash uchun bir nechta qo‘shimcha tushunchalarni kiritish kerak. Modulning axborot murakkabligini uni tashkil etuvchi protseduralarning axborot murakkabligi yig’indisi sifatida aniqlash mumkin.

Ushbu guruhning yana bir o‘lchami Oviedo o‘lchovidir. Uning mohiyati shundan iboratki, dastur chiziqli kesishmaydigan bo‘limlarga - dasturning boshqaruv grafini tashkil etuvchi operatorlar yoylariga bo‘linadi.

11-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minot o‘lchovlari deganda nimani tushunasiz.
2. Amaldagi o‘lchovlar to‘plamini aytib bering.
3. Halsted o‘lchamlari haqida nimalar bilasiz?
4. Dasturning siklomatik murakkabligi deganda nimani tushunasiz.
5. Miqdoriy xarakteristikalar iborasi nimani anglatadi.
6. Halsted o‘lchamlari bo‘yicha **n1** nimani anglatadi?
7. Halsted o‘lchamlari bo‘yicha **n2** nimani anglatadi?
8. Halsted o‘lchamlari bo‘yicha **N1** nimani anglatadi?
9. Halsted o‘lchamlari bo‘yicha **N2** nimani anglatadi?
10. Halsted o‘lchamlari bo‘yicha **V** nimani anglatadi?

12-BOB. DASTURIY TA’MINOTNI ISHLAB CHIQISH METODOLOGIYASI

12.1. Metodologiya asoslari

Loyihalashtirish va ishlab chiqarishning asosiy maqsadi rejalashtirilayotgan byudjetda yuqori sifatdagi dasturiy ta’minot ishlab chiqishdan iborat. Bu esa o‘z navbatida DT ni ishlab chiqish muddati va sifati buyturtmachini qoniqtirishi kerak. Bunga esa faqatgina DT ishlab chiqishning to‘g’ri tashkillashtirish orqali erishish mumkin. Quyidagi fikrlarni nazarda tutgan holda, ushbu bir ildizli so‘zlarni ishlatishda “metod”, “metodologiya” va “metodika” bir fikrni aytib o‘tish kerak. **Metodologiya** deb, dasturiy tizimlarni yaratish va umumiy falsafiy birlashtirishda foydalaniladigan mexanizmlar to‘plamiga aytiladi. **Metod** deb, konseptual tushunchalar, bazaviy notatsiya, ushbu tushunchlarning grafik ma’nosi va modellarni qurishni qoidalari, shuningdek loyihalashtirish va ishlab chiqish jarayonini tushunamiz. **Metodika** tushunchasi esa, ma’lum metod asosida qurilidagian dasturiy tizimni qurishning qadamalarining yetarlicha tushunarli berilishini tushunamiz. Metodika odatda bir yoki bir nechta **vositaviy usullarni** ishlatishni ehtimol qiladi.

Shuni anglash mumkinki, UML ga asoslanga har bir metod, dasturlash tiliga qo‘shib faqatgina jarayonlar ta’rifini ham so‘raydi, qaysiki o‘zida metodikalar jamlanmasi birlashtirib, ularning har bir qadami haqida ma’lumot beradi. UML – bu faqatgina til. U yordamida har xil metodlar yaratish mumkin, va bu metodlar bir-biridan ajralib turadi, o‘zlarini grafik notatsiyasiga qaramasdan. Bu yerda dasturlash tillari bilan to‘liq o‘xshashlik mavjud: ikki dasturchi, bittta masalani yechayotgan, bitta dasturlash tilida turli xil dasturlani yozadi. Rational Unified Process(RUP) metodologiyasi o‘zida hozirgi kundagi DT ishlab chqiishning eng yaxshi taraflarini o‘zida mujassam etgan, bularga biznes-modellash, talablarni boshqarish, loyihalashtirishni analiz qilish, KB ishlab chiqish, testlash, konfiguratsiyalarni boshqarish va o‘zgarishlarni boshqarish. Metodologiya — bu prinsiplar tizimi, shuningdek dasturiy ta’minotni ishlab chiqish uslubini aniqlaydigan g’oyalar, tushunchalar, usullar, yo‘llar va vositalar birligi hisoblanadi. Metodologiya — bu standartni ishlatilishi hisoblanadi. Standartlarni o‘zi faqat tanlash va moslashtirish erkinligini qoldirish bilan nima bo‘lishi kerakligini bildiradi. Aniq bir narsalar tanlangan uslubiyat orqali ishlatiladi. Aynan

u ishlab chiqish qanday bajarilishini aniqlaydi. Dasturiy ta’minotni yaratishning ko‘plab muvaffaqiyatli uslubiyatlari mavjud. Aniq bir uslubiyatni tanlash jamoaning (guruhning) hajmiga, loyihaning o‘ziga xosligi va murakkabligiga, kompaniyadagi jarayonlarning barqarorligi va yetilganligiga va xodimlarning shaxsiy sifatlariga bog’liq bo‘ladi. Metodologiya dasturiy ta’minotni ishlab chiqishni boshqarish nazariyasining yadrosi hisoblanadi. Unda ishlatiladigan hayot sikli modeliga (sharshara va iterasion uslubiyatlar) bog’liq ravishda mavjud tasniflashga taxmin qilinadigan va adaptiv uslubiyatga umumiyroq tasniflash qo‘shildi.

Taxmin qilinadigan metodologiya kelajakni atroflicha rejalashtirishga qaratiladi. Loyihaning butun muddatiga rejalashtirilgan masalalar va resurslar ma’lum. Jamoa bo‘lishi mumkin o‘zgarishlarni qiyinchilik bilan qabul qiladi. Reja ishlarning tarkibi va mavjud talablardan kelib chiqish bilan optimallashtirilgan. Talablarning o‘zgarishi rejaning, shuningdek loyiha dizaynining sezilarli o‘zgarishiga olib kelishi mumkin. Ko‘pincha loyihada faqat eng muhim talablar hisobga olinishi uchun “o‘zgarishlarni boshqarish” bo‘yicha maxsus qo‘mita tashkil etiladi.

Adaptiv metodologiya talablarning kutiladigan to‘laqonli emasligi va ularning doimo o‘zgarishini yengib o‘tish maqsadiga qaratilgan. Talablar o‘zgarganida ishlab chiquvchilar jamoasi ham o‘zgaradi. Adaptiv ishlab chiqishda qatnashadigan jamoa loyihaning kelajagini qiyinchilik bilan oldindan aytishi mumkin. Faqat yaqin vaqtga aniq reja mavjud. Uzoqroq vaqtga rejalar faqat loyihaning maqsadlari, kutiladigan harajatlar va natijalar haqidagi deklarasiyalarda mavjud bo‘ladi.

SCRUM uncha katta bo‘lmagan jamoalar (10 tagcha kishili) uchun metodologiya hisoblanadi. Butun loyiha har biri 30 kunlar davomiylikdagi iterasiyalarga (sprintlarga) bo‘linadi. Navbatdagi sprint davomida ishlatilishi rejalashtiriladigan tizimning funksiyalari ro‘yxati tanlanadi. Eng muhim shartlar bitta iterasiyaning bajarilishi vaqtida tanlangan funksiyalarning o‘zgarmasligi va hatto agar relizning chiqariliga barcha rejalashtirilgan funksionalni ishlatishga erishilmasa ham navbatdagi relizni chiqarilishi muddatlariga qat’iy rioya qilish hisoblanadi. Ishlanmaning rahbari scrum deyiladigan har kungi 20 minutlik kengashlarni o‘tkazadi, ularning natijasi oldingi kunda ishlatilgan tizimning ma’lum funksiyalarini, yuzga kelgan qiyinchiliklar va keyingi kunga rejani aniqlash hisoblanadi. Bunday kengashlar

loyihaning borishini doimo kuzatishga, yuzaga kelgan ma’lumotlarni tez aniqlash va ularga operativ harakat qilishga imkon beradi.

KANBAN – masalaga yo‘naltirilgan dasturiy ta’minotni ishlab chiqishni tez moslashuvchan uslubiyati hisoblanadi. Asosiy qoidalari:

– Ishlanmani vizuallashtirish;

– Ishning topshiriqlarga taqsimlanishi;

– Ishlanmada topshiriqning holati haqidagi belgilashlardan foydalanish;

– Ishlanmaning har bir bosqichida bir vaqtda bajariladigan ishlarni cheklash;

– Davriylik vaqtini o‘lchash (bitta topshiriqni bajarilishiga o‘ratacha vaqt) va jarayonni optimallashtirish.

KANBANning afzalliklari:

– Parallel bajariladigan topshiriqlar sonini kamayishi har bir alohida topshiriqning bajarilishi vaqtini sezilarli kamaytiradi;

– Muammoli topshiriqlarni tezkor aniqlash;

– O‘rtachalashtirilgan topshiriqlarni bajarilishi vaqtini aniqlash.

12.2. Rational Unified Process (RUP) metodologiyasi

RUP(Rational Unified Process – Oqilona birlashtirilgan jarayon) Rational Software kompaniyasi tomonidan ishlab chiqilgan dasturiy ta’minotni ishlab chiqish metodologiyasidir. Dasturiy ta’minotni ishlab chiqishda etakchi metodologiyalardan biridir. RUP iterativ va qo‘shimcha rivojlanish modelidan foydalanadi. Har bir iteratsiya davomiyligi (umuman 2 dan 6 xaftaga qadar davom etadigan) loyiha jamoasi ushbu iteratsiya uchun maqsadlarga erishish, dizayn asarlarini yaratishi yoki tozalashi va yakuniy mahsulotning oraliq, ammo funktsional versiyasini olishlari kerak. Iterative rivojlanish siz o‘zgaruvchan talablarga tezda javob berishga, loyihaning dastlabki bosqichlarida xatarlarni aniqlashga va bartaraf etishga hamda yaratilayotgan mahsulot sifatini samarali boshqarishga imkon beradi. RUP quyidagi asosiy printsiplarga asoslanadi:

1) Erta identifikatsiya qilish va uzluksiz (loyihaning oxirigacha) asosiy xavflarni bartaraf etish.

2) Amalga oshiriladigan dastur uchun mijozlar talablarini bajarish uchun sarf-xarajatlar (foydalanish modelining tahlili va qurilishi).

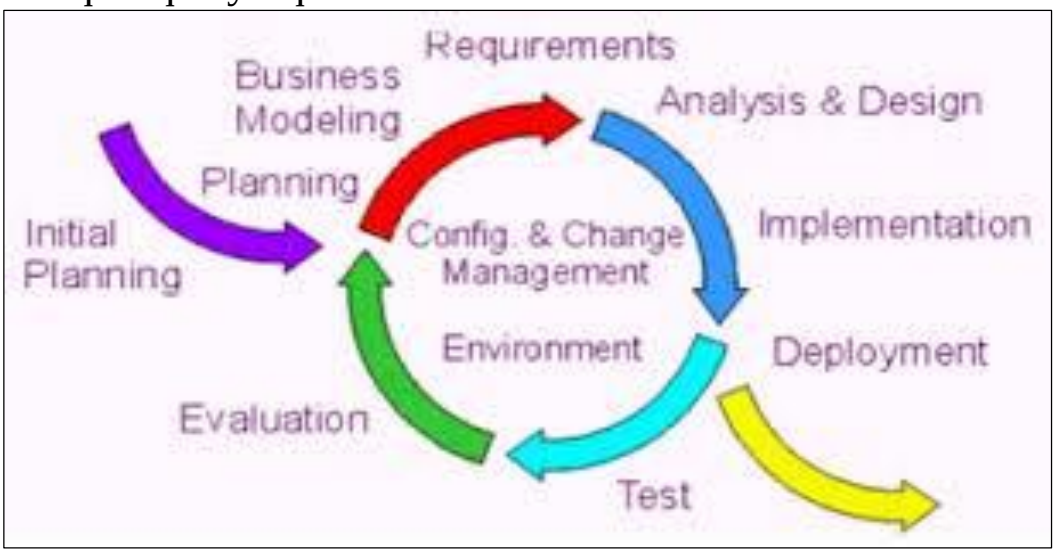
3) Talabalardagi o‘zgarishlarni kutish, dizayn qarorlarini qabul qilish va ishlab chiqish jarayonida amalga oshirish.

4) Komponent arxitekturasi, loyihaning dastlabki bosqichlarida amalga oshirildi va sinovdan o‘tkazildi.

5) Loyihani ishlab chiqishning barcha bosqichlarida doimiy sifatni ta’minlash (mahsulot).

6) Loyihani birgalikdagi guruhda bajarish, asosiy rol me’morlarga tegishli.

RUP texnologiyasidan foydalangan holda amalga oshirilgan loyihaning hayot aylanish jarayonining tuzilishi koordinata tekisligida ko‘rib chiqish qulayroq bo‘ladi.



12.1-rasm. RUP metodologiyasidan foydalanish[26].

RUPda loyihaning hayotiy jarayoni va bosqichlari

Vaqt, loyihaning hayot aylanish jarayonining dinamik jihatini aks ettiradi, bu ikki alohida bosqichni ajratuvchi davrlar, bosqichlar, izlanishlar va nazorat nuqtalari jihatidan ifodalanadi. Vertikal eksa loyihaning statik jihatini aks ettiradi. Bu jarayonlar, eksponatlarni (har qanday jarayon davomida yaratilgan yoki o‘zgartirilgan ma’lumotlarning birligi) va rollarda (jarayondagi jarayonning munosib bajarilishi uchun javobgarlik) tavsiflanadi.

RUP loyihani amalga oshirilishini turli bosqichlarda tashkil etadi, ularning har biri bir yoki bir nechta elementlardan iborat. Iterativ yondashuvda, har bir jarayon uchun ish hajmlari hayot aylanish jarayonida o‘zgarib turadi. Bosqichning oxiridagi nazorat punktlari avvalgi bosqichning qanchalik muvaffaqiyatli ekanligi va loyihaning qanchalik muvaffaqiyatli ekanini baholash imkonini beradi. To‘liq

[26] Кролл П., Крачтен Ф. Rational Unified Process - это легко. Руководство по RUP для практиков. Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2004 г

mahsulotni ishlab chiqarish jarayoni to‘rt bosqichdan iborat bo‘lib, ularning har biri bir yoki bir necha integratsiyani o‘z ichiga oladi:

1) Boshlanishi (Inception)

Ushbu bosqichda:

a) Loyihaning tafakkuri va chegaralari shakllantiriladi.

b) Tadbirkorlik ishi yaratiladi.

c) Mahsulotning asosiy talablari, cheklovlari va asosiy funksiyalari aniqlanadi.

d) Foydalanish holatlari diagrammasining asosiy versiyasi yaratiladi.

e) Xatarlar baholanadi.

f) Dizayn ishlab chiqiladi.

2) Loyihalash (Elaboration)bosqichida domen tahlil qilinadi va bajariladigan arxitektura quriladi. Bunga quyidagilar kiradi:

a) Hujjatlarni talab qilish (ko‘p hollarda foydalanish holatlarini batafsil tavsifi bilan).

b) Amalga oshiriladigan arxitekturani loyihalash, amalga oshirish va sinovdan o‘tkazish.

c) Vaqt va xarajatlarning yangilangan ishi va aniqroq hisoboti.

d) Asosiy xavflarni kamaytirish.

3) Qurilish (Construction)

Ushbu bosqichda mahsulotning ko‘pgina funktsiyalari amalga oshiriladi. O‘zgarishlar Tizimning birinchi tashqi versiyasi bilan qurish yakunlandi.

4) Dastur(Transition)

Dastur bosqichida mahsulotning yakuniy versiyasi ishlab chiqaruvchidan xaridorga etkaziladi. Bunga beta test dasturi, foydalanuvchi treningi va mahsulot sifatini aniqlash kiradi. Agar sifat foydalanuvchilarning talablarini yoki boshlang’ich bosqichda belgilangan mezonlarni qondirmasa, Faza Implementation fazasi yana takrorlanadi.

RUP – bu jarayon bo‘lib, Dasturiy ta’minotni kollektiv ishlab chiqishga qaratilgan bo‘ladi .Barcha qatnashuvchilar umumiy bilimlar bazasini, umumiy jarayon, ishlab chiqishga qaratilgan umumiy qarash, umumiy modellash tizlidan foydalanishadi. RUP UML bilan bir vaqtda OO modellashtirish standarti tomonidan ishlab chiqilgan. RUP ning barcha modellari UMLG notatsiyasida tashkillashtirilgan. RUP – bu DT ishlab chiqishning texnologik jarayon bo‘lib, kollektiv ishlab chiqarishni yaxshilishga qaratilgan va metodikalarning barcha xayot siklini xujjatlar shabloni, instrumental vositalar bilan ishlashda asosiy ishlarni bajarishga qaratadi.

RUP doimiy ravishda umumiy bilimlar bazasi asosida doimiy rivojlanib kelmoqda, internet ordamida to‘ldirishlar kiritilib kelinadi.Bu mashxur vosita bo‘lib, ishlab chiqilayotgan dasturlarning keng spektrida ishlatilishi mumkin.

RUP ning ajralib turuvchi sohalari

– RUP –itterativ jarayon (Controlled Interactive);

– Use Cases larning oraliq apparatlarini ishlatishni ehtimol qiladi

– Asosiy e’tibor arxetikturani ishlab chiqishga qaratiladi (Architecture Centric)

– Talablarni va o‘zgarishni boshqarish (Requirements Configuration and Change Management)

– DT ishlacb chiqishda KB konsepsiyasiga asoslanadi (Component Based Development)

– Visual modelashtirishga e’tibor qaratadi (Visual Modeling Techniques)

Iteratsiyalar

Klassik sharshara hayotiy sikli, o‘zida DT talablarni, loyihalashtirishni,ishlab chiqarishni, yig’ish va testlashni ketma-ket bajarilishini ta’minlaydi. Ushbu usulning asosiy kamchiligi, DT talablariga kirtiladigan har bir o‘zgartirish qimmatga tushadi. Buning asosiy sababi, ushbu metod orqali o‘zgartirish kiritishni aniqlah loyiha oxirida aniqlanadi va o‘zgartirish kiritish uchun katta miqdordagi ishni bajarishga to‘g’ri keladi. Ushbu usul kichik loyihalar uchun qulay, qachonki DT ga bo‘lgan talab qat’iy aniqlangan va o‘zgartirish kiritilmaslikka kafolat bor. Afsuski bunaqa loyihalar hayotda juda kam.

RUP DT ni ishlab chiqishda itterativ usulni taklif qiladi, spiral xayotiy siklga asoslangan. Umumiy hayotiy sikl asosiy 4 fazadan iborat. –loyihaga kirish(o‘rganish), rivojlanish (rejalarni aniqlashtirish), qurish va qayta ishlash. Har bir faza itteratsiyalar ketm-ketligidan iborat, va ularning soni hohlagancha bo‘lishi mumkin. Har bir itteratsiyada DT ning uncha katta bo‘lmagan qismini texnologik ishlab chiqishda qo‘llaniladigan jarayonlardan iborat. Shu bilan birga buyurtmachiga natijalarni ko‘rsatish imkoni mavjud. Bu esa amalga oshirilgan ishni baholash, fikr bildirish va DT ga bo‘lgan talabni aniqlash va o‘zgartirish kiritishga imkon beradi. Bu kabi tashkilot bir qancha ustivorliklarga ega.

– talabga o‘zgartirish va aniqlashtirishlar ishlab chiqishning boshlang’ich bosqichida qachonki kod miqdori uncha katta bo‘lmaganda, o‘zgartirish kiritish unchalik murakkab bo‘lmaganda amalga oshiriladi.

– Boshlang’ich bosqichlarning o‘zida buyurtmachi mutahasislarini boshlang’ich versiya blan tanishtirish imkoni mavjud. Natijada esa oxirgi maxsulot buyurtmachi hohlagan maxsulot bo‘lish ehtimoli va sifatli DT ishlab chiqish ehtimoli yuqori bo‘ladi.

– Arxitekturaviy va integratsion risklar kamayadi. Itterativ usul yordamida o‘rganishlar vaqtida ishlab chiqiladigan mustahkam arxitektura quriladi va keyinchalik bir qancha boshlang’ich itteratsiyalarda tekshiriladi. Har bir itteratsiya tizimga yangi elementlarni kiritishni talab qiladi, ya’ni har bir itteratsiyadagiintegratsiya qilinayotgan elementlar soni unchalik katta emas va DT ning ishlab chiqilgan barcha elementlarini gloab itteratsiyasiga qaraganda osonroq boshqarish bo‘ladi.

– Itterativ usul Dasturiy elementlarni to‘liqroq qayta ishlatishga ko‘proq moslashgan. Har bir itteratsiyani analiz qilish tizim arxitektorlariga qayta ishlashga moyil bo‘lgan qismlarni ajratib olish va keyingi itteratsiyada qayta ishlanayotgan kod sifatida foydalanish mumkin bo‘ladi. Bu xususiyat OY ca KB loyihalashtirish g’oyalari mos keladi.

RUP – ishlatish variantlariga yo‘naltiruvchi jarayon

UML tilidagi aytib o‘tilgan use case tushunchasi RUP da ko‘rib chiqiladigan xayot siklining barcha etaplarini bajarilishida asos bo‘lib xizmat qiladi. “business use case” (faoliyat turi) tushunchasi biznes analitikasida asos deb hisoblanadi. Talablarni analiz qilayotganda, ishlatish variantlari(IV) belgilab biz Dtga bo‘lgan talablarning ierarxik darajalarini yoki talablarning o‘zini belgilaymiz. IV ni detalizatsiya qilish vaqtida obyektlar aniqlanadi, va ular bilan bog’lanish usullari, qaysiki dastur kodida amalga oshirilishi kerak bo‘lgan.

IV ni itteratsiyani rejalashtiridagi rolini aytib o‘tish lozim. Keyingi itteratsiya qaysi funksionallik amalga oshirilishi kerakligini qanday bilish mumkin Bu yerda bizga yordamga IV diagrammalari keladi. Har bi Iv ga uning qaysi bo‘limda amalaga oshirishilishini bildiruvchi vakolat beriladi. Barcha IV larni ularning navbatini ko‘rsatadigan turli diagrammalarda tasvirlash mumkin.

RUP – arxitekturaga asoslangan jarayon

Har bir loyiha asosidaDT ni qurishning asosini belgilab beruvchi kardinal qarorlar yotadi, qaysiki o‘zining aksini **tizim arxitekturasida** topadi. DT arxitetkturasi va RUP ni ishlab chiqish maqsadlari:

– DTing tashkiliy izohi

– Komponentalarni va ularning interfeysini aniqlash

- Tagosti tizimlarni aniqlash va komponentlarni tag osti tizimlarga birlashtirish
- Komponentlarani qayta ishlatish uchn asos yaratish
- Ishlab chiqarish uchun bazis yaratish
- Loyiha ustidan nazorat va tizimning butunligini ta’minlash

DT arxitekturasi turli **arxitekturaviy tushunchalar** shaklida bo‘lishi mumkin.

Manqitiy tasavvur tizimga bo‘lgan talablarni aks ettiradi. U asosiy arxitekturaviy muhim bo‘lgan paketlar, tagosti tizimlar va loyiha klasslarini aniqlaydi.

Amalga oshiruvchanlik tasavvuristatik dasturiy modullarni tashkilotini paketlar va darajalar tushunchasida ta’riflaydi.

Jarayonli tasavvur Tizim ishlash jarayonida jarayonlar, oqim va topshiriqlarning paralelligini va ular o‘rtasidagi bog’lanishlarni ifoadalaydi.

Use case tasavvuri asosiy IV va ssenariylardan tashkil topgan.

Arxitekturaviy tushunchalar asosiy e’tiborini DT ning strukturasiga sezirarli ta’sir ko‘rsatadigan elementlarga qaratadi, bularga ishlab chiqarish samaradorligi, masshtablilik, mustahkamlik, yangilash imkoniyati mavjudliligi, rivojlanish ikmoniyati. Shu kabi elementlari qatoriga:

- Asosiy faoliyat obyektlarini modellashtiruvchi klasslar;
- Ushbu klasslarning harakatini aniqlovchu mexanizmlar;
- Darajalar va tagosti tizimilar;
- Interfeyslar;
- Asosiy jarayonlar va boshqaruvchi oqimlar.

RUP da texnologik jarayonlar

RUP da 9 ta texnologik jarayon aniqlangan, va har biri uchun bajarilish metodikasi belgilangan. Texnologik jarayonlar ikki kategoriyaga bo‘linadi –Asosiy jarayonlar va Yordamchi jarayonlar. Asosiy jarayonlarga:

- Biznes analiz;
- Talablarni boshqarish;
- Analiz va loyihalashtirish;
- Realizatsiya;
- Testlash;
- Rivojlanish.

Yordamchi jarayonlarga :

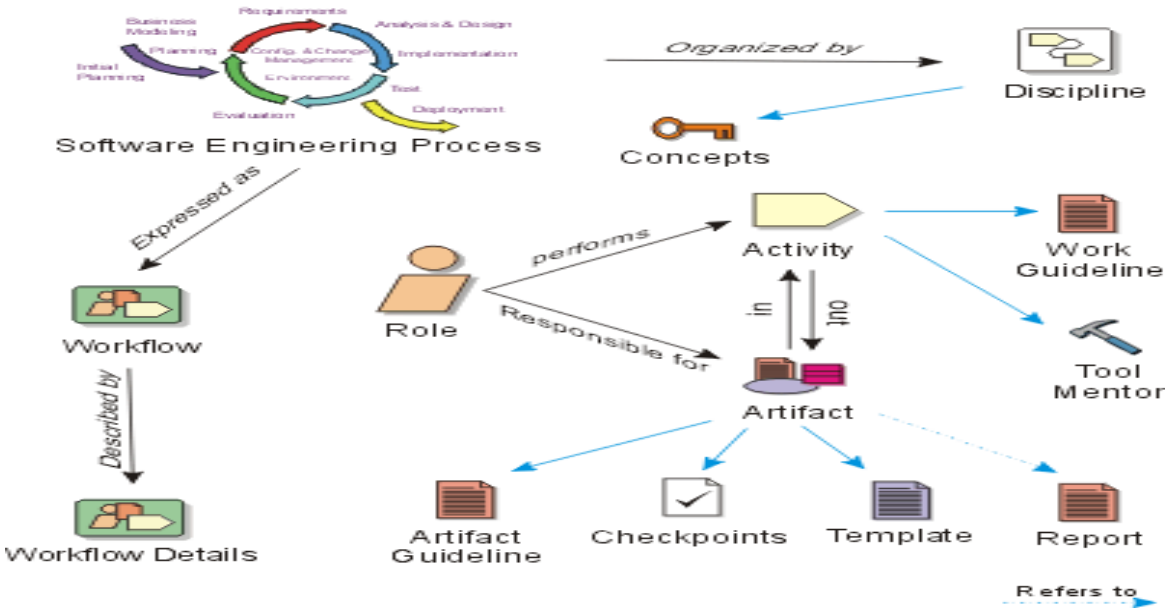
– Loyihani boshqarish;
– Konfiguratsiyani boshqarish;
– Muhitni boshqarish.

Har bir texnologik jarayon uchun **rollar** belgilangan, ajratilgan guruh yoki shaxslarning harakatini ifodalovchi, **faolyat turlari**bajaruvchilar tomonidan bajrilishi lozim bo‘lgan ishlar va jarayon tomonidan paydo bo‘ladigan va modifikatsiya qilinadigan xujjatlar – **artefaktlar.** RUP da asosiy artefaktlar –model, model elementi, xujjat, tashqi kod, bajariluvchi dastur.

EUP Metodologiyasi

SysML tilidan samarali foydalanish, murakkab boshqaruv jarayonlarini tadqiq qilish va tahlil qilish bilan bir qatorda, ayrim jarayonlarni, masalan, elektron boshqaruv tizimida avtomatlashtirilgan qo‘llab-quvvatlashni amalga oshirishda ularning natijalarini amalga oshirishni o‘z ichiga oladi, bu IT sohasida to‘plangan tajribaga asoslangan. Ushbu tajribani qo‘llashning yaqqol namunasi - axborot tizimlarini rivojlantirishning metodologiyasi (Enterprise Single Process).

EUP metodologiyasi ob’ektga asoslangan paradigmdan foydalanadigan dasturiy ta’minot loyihalarini qo‘llab-quvvatlash uchun mo‘ljallangan RUP (Rational Unified Process) taniqli metodologiyasining kengaytmasi hisoblanadi. Uslubiyat hayotning aylanish jarayonidan foydalanishni o‘z ichiga oladi(12.2-rasm).



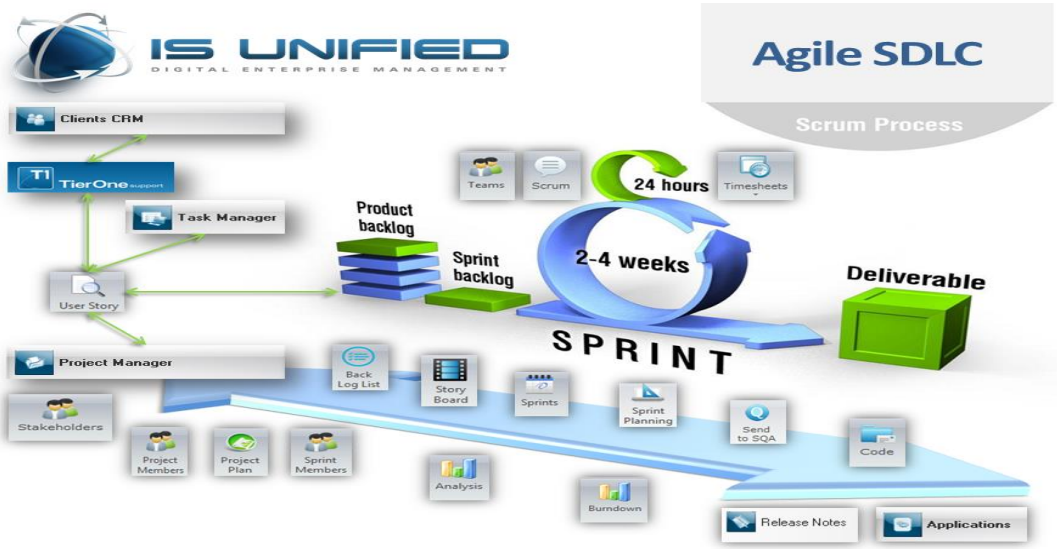
12.2-rasm. EUP metodologiyasidan foydalanish[27].

[27] Кролл П., Крачтен Ф. Rational Unified Process - это легко. Руководство по RUP для практиков. Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2004 г.

12.3. Agile Unified Process metodologiyasi

Agile ishlanmasi (ingliz tilidan - Agile dasturiy ta’minot ishlab chiqish) - bir fikrlash yo‘li va asosiy qadriyatlar va bir necha yondashuvlarni qo‘llamoq tamoyillarini o‘z ichiga belgilaydi. Manifesti dasturiy ta’minot ishlab chiqish uchun (so‘nggi yillarda da bor tendentsiyasi emas, balki faqat axborot texnologiyalari bo‘yicha) ham, interaktiv rivojlantirish, davriy risk ishchi guruhlari o‘z-o‘zini tashkil etish orqali ifodalanadi.

Buyurtmachining talablar va ularni amalga oshirish orqali anglatadi ekspertlar tashkiliy faoliyatining boshqa sohalarida, tezkor rivojlantirish foydalanish uchun harakat esa turli profili (chiquvchilar, sinov, amaliyot, va hokazo). "Agile rivojlantirish" kabi Agile tarjima chunki butunlay to‘g’ri emas odatda Agile metodologiyasi chaqirdi, lekin bu deklaratsiyasi asosida yondashuvlar metodologiyasi, lekin Bilingani jihatidan ular deyiladi emas - ramkalar. va hokazo scrum, haddan tashqari dasturiy, FDD, DSDM: Ayni paytda, bunday kabi epchil rivojlantirish uslubiyati, asoslangan ko‘p ramkalar (uslubiy), yondashuvlar mavjud.

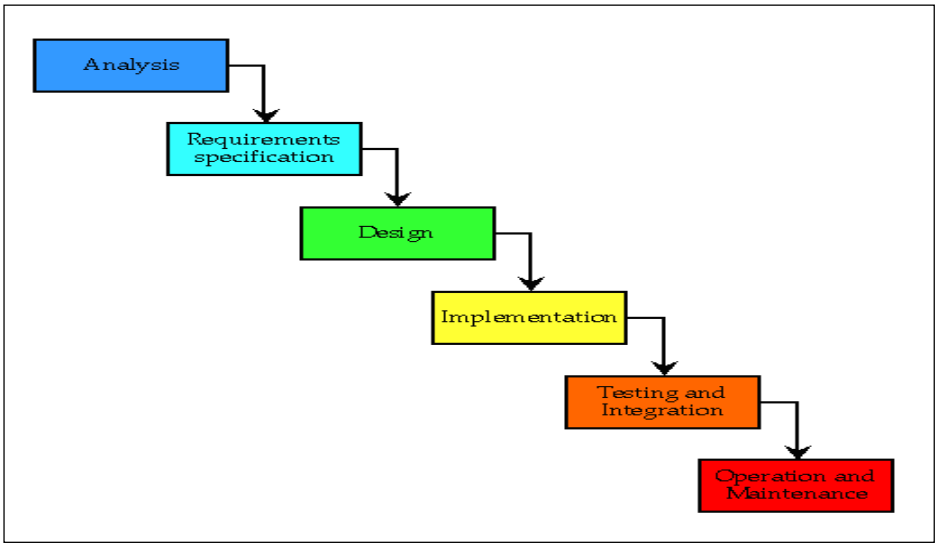


12.3-rasm. Agile metodlogiyasidan foydalanish[28].

Agile - an’anaviy chiziqli usullari "sharshara" farqli o‘laroq metodologiyasi, iteratif va ortib borayotgan dasturiy ta’minot ishlab chiqish, biri. Bilingani metodik dasturi hayot tsikli davomida tizimi dizayn usullari, dizayn va test belgilaydi. Tezkor usullarini (masalan, scrum) adaptiv rejalashtirish, talablarni qo‘shma rivojlantirish, o‘z-o‘zini

[28] Ивутин А.Н., Волошко А.Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.

tashkil ko‘ndalang funktsional rivojlantirish jamoalari rasionalizatorlik, shuningdek aniq muddatlarda bilan navbat asoslangan dasturiy ta’minot ishlab chiqish foydalanish orqali o‘zgartirish javob asoslangan. Bunday yondashuv bugungi tijorat dasturiy ta’minot ishlab chiqish loyihalari ko‘p ishlatiladi.



12.4-rasm. Agile metodlogiyasidan foydalanish[29].

Tezkor rivojlantirish uslubiyatlarini zamirida boshqarish va a’zolari muayyan dasturiy ta’minotni ishlab chiqish bilan band mehnat jamoalari, tashkil bir liberal-demokratik yondashuv hisoblanadi. Shu tufayli gibkiy (epchil) usuli yordamida dasturiy ta’minot ishlab chiqish 2-3 hafta davomiyligi bilan, qisqa ko‘chadan (iteratsiya) bir qator belgilaydi aslida, deb risklarni kamaytirish uchun erishilgan har bir iteratsiya to‘ldirilishi, Mijozlar natijalarini qabul qiladi va yangi yoki tuzatish talablari, ya’ni beradi rivojlanishini nazorat qiladi va darhol unga ta’sir qilishi mumkin. Har bir takrorlash rejalashtirish, talablar tahlil, dizayn, ishlab chiqish, sinov va hujjatlarni qadamlar o‘z ichiga oladi. Odatda, bir takrorlash to‘laqonli dasturiy mahsulotni ozod qilish uchun etarli emas, balki rivojlantirish har bir bosqichi oxirida "moddiy" mahsulotni yoki sinash va qo‘shimcha yoki tuzatish chora-tadbirlar berish ko‘rish mumkin funksional qismini sodir kerak. bajarilgan ishlar asosida, har bir bosqichdan so‘ng, jamoa to‘playdi va dasturiy ta’minot ishlab chiqish rejasiga o‘zgarishlar qilish uchun asos bo‘lgan yangi talablarni umumlashtiradi.

Agile asosiy g’oyalaridan biri, jamoa ichida va mijoz yuzi bilan hamkorlik tezlik qarorlar qabul qilish va dasturiy ta’minot ishlab chiqish

[29] Ивутин А.Н., Волошко А.Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.

xavfini kamaytirish imkonini beradigan, yuz emas, shuning uchun jamoa nuqtai geografik nuqtadan, bir joyda joylashtirilgan. (Mijoz yoki buyurtmachi o‘zi vakili vakolatli, mahsulot talablariga;. A loyiha menejeri roli mijozning yoki ish tahlilchisi tomonidan amalga oshiriladi vakili mahsulot egasi) Bundan tashqari, jamoa mijozlarga bir vakili o‘z ichiga oladi. Agile Manifesto‘da rivojlantirish va bo‘shatish quyidagi uslubiy vakillari ishtirok etadi:

- Adaptive Dasturiy ta’minot ishlanmasi (ASD);
- Crystal Clear;
- Dinamik tizimlari taraqqiyot usuli (DSDM);
- Ekstremal dasturlash (XP);
- Xususiyati gijgijlash ishlanmasi (FDD);
- pragmatik dasturlash;
- scrum.

Tezkor rivojlantirish uslubiyatlarini haqiqiy ma’lumotlar oldindan Deklaratsiyani ozod qilish mavjud. relizlar ochiq-oydin o‘zi chaqqon rivojiga yangi turtki berdi, poydevor qo‘ydi, bir dasturiy ta’minot ishlab chiqish uchun Konstitutsiya moslashuvchan yondashuvni aytish mumkin. Dasturiy ta’minot ishlab chiqish Agile-Manifesti: Asosiy metrik tezkor-usullari ishchi mahsulot hisoblanadi, to‘g’ridan-to‘g’ri muloqot afzal, epchil-usullari boshqa usullar bilan yozilgan hujjatlarga nisbatan miqdorini kamaytirishdir.

Tezkor rivojlantirish uslubiyatlari Qadriyatlar va Agile Manifesto‘da tashkil etildi quyidagi tezkor rivojlantirish uslubiyati tomonidan belgilangan tamoyillarga asoslangan. Agile modellashtirish (AM) - bu yondashuv tubdan dasturiy ta’minot ishlab chiqish qismi sifatida modellashtirish (model kodi tekshirish, shu jumladan) tartib va hujjatlarni belgilaydi. kamroq darajada UML dizayn tartib va diagrammada ta’riflaydi. Bundan tashqari, ishlab chiqish, test, loyiha boshqaruvi, joylashtirish va ularga xizmat ko‘rsatish bilan ta’sir ko‘rsatdi.

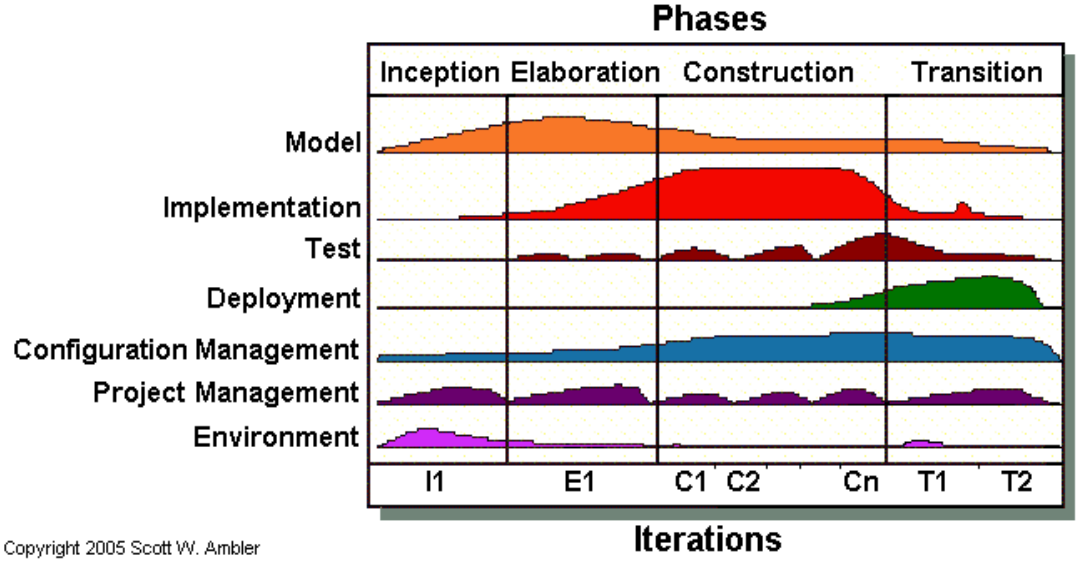
Agile Unified Process (KKS) - Scott Ambler tomonidan tashkil etilgan metodologiyasi RUP (IBM Rational Unified Process), yagona versiyasi. KKS ish ilovalar ichida dasturiy ta’minot yaratish uchun bir model belgilaydi.

Agile Data usuli (ADM) - e’tibor turli ko‘ndalang funktsional jamoalari hamkorlikda orqali talab va yechimlari shakllantirish bo‘yicha joylashtirilgan bo‘lgan tezkor dasturlar ishlab chiqish iterativ usullari, to‘plamidir.

Dinamik tizimlari taraqqiyot usuli (DSDM) - e’tibor bir dasturiy mahsulot ishlab chiqish uchun oxirgi foydalanuvchining maksimal tezlikda joylashtiriladi.

Skram - loyihalarni boshqarish usuli, metodologiyasi bo‘lib, u vaqtni boshqarish tamoyillari asosida qurilgan. Uni boshqa loyihalarni ishlab chiqish jarayonini boshqarish usullardan farqi, takrorlanuvchi jarayon hisoblanadi. Mahsulot yaratish jarayoni bir nechta vaqt oraliqlariga bo‘linadi. Har bir vaqt oralig’i tugagandan so‘ng, loyiha egasiga ya’ni mijozga qandaydir tugallangan mahsulot beriladi. Mijoz bildirgan fikrlar, taklif va e’tirozlar asosida loyiha boshida tuzilgan rejalardagi kamchilik va xatolarni, asosiy maqsad sari to‘g’ri ketilayotganligi haqida qayta o‘ylanadi.

Rational Unified Process (RUP) soddalashtirilgan Agilening versiyasi. Bu oddiy, hali epchil texnikasi va tushunchalarni yordamida ish dasturi dasturiy ta’minot ishlab chiqish RUP uchun haqiqiy qolgan yondashuv tushunish oson. Birinchidan, Model intizom Rup biznes modellashtirish, talablar va tahlil va tarkib fanlarni o‘z ichiga oladi. Ko‘rib turibsizki, model, Agile muhim qismidir, lekin u jarayonini hukmronlik emas - sizga faqat etarlicha zo‘rg’a yaxshi modellar va hujjatlarni yaratish orqali tezkor qolishni istayman. Ikkinchidan, Konfiguratsiya va o‘zgarishlar boshqarish intizom endi Configuration Management intizomga, tezkor rivojlantirish o‘zgarishlar boshqarish faoliyati, odatda, Model intizomini qismidir sizning talablari boshqaruv harakatlari, bir qismidir.



12.6-rasm. Agile Unified Process hayot davriga misol[30].

[30] Ивутин А.Н., Волошко А.Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.

Katta seriyalli takrorlanish

Kengaytmali xarakteri uni tez harakatlanuvchi 4 bosqichda namoyon qiladi.

1. Inception-Boshlash. Maqsad loyiha, sizning tizimi uchun salohiyatli me'moriy boshlang'ich ko'lamini aniqlash va dastlabki loyiha moliyalashtirish va manfaatli tomon qabul qilishdan iborat.

2. Elaboration-ishlab chiqish

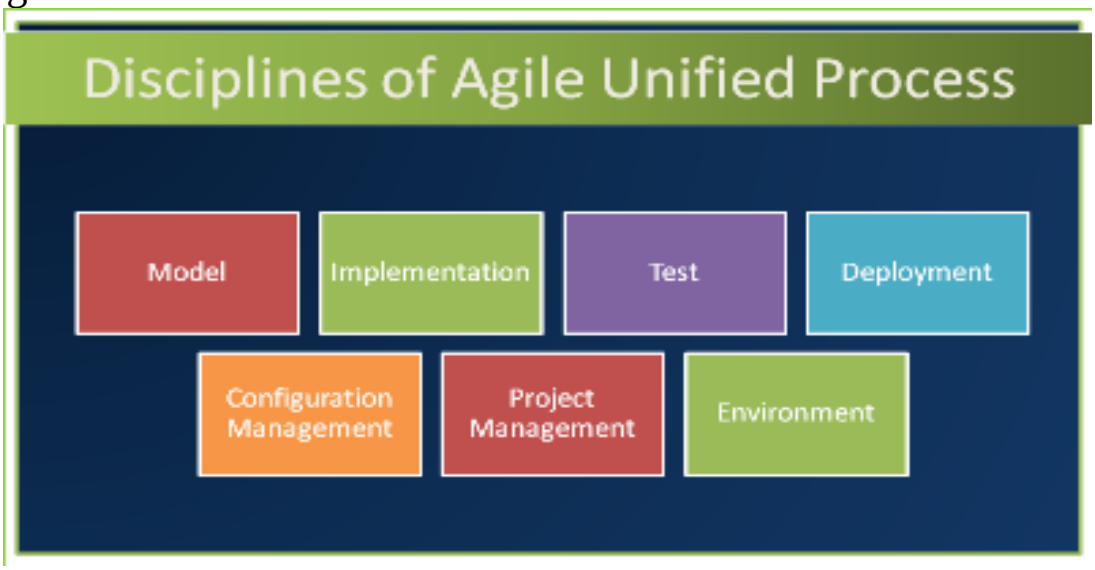
Maqsadi: Tizim arxitekturasini isbotlash.

3. Construction - konstruktsiyalash. maqsad loyiha manfaatdor tomonlar eng ustuvor ehtiyojlarini javob beradigan muntazam, borayotgan asosda ishlovchi dasturiy ta'minot yaratish.

4. Transition. O'tish. maqsad tasdiqlash va ishlab chiqarish muhiti ichiga tizimni tarqatilishi.

5. Kichik serialli takrorlanuvchi

Fanlar rivojlantirish guruhi a'zolari, qurish tasdiqlash va ularning manfaatdor tomonlar ehtiyojlarini javob beradigan ish dasturiy ta'minot yetkazib berish amalga faoliyatini belgilaydigan, yinelemeli tarzda amalga oshiriladi.



12.7-rasm. Agile Unified Process modeli[31].

Model. bu intizom maqsadi, muammo domen loyihasi tomonidan ko'rib chiqilmoqda tashkilotning biznes tushunish uchun, va muammo domen murojaat uchun yashovchan hal aniqlash hisoblanadi.

Amalga oshirish. bu intizom maqsadi bajariladigan kodi ichiga modelini (lar) aylantirish uchun va ayniqsa birligi sinov, sinov asosiy darajasini amalga oshirish hisoblanadi.

[31] Ивутин А.Н., Волошко А.Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.

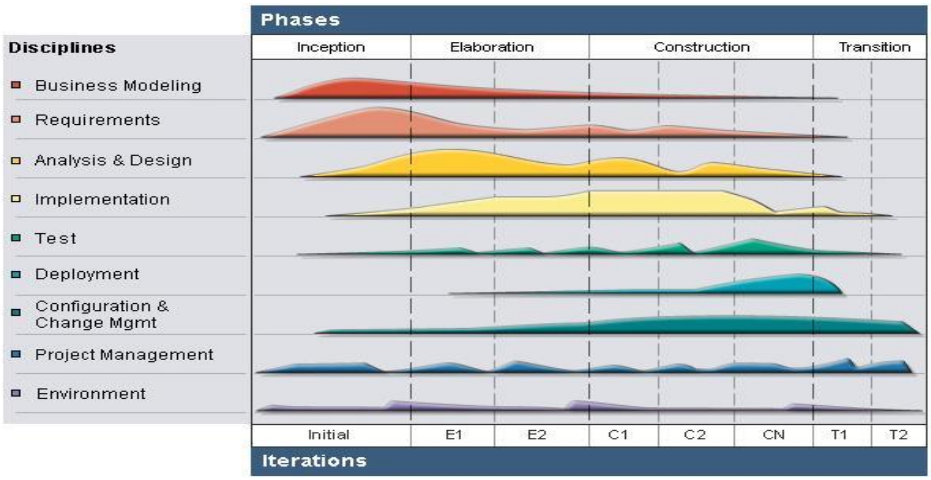
Test. bu intizom maqsadi sifatini ta’minlash uchun ob’ektiv baholash amalga iborat. Bu kamchiliklarni topish mo‘ljallangan bo‘lib tizimi ishlaydi deb tasdiqlagan, va talablar uchrashdi, deb tasdiqlovchi o‘z ichiga oladi.

Tarqatish. bu intizom maqsadi tizimini yetkazib berish uchun rejalashtirish va oxirgi foydalanuvchilar uchun tizim mavjud qilish rejasini amalga oshirish uchun emas.

Konfiguratsiya boshqarish. bu intizom maqsadi sizning loyiha asarlar kirish boshqarish uchun emas. Bu vaqt davomida qo‘lyozmasi versiyalarini kuzatish balki nazorat qilish va ularga o‘zgartirishlar boshqarish nafaqat o‘z ichiga oladi.

Loyiha boshqaruvi. bu intizom maqsadi loyihasi bo‘yicha bo‘lib o‘tadi faoliyatini yo‘naltiradi hisoblanadi. Bu odamlarni (va hokazo, taraqqiyot va ta’qib qilish, vazifalarni belgilash) yo‘l-yo‘riq, va u o‘z vaqtida va byudjet ichida taslim ishonch hosil bo‘lishi uchun loyiha doirasida tashqarida odamlar va tizimlari bilan muvofiqlashtirish, boshqarish xavflarni o‘z ichiga oladi.

Atrof-muhit. bu intizom maqsadi kerak bo‘lsa, to‘g’ri jarayoni, hidoyat (standartlar va ko‘rsatmalar) va vositalari (apparat, dasturiy ta’minot va boshqalar) jamoasi uchun mavjud ta’minlash bilan harakat qolgan qo‘llab-quvvatlashdir.



12.7-rasm. Agile Unified Process modeli fazalari[32].

Inkramental vaqt oraligida yetkazish relizlari

Buning o‘rniga siz barcha o‘rniga bo‘limlari ishlab chiqarish (masalan, versiya 1, shunday qilib, keyin versiya 2 va) uni qo‘yib darrov

[32] Ивутин А.Н., Волошко А.Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.

dasturiy ta’minot yetkazib "Katta portlash" yondashuv. KKS jamoalar odatda oldindan ishlab chiqarish bosqichlari maydoni (lar) har bir iterasyonun oxirida rivojlantirish reliz yetkazib. Talabnomaning rivojlantirish relizlar bu sizning oldingi ishlab chiqarish sifat kafolati (QA), sinov va tarqatish jarayonlari orqali qo‘yish kerak edi, agar mumkin bo‘lgan ishlab chiqarishga ozod bo‘lishi mumkin narsadir. 2-shakl siz birinchi ishlab chiqarish relizlar ko‘pincha keyingi versiyalar ko‘ra qutqarishni uzoq davom etadi, deb ko‘rish; tizimini birinchi ozod siz ehtimol joyda "sanitariya" juda ko‘p qabul qilish kerak va sizning jamoasi ehtimol hali hamkorlikda samarali bo‘lib, ularni imkon "jölelenmişti" yo‘q. birinchi ishlab chiqarish relizlar to‘qqiz oy, xalos qilish uchun o‘n ikki oy ikkinchi ozod sizni olishi mumkin, va keyin boshqa relizlar har olti oyda yetkazib berilmoqda. kengaytirish masalalari yuzasidan erta e’tibor nafaqat siz u ham siz rivojlanishi davomida sizning tajriba foyda olish imkonini beradi muammolarni oldini olish imkonini beradi. Misol uchun, siz bosqichlari maydoni ichiga dasturiy ta’minot tarqatish qachon siz o‘rnatish scriptlar asos bo‘lib xizmat qilishi mumkin, nima ishlaydi va nima yo‘q, notalar eslatmalarni olish kerak.

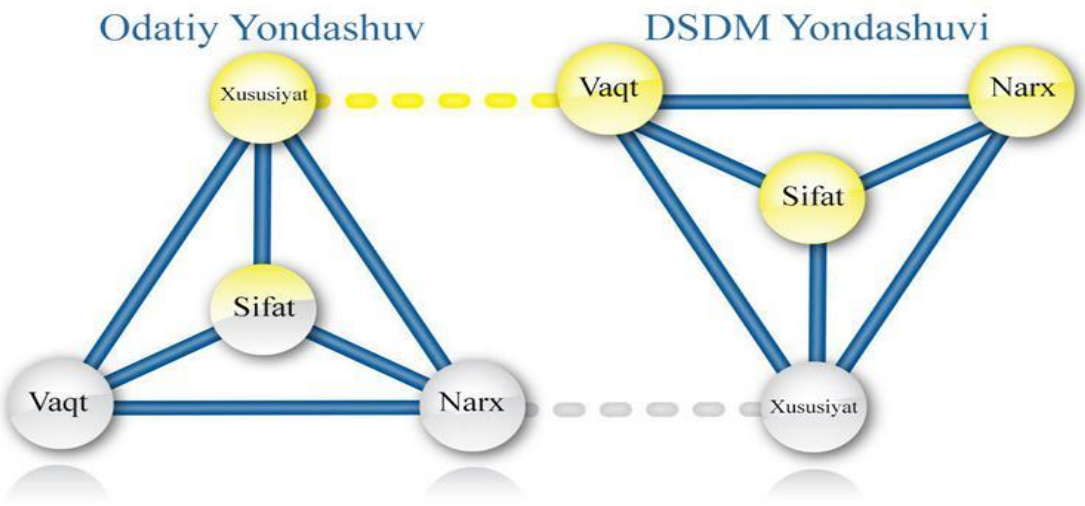
12.4. Dinamik tizimlarni ishlab chiqish usuli

Dinamik tizimlarni ishlab chiqish usuli (Dynamic Systems Development Method, DSDM)- ushbu usul tezkor dastur ishlab chiqish(Rapid Application Development, RAD) kontseptsiyasiga asoslangan. DSDM foydalanuvchining yoki iste’molchining foydalanish jarayonida uzoq muddatli ishtirokini ta’minlaydigan takrorlanuvchi va takomillashtirilgan yondashuv.

Usulning maqsadi - tugallangan loyihani qisqa vaqtda yetkazib berish va kam jamg`armali byudjet doirasida bo‘lishi, ayni paytda loyihani ishlab chiqishda uning talablariga o‘zgartirishlarni tartibga solishdir. DSDM dasturiy ta’minotni ishlab chiqishda moslashuvchan uslubiyot oilasining bir qismiga kiradi, bundan tashqari ishlab chiqarilayotgan maxsulot axborot texnologiyalariga bog`liq bo`lmasligi kerak.

DSDM axborot tizimini qisqa vaqt ichida va kam harajat bilan ishlab chiqishga asoslanganligini qanday tushunsa bo`ladi? DSDMda axborot tizimi loyihalarida odatdagi xatoliklar, masalan, byudjet ortiqligi, yetkazib berish muddati kechikishi, loyiha ishlarida foydalanuvchilar va

yuqori menejerlarning yetarli darajada ishtirok etmasligi haqida ko‘rsatmalar mavjud.



12.8-rasm. Dinamik tizimlarni ishlab chiqish usuli[33].

DSDM quyidagi ko`rinishga ega:

– Uch bosqichdan: Loyiha boshlanishi, Loyihaning yashash sikli va loyiha so`ngi bosqichlari;

– Loyihaning yashash sikli 5 ta bosqichdan iborat: texnik-iqtisodiy asoslash, texnik-iqtisodiy asoslash, funktsional modelni yaratish, loyihalashtirish va ishlab chiqish, amalga oshirish bosqichi.

9 ta prinsipdan tashkil topgan:

1) Foydalanuvchining jalb etilishi –bu samarali loyihani amalga oshirish uchun asos bo`lib xizmat qiladi, bu yerda ishlab chiqaruvchilar foydalanuvchilar bilan birga ish olib borishadi va shuning uchun qabul qilinadigan qarorlar aniqroq bo`ladi;

2) Jamoa loyiha uchun muhim qarorlar qabul qilish vakolatiga ega bo‘lishi kerak;

3) “Kerakli narsani ertaroq qo`yish har doim yaxshi – oxirida o`ylab o`tirgandan ko`ra” qoidasini hisobga olgan holda natija versiyasini tez-tez yetkazib berish. O‘tgan iteratsiya versiyalarining tahlili keyinchalik tahlil qilinadi;

4) Loyiha boshlanishidan oldin talablar yuqori darajada o`rnatilgan bo`ladi.

5) Sinov jarayoni ishlab chiqarishning yashash sikliga iteratsiyalashganligi.

[33] Черников. Управление качеством программного обеспечения: Учебник - М.: ИД ФОРУМ: ИНФРА-М, 2012.

6) Asosiy mezon bugungi bozor ehtiyojlariga javob beradigan dasturiy ta’minotni eng tez etkazib berishdir. Shu bilan birga, bozorning talablariga javob beradigan mahsulotni yetkazib berish, mahsulotning funktsional imkoniyatlari bilan bog’liq muhim muammolarni hal qilishdan ko‘ra muhimroqdir.

7) Ishlab chiqarish – takrorlanuvchi(iterativ) va takomillashtirilgan(inkremental) bo`lishi kerak. Iqtisodiy jihatdan optimal echimga erishish uchun foydalanuvchilarning fikriga asoslanadi.

8) Ishlab chiqarish vaqtidagi har qanday o`zgarishlar – qayta tiklanishi mumkin.

9) Barcha ishlab chiqaruvchilar guruhi birgalikda faoliyat yuritishi va hamkorlik qilishi kerak. Bu ishlab chiqarilayotgan dasturiy maxsulot effektivligiga bog`liq.

DSDM yashash sikli

DSDMning asosini uchta bosqich tashkil etadi. Bular: proekt boshlanish bosqichi, proektning yashash sikli bosqichi va proekt so`ngi bosqichi. Proektning yashash sikli – boshqa bosqichlardan ko`ra eng batafsil va chuqur o`ylanganidir. U 4 ta etapdan tashkil topgan. DSDM bosqichlari:

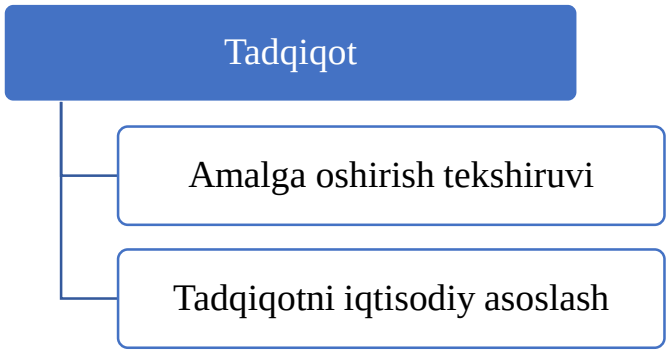
1-bosqich. Proekt boshlanishi: Ushbu bosqichda loyiha g`oyalari ishlab chiqiladi, mablag` va loyihani ishlab chiqaruvchi komanda shakllantiriladi. Ushbu bosqichdagi muammolarni yechish loyihaning keyingi bosqichlarida muammolarni bartaraf etishga yordam beradi.

2-bosqich. Proektning yashash sikli: Pastdagi jadvalda ushbu bosqichning asoslari ko`rsatilgan. Unga ko`ra, sizning axborot tizimingiz muvaffaqqiyatli ishlab chiqilishi uchun 4 ta etap ishlab chiqilgan. Birinchi ikkita bosqich, iqtisodiy umumiylikni amalga oshirish va bir-birini to`ldirishi haqida.

Ushbu bosqichlar muvaffaqqiyatli tugatilganligidan so`ng, axborot tizmining takrorlanuvchi(iterativ) va takomillashtirilgan(inkremental) bosqichlari ishlab chiqiladi: funksional modelni yaratish, loyihalash va ishlab chiqish va so`ngida amalga oshirish bosqichi. DSDM ning takrorlanuvchi va takomillashtiruvchi xususiyati keyinroq tavsiflanadi.

3-bosqich. Proekt so`ngi: Ushbu bosqichda tizim samarali ishlashi ta`minlanadi. Bunga loyihani qo‘llab-quvvatlash, uni takomillashtirish va xatolar DSDM tamoyillariga muvofiq tuzatilishi orqali erishiladi. Ushbu loyiha DSDMning iterativ va inkremental darajasiga asoslangan holda rivojlanish davom etishi bilan qo‘llab-quvvatlanadi.

Amalga oshirish tekshiruvi. Ushbu bosqichda, loyiha DSDM doirasidaligi aniqlanadi. Loyihaning turi, tashkiliy va kadrlar masalalarini hisobga olgan holda, DSDM usulidan foydalanish yoki yo‘qligi haqida qaror qabul qilinadi. DSDM yashash davri bosqichining etapi 12.9-rasmda ko‘rsatilgan.



12.9-rasm. DSDM yashash davri bosqichining etapi [34].

Shunday qilib, qo‘llanilishi mumkin bo`lgan hisobot qabul qilinadi, prototip, rivojlanish rejasi va yuzaga kelishi mumkin bo‘lgan xavf protokolini o‘z ichiga olgan holda taxminiy loyiha rejasi olinadi.

Tadqiqotni iqtisodiy asoslash

Ushbu bosqichda asosiy iqtisodiy va texnologik xususiyatlar tahlil qilinadi. Tizimning eng muhim jihatlarini ko‘rib chiqadigan va rivojlanishning ustuvor yo‘nalishlarini belgilaydigan ekspertlar yig’iladi. Ushbu bosqichda asosiy talablar ro‘yxati, tijorat faoliyati doirasi tavsifi, tizim arxitekturasining ta’rifi va prototip yaratish uchun taxminiy reja ishlab chiqiladi.

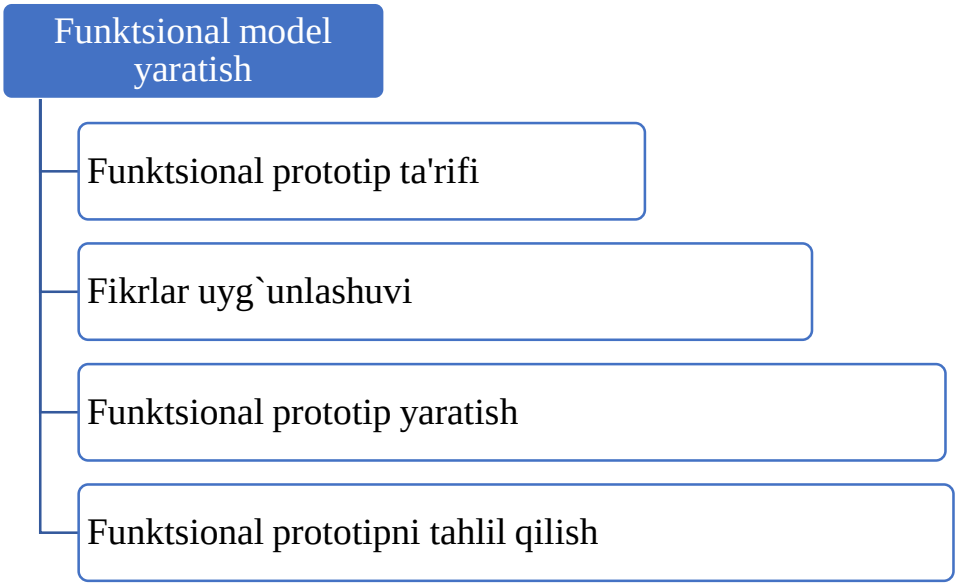
– Funktsional prototip ta’rifi. Ushbu bosqichda prototiplarga qo`yiladigan funksiyalarning ta`rifi keltiriladi. Ushbu kichik bosqichda iqtisodiy asoslash bosqichida olingan natijalarga ko‘ra model ishlab chiqiladi.

– Fikrlar uyg`unlashuvi. Prototipning funktsional imkoniyatlari qancha muddatda ishlab chiqilishi kerakligi haqida fikrlar uyg`unlashadi.

– Funktsional prototip yaratish. Funktsional prototipni fikrlar uyg`unlashuvi va funktsional modelga muvofiq ishlab chiqish.

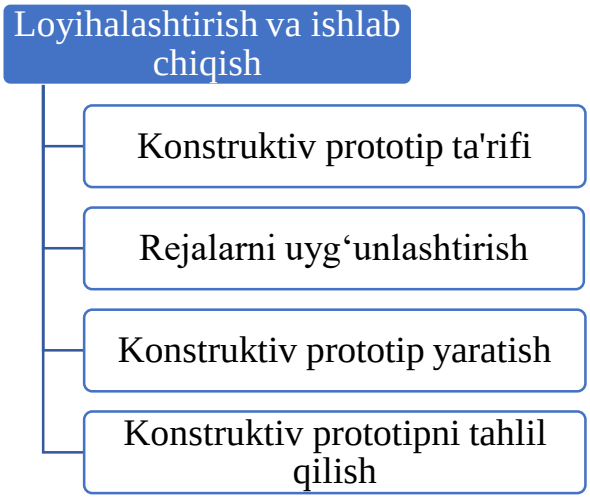
[34] Черников. Управление качеством программного обеспечения: Учебник - М.: ИД ФОРУМ: ИНФРА-М, 2012.

– Funktsional prototipni tahlil qilish. Prototipning xizmat ko‘rsatish imkoniyatini tekshiradi. Bu ishlab chiqilgan prototipni foydalanuvchi tekshirishi yoki hujjatlarni ko`rish orqali amalga oshiriladi.



12.10-rasm. DSDM Funktsional model yaratish bosqichi[35].

Natija-funktsional prototipning umumiy ko‘rinishini o‘z ichiga olgan hujjat shakllanadi.



12.11-rasm. DSDM Loyihalashtirish va ishlab chiqish bosqichi.

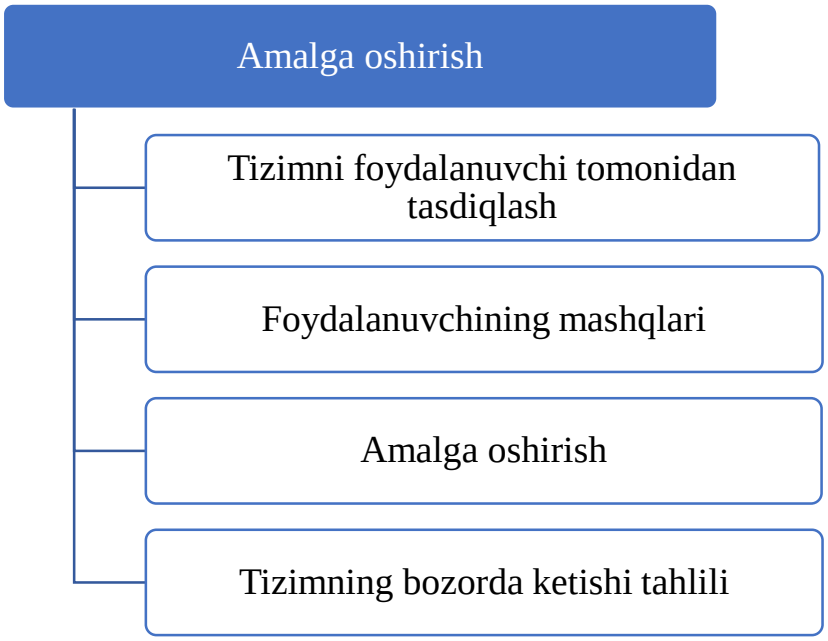
– Konstruktiv prototip ta’rifi. Tizimning funktsional va nofunktsional talablarini aniqlash. Ushbu talablarga asoslanib, dastur strategiyasi yaratilishi kerak.Agar avvalgi iteratsiyadan test yozuvilari mavjud bo‘lsa, u holda u ham dastur strategiyasini yaratish uchun ishlatiladi.

[35] Benjamin J. J. Voigt. Dynamic System Development Method. Department of Information Technology University of Zurich. 2004.

– Rejalarni uyg’unlashtirish. Belgilangan talablar qanday va qancha muddatda amalga oshirilishi to‘g’risida kelishuvlar amalga oshiriladi.

– Konstruktiv prototip yaratish. Testlash uchun foydalanuvchilarga berilishi mumkin bo‘lgan tizimni (konstruktiv prototip) yaratish.

– Konstruktiv prototipni tahlil qilish. Tayyorlangan tizimning funksiyasini tekshirish. Ushbu pastki bosqichda testlanadi va natijalar ko`rib chiqiladi. Foydalanuvchi hujjatlari va test hisoboti yaratiladi.



12.12-rasm. DSDM Amalga oshirish bosqichi[36].

– Tizimni foydalanuvchi tomonidan tasdiqlash. Foydalanuvchilar sinovdan o‘tgan tizimni keyinchalik qo‘llanilishi yoki qo‘llanilmasligi haqida tasdiqlashadi.

– Foydalanuvchining mashqlari. Kelajakda tizimda ishlaydigan foydalanuvchining o`rganishi kerak bo`lgan narsalari. Natija – foydalanuvchilarning o`rganganligi haqida natijalar.

– Amalga oshirish. Foydalanuvchilar orasida testlangan tizimni joriy qilish.

– Tizimning bozorda ketishi tahlili. Ishlab chiqilgan tizimning bozorga ta`sirini tahlil qilish. Asosiy masala - tizimni ishlab chiqishda belgilangan maqsadga erishilganmi. Bunga asoslanib, loyiha keyingi bosqichga o‘tadi (loyihadan keyingi) yoki qayta ko‘rib chiqish uchun avvalgi bosqichga qaytadi. Loyiha tahlili hujjat asosida taqdim etiladi.

[36] Benjamin J. J. Voigt. Dynamic System Development Method. Department of Information Technology University of Zurich. 2004.

12-bob bo‘yicha xulosalar

Metodologiya — bu prinsiplar tizimi, shuningdek dasturiy ta’minotni ishlab chiqish uslubini aniqlaydigan g’oyalar, tushunchalar, usullar, yo‘llar va vositalar birligi hisoblanadi.

Dasturiy ta’minotni yaratishning ko‘plab muvaffaqiyatli uslubiyatlari mavjud. Aniq bir uslubiyatni tanlash jamoaning (guruhning) hajmiga, loyihaning o‘ziga xosligi va murakkabligiga, kompaniyadagi jarayonlarning barqarorligi va yetilganligiga va xodimlarning shaxsiy sifatlariga bog’liq bo‘ladi.

Metodologiya dasturiy ta’minotni ishlab chiqishni boshqarish nazariyasining yadrosi hisoblanadi. Taxmin qilinadigan metodologiya kelajakni atroflicha rejalashtirishga qaratiladi.

Adaptiv metodologiya talablarning kutiladigan to‘laqonli emasligi va ularning doimo o‘zgarishini yengib o‘tish maqsadiga qaratilgan. SCRUM uncha katta bo‘lmagan jamoalar (10 tagcha kishili) uchun metodologiya hisoblanadi. KANBAN – masalaga yo‘naltirilgan dasturiy ta’minotni ishlab chiqishni tez moslashuvchan uslubiyati hisoblanadi.

RUP iterativ va qo‘shimcha rivojlanish modelidan foydalanadi. Iterativ yondashuvda, har bir jarayon uchun ish hajmlari hayot aylanish jarayonida o‘zgarib turadi. Bosqichning oxiridagi nazorat punktlari avvalgi bosqichning qanchalik muvaffaqiyatli ekanligi va loyihaning qanchalik muvaffaqiyatli ekanini baholash imkonini beradi.

EUP metodologiyasi ob’ektga asoslangan paradigmdan foydalanadigan dasturiy ta’minot loyihalarini qo‘llab-quvvatlash uchun mo‘ljallangan RUP taniqli metodologiyasining kengaytmasi hisoblanadi.

Dinamik tizimlarni ishlab chiqish usuli (Dynamic Systems Development Method, DSDM)- ushbu usul tezkor dastur ishlab chiqish(Rapid Application Development, RAD) kontseptsiyasiga asoslangan. Xulosa qilib aytganda ushbu DSDM usulida ishlab chiqilgan dasturiy mahsulot asosan vaqtdan va mablag`dan yutish uchun ishlab chiqiladi.

Agar buyurtmachi tezda dasturiy mahsulotni qo`lga kiritmoqchi bo`lsa, unda siz ushbu DSDM usulida ishlashingiz maqsadga muvofiq bo`ladi. Misol uchun hozirgi kundagi web mahsulotlarni qarasak. Bularning barchasi DSDM usulida ishlab chiqilgan.

DSDMning asosini uchta bosqich tashkil etadi. Bular: proekt boshlanish bosqichi, proektning yashash sikli bosqichi va proekt so`ngi bosqichi.

12-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotni ishlab chiqish metodologiyasi deganda nimani tushunasiz?
2. Rational Unified Process(RUP) metodologiyasi mazmunini aytib bering.
3. Taxmin qilinadigan metodologiya haqida nimalar bilasiz?
4. Adaptiv metodologiya deganda nimani tushunasiz.
5. SCRUM iborasi nimani anglatadi.
6. KANBAN nimani anglatadi?
7. KANBAN metodologiyasining asosiy qoidalari nimalardan iborat?
8. Rational Unified Process(RUP) metodologiyasining asosiy tamoyillarini aytib bering.
9. To‘liq mahsulotni ishlab chiqarish jarayoni necha bosqichdan iborat bo‘ladi.
10. RUP da nechta texnologik jarayon aniqlangan.

13-BOB. DASTURIY TA’MINOTNI TESTLASH VA TEKSHIRISH

13.1. Testlash jarayonining maqsadlari

Dasturiy ta’minotni testlash (Software Testing) - dasturning real va kutilgan natijalari orasidagi moslikni ma’lum tartibda tanlangan chekli test to‘plami (sun’iy tarzda tashkil etilgan vaziyatlar) asosida tekshirish.

Tekshirish (Verification) - bu tizimni yoki uning komponentlarini joriy bosqich boshida shakllantirilgan talablarga qanoatlantirishini baholash jarayonidir. Ya’ni joriy bosqichda loyihani ishlab chiqish uchun aniqlangan vazifalar, qo‘yilgan maqsadlar, muddatlarni bajarilishidir.

Validasiya (Validation) - bu ishlab chiqilgan DTning kutilgan natijalarga va foydalanuvchi extiyojlariga, tizim talablariga mosligini aniqlash.

Testlashni rejalashtirish (Test Plan) - bu testlash bo‘yicha bajarilishi kerak bo‘lgan ishlarni qamrab olgan xujjat bo‘lib, unda obyektni izohi, strategiyasi, jadvali, testlashni boshlash va tugatish bo‘yicha mezonlarni, zarur bo‘lishi mumkin bo‘lgan qurilmalar, maxsus bilimlarni, hamda xavflarni baholash va ularni bartaraf etish variantlarni mavjudligini nazarda tutadi.

Bag/Defekt Report (Bug Report) - bu testlash obyektini noto‘g’ri ishlashiga olib kelgan harakatlar ketma-ketligi yoki shu holatga olib kelgan vaziyatlarni izohi bo‘lib unda kelib chiqish sabablari va kutilayotgan natijalar keltirilgan xujjatdir.

Testni Qoplanishi (**Test Coverage**) – bu testlash sifatini baholash mezonlaridan biri, bajariladigan kodning yoki talablarni testlar yordamida qoplanishidir.

Dasturiy ta’minotni testlash darajalari deganda uning alohida modul ustida, bir guruh modullar yoki butun tizim ustida tekshirilishi tushuniladi. Testlashni hamma darajada amalga oshirilishi - bu loyihani muvaffaqiyatli ishlab chiqish va topshirish degani.

Testlash - bu dasturiy ta’minotni foydalanishga qo‘yishdan oldin dastur nuqsonlarini topish va ularni to‘g’irlashga mo‘ljallangan jarayon hisoblanadi. Siz dasturiy ta’minotni testlagan chog’ingizda, sun’iy ma’lumotlardan foydalanib dasturni ishga tushirasiz. Siz dasturni testlash natijalarini xatolarga, anomaliya (normal holatdan chetlashish) ga yoki dasturning nofunksional sifatlari haqida ma’lumotga tekshirasiz.

Testlash – bu dasturiy maxsulotning ishlash jarayonida ishlamaydigan qismini topishdir. Testlash dasturiy maxsulotning xatoliklarini tuzatishdagi asosiy usul hisoblanadi. Dasturni qayta tayyorlash dasturiy maxsulotning yetishmovchiligini to‘ldirish jarayonidir. Agar dastur anchagina testlar to‘plamidan keyin ham to‘g’ri natijalar chiqarib bersa, bu dasturda xato yo‘q degan asos bo‘lishi kerak emas. Bu paytda dasturni to‘g’rilik darajasida gapirish mumkin.

Test – bu oldindan hisoblangan oraliq va oxirgi natijalari ma’lum bo‘lgan dasturning to‘g’riligini nazorat qiluvchi vositadir. Testni shunday tanlab olish kerakki, dasturchi natijani testni ishlatishdan oldin xisoblab bilsin. Testlash jarayoni quyidagi bosqichlarga bo‘linadi:

1. Normal sharoitda tekshiruv;
2. Ekstrimal sharoitda tekshiruv;
3. Ayrim sharoitda tekshiruv.

Testlash jarayonida ikkita alohida maqsadlar mavjud:

1. **Ishlab chiqaruvchi va buyurtmachiga ularning dasturiy ta’minoti talablari bajarilayotganini namoyish etish.** Buyurtma qilingan dasturiy ta’minot uchun hujjatdagi talablarning har biri uchun kamida bitta testlash bo‘lishi lozim. Umumiy dasturiy ta’minot mahsulotlari uchun esa, tizimning barcha funksiyalari uchun, shuningdek, tayyor mahsulotda ishlatiladigan funksiyalar aralashmasi uchun testlashlar bo‘lishi kerak.

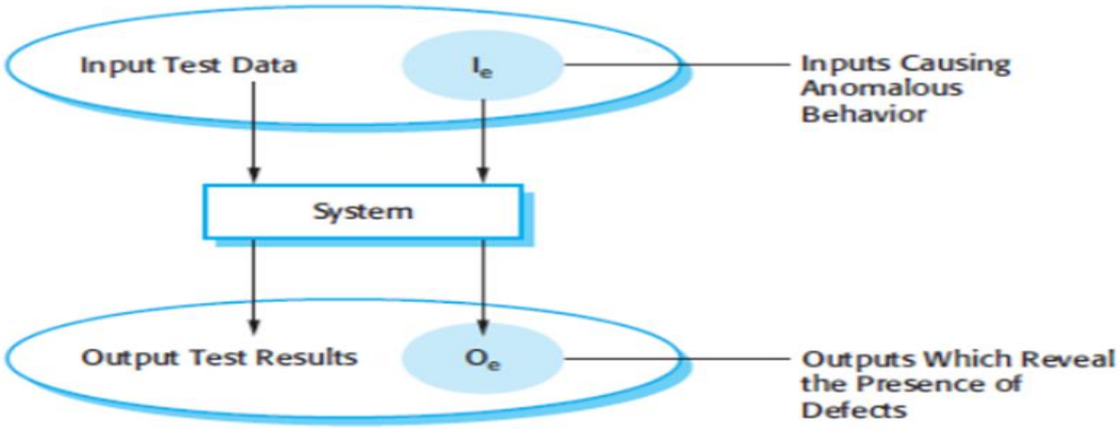
2. **Dasturiy ta’minot noto‘g’ri, ishonarsiz yoki spetsifikatsiyalarga mos kelmagan hollarni aniqlash.** Ular dasturiy ta’minotning nuqsonlari hisoblanadi. Nuqsonlarni testlash keraksiz tizimlarning nuqsonlariga barham berish bilan bog’liq, masalan, tizimning to‘xtab qolishi, boshqa tizimlar bilan keraksiz bo‘g’lanishi, ma’lumotlarning noto‘g’ri hisoblanishi va buzilishi.

Birinchi maqsad -bu ishlatilishi kutilayotgan tizim tekshirishlarini testlashdir. Bunda ishlatilishi kutilayotgan tizimni tekshiruvchilarni berilgan testlar jamlanmasidan to‘g’ri foydalangan holda testlash lozim.

Ikkinchi maqsad- tizim nuqsonlarini testlashga olib keladi. Bunda tizim nuqsonlarini ko‘rsatishi uchun nazorat misollari qo‘yilgan bo‘ladi.

Albatta, testlashning bu ikki yo‘li o‘rtasida aniq bir chegara yoq. Tekshiruvlarni testlash vaqtida siz tizimdagi nuqsonlarni topasiz. Nuqsonlarni testlash vaqtida bazi testlar dasturiy ta’minot ularning talablari javob berayotganini ko‘rsatadi. Testlovchi tizimingizni **qora quti** deb tasavvur qiling. I deb o‘rnatilgan kiruvchi signallardan tizim kiruvchi signallar qabul qiladi va chiquvchi signallarni O deb o‘rnatilgan

chiqishga uzatadi. Chiqishlarning ba’zilari xatolikka tutilishi mumkin. Bular Oye jamlanmasidagi chiquvchilardir, ular Ie jamlanmasidagi kiruvchi signallarga javoban ishlab chiqilgandir. Nuqsonlarni testlashda birinchi o‘rinda Ie da o‘rnatilgan kiruvchilarni topishdir, chunki ular tizim bilan bog’liq muammolarni ochib beradi. Tekshirishlarni testlash Ie dan tashqarida joylashgan to‘g’ri kiruvchilar bilan teslashni o‘z ichiga oladi. Ular kutilayotgan to‘g’ri natijalarni olish uchun tizimni kuchaytiradi.



13.1-rasm. Dasturiy ta’minotni testlash[37].

Testlash dasturiy ta’minotning nuqsonlarga egaligi yoki bir aniq vaqtda u o‘zini ko‘rsatilgandek tutishini namoyon qilmaydi. Siz kuzatayotgan test kelgusida tizim bilan yuz beradigan muammolarni topishi imkoniyati doimo mavjud.

Verifikatsiya va validatsiya jarayonlari dasturiy ta’minotning funksional imkoniyatlarini ta’minlash va uning shartlariga mos kelishini tekshirish bilan bog’liq. Bu tekshirish jarayonlari talablar mavjud hollarda va ular ishlab chiqarish jarayonlarining barcha bosqichlarini davom ettirish vaqtida boshlanadi. Verifikatsiyaning maqsadi dasturiy ta’minot unga qo‘yilgan funksional va nofunksional talablarga javob berishini tekshirishdir. Verifikatsiya umumiy jarayon hisoblanadi. Validatsiyaning maqsadi dasturiy ta’minot buyurtmachi kutayotgan natijalarga mosligini ta’minlashdir. Validatsiya muhim rol o‘ynaydi, sababi, ma’lumotlarni tasniflash talablari har doim ham mijozlar va foydalanuvchilar talablari va istaklarini amalga oshiravermaydi. Verifikatsiya va validatsiya jarayonining yakuniy maqsadi esa dasturiy ta’minot tizimi “maqsadga mos kelishi” ga ishonchni o‘rnatishdir. Yetarli darajadagi ishonch bosqichi, tizimning maqsadi,tizim foydalanuvchilarning talablaridan, shuningdek, tizim uchun joriy marketing muhitidan kelib chiqadi:

[37] Patton, Ron. Software Testing: монография / R. Patton. - 2th. ed. - Indiana: SAMS Publishing, 2006.

1.Dasturiy ta’minot. Dasturiy ta’minotning eng kritik, eng muhimligi bu uning ishonchliligidir. Masalan, tizimlarni ehtiyot qilishni boshqarish uchun qo‘llaniladigan dasturiy ta’minot uchun kerak bo‘lgan ishonch darajasi, mahsulotni yangi g’oyalarini namoyon etish uchun ishlab chiqilgan prototip uchun bo‘ladigan talabdan ancha yuqoridir.

2.Foydalanuvchi kutayotgan natijalar. Ularning tajribalari ishonarsiz dasturiy ta’minot bilan bo‘g’langanligi tufayli, ko‘plab foydalanuvchilar sifatli dasturiy ta’minotni uqadar kutmadilar. Ular dasturiy ta’minot ishdan chiqishidan hayratga tushmaydilar. Yangi tizimni o‘rnatayotganlarida foydalanuvchilar muvoffaqiyatsizlikka ham chidashadi, sababi dasturiy ta’minotdan foydalanish dasturni to‘xtashlaridan keyingi qayta tiklanishiga ketgan harajatlardan ustunroq keladi. Bu kabi holatlarda dasturiy ta’minotni testlashga ko‘p vaqt ajratishingiz kerak bo‘lmaydi. Biroq, dasturiy ta’minot tugallanishga yetishi bilan foydalanuvchilar uning ishonchli, ular xoxlaganidek bo‘lishini kutadilar, shuning uchun dasturiy ta’minotni ishlab chiqishni oxirgi bosqichlarida ko‘proq testlash talab qilinishi mumkin.

3.Marketing muhiti. Tizim bozorga chiqqanida, tizimni sotuvchilar raqobatchi mahsulotlarni, sotib oluvchilar to‘lashga tayyor bo‘lgan narxlarni, shuningdek, ushbu tizimni yetkazilib berilishi uchun talab qilinadigan grafikni inobatga olishlari lozim. Raqobatbardosh muhitda, dasturiy ta’minotni ishlab chiqaradigan korxonalar, uni umumiy testlanishidan oldin dasturni foydalanishga topshirishga qaror qabul qiladilar, sababi ular bozorda birinchi o‘rinda turishni xoxlaydilar. Agar dasturiy ta’minot judayam arzon baholangan bo‘lsa, foydalanuvchilar uning ishonarsizligiga sabr qilishga tayyor bo‘ladilar.

13.2. Testlash tamoyillari

Testlash tamoyillari quyidagilardir:

1. Dasturiy maxsulotni ishlanmasini testlashni asosiy masala deb xisoblab, dasturchi yozgan dasturni testlash maqsadga muvofiq emas.
2. Dasturni to‘g’riligini ko‘rsatadigan emas, balki xatolarni toppish extimoli katta bo‘lgan test.
3. Testlarni to‘g’ri va to‘g’ri bo‘lmagan qiymatlar uchun tayyorlang.
4. Testlarni kompyuterda o‘tkazishni xujjatlashtiring, har bir testning natijasini maydalab o‘rganing.
5. Har bir modulni dasturga faqat bir marta ulang. Testlashni osonlashtirish uchun xech qachon dasturni o‘zgartiring.

6. Dasturiy maxsulotni tekshirish bilan bog’liq bo‘lgan testlarni qayta o‘tkazish yoki dasturiy maxsulotni ishlatish bilan va agar ko‘p o‘zgartirish kiritilgan bo‘lsa, boshqa dasturlar bilan bog’liqligini tekshirish bilan bog’liq bo‘lgan testlarni qayta o‘tkazib yuboring.

Testlash aksiomalari quyidagilardir:

1. To‘g’ri test deb, xatoni topish imkoniyatini beradigan testga aytiladi.
2. Testlarni xujjat ko‘rinishida rasmiylashtirib dasturiy maxsulot xujjatlari bilan birgalikda saqlash zarur.
3. Testlarni nafaqatgina to‘g’ri kirish ma’lumotlar uchun tayyorlash kerak, noto‘g’ri ma’lumotlarni xam (ayrim xolatlar uchun) kiritish kerak.
4. Har bir test natijasini taxlil qilish kerak.

Testlashning asosiy usullari quyidagilardir:

1. Yuqorilovchi testlash
2. Pasayuvchi testlash
3. “Sandwich”
4. Avtonom rostlash

Dasturiy maxsulot bir necha moduldan iborat yuqorilovchi testlash usulida eng pastdagi sodda oddiy modullardan boshlab testlash boshlanadi. Eng asosiy moduldan boshlab test jarayoni o‘tkaziladi va keyin oddiy modullarga test o‘tkaziladi. Ko‘p xollarda yuqorilovchi va pasayuvchi testlashning kombinasiyasidan iborat “Sandwich” degan usul ishlatiladi. Bu usulning ma’nosi shundaki bu ikki testlar jarayoni rostmanayotkan dastur strukturasi o‘rtasida qandaydir bir modulga uchrashib qolmaguncha pasayuvchi va yuqorilovchi testlash bir vaqtda bajariladi.

Yana bir usul bu avtonom rostlash usuli. Ya’ni xir-bir modul aloxida (avtonom) ko‘rinishda testlanadi. Dasturlash vositalarini loyixalashtirishning texnologik ta’minoti. Bu ta’minotga dasturlash vositasining yashash siklida aniqlangan barcha texnologik jarayonlar kiritiladi. Bu jarayonda olib borilgan xujjatlarda dasturlash vositalarini ekspluatasiya qilish va kuzatib borish akslantirilgan bo‘ladi. Ular loyixalash bosqichlarini, ularning natijalarini va sinov usullarini aniqlaydi. Shu bilan birgalikda ta’minotda dasturlash vositasini avtomatlashtirish uchun zarur bo‘lgan instrumental vositalar keltiriladi.

Testlash vositalari

Dasturni testlash jarayonida muhim shartlardan biri shuki, dasturlashga qancha vaqt sarf bo‘lgan bo‘lsa testlashga ham shuncha vaqt

ajratish kerak. Chunki dasturlashda rejalashtirish ishlariga intilish tendentsiyasi mavjud bo‘ladi, shuning uchun barcha loyihalarni muddatlarini buzgan xolda bajarilishini tushinish mumkin. Bu esa ishlab chiqishga, kodlashga, sozlashga va dasturni testlashga aloxida vaqtlar ajratishni va biron bir jarayonni bajarish vaqtini biror xaftaga surilsa, qolgan bosqichlarni bajarish mudatlari xam surilishi mumkin. Test o‘tkazish bosqichlari quyidagilardir:

1. Normal sharoitda tekshirish.
2. Ekstremal sharoitda tekshirish.
3. Favqulot xolatlarda tekshirish.

13.3. Oq, qora va kulrang qutilarni sinovdan o‘tkazish

White-Box va Black-Box atamalari dasturiy ta’minot muhandisligida qo‘llaniladi. Bular dasturiy ta’minotni sinovdan o‘tkazishda qo‘llaniladigan ikkita sinov yondoshuvi, mijozga dasturiy ta’minot sifati to‘g’risida ishonchni berish jarayoni. Dasturiy ta’minotni sinovdan o‘tkazish (odatda dasturni bajarish orqali amalga oshiriladi) dasturiy ta’minotda xatolarni topish maqsadida (dasturiy xatolar deb ham ataladi) o‘tkaziladi.

White-Box testi nima?

Oq quti testi tizimning tuzilishiga asoslangan dasturiy ta’minot tizimini sinash uchun ishlatiladi. Bu ichkarida nimalar sodir bo‘layotganini ko‘rishimiz mumkin bo‘lgan shaffof qutiga o‘xshaydi. Bu tizimning har bir moduli berilgan ma’lumotlarga muvofiq qanday javob berishini chuqur sinovdan o‘tkazadi. Bunday testlash ko‘p vaqt talab etadi, chunki boshqarish tuzilmalarini, ko‘chadan, sharoitlarni, funktsiyalarni va hokazolarni tekshirish kerak. Ushbu yondashuvning sinov usullari har bir birlik uchun ma’lumotlar oqimini sinash, boshqarish oqimini sinash, tarmoq va yo‘l sinovlarini sinashni o‘z ichiga oladi. Bunday sinovni o‘tkazish uchun yuqori texnik sinovchilar kerak.

Oq quti sinovini o‘tkazib, tizimda mavjud bo‘lgan xatolarni kuzatib borish osonroq. Oq-quti sinovi loyihaga qo‘shimcha yuklarni qo‘shadi, chunki ba’zi holatlarda sinov maydonchalarini alohida sinov maydonchalari uchun alohida loyihalar sifatida yaratish kerak. Shuning uchun, bu nihoyat loyiha narxiga va jadvalga salbiy ta’sir qiladi.

Oq qutini sinovdan o‘tkazish:

– dasturiy ta’minotni ichki ma’lumotlari bilan Ichki dasturlash to‘liq ma’lum.

– Tester dasturning ichki ishlashi to‘g’risida to‘liq ma’lumotga ega

– Shisha, ochiq quti, ochiq quti, tarkibiy sinov yoki kod asosida sinov sifatida tanilgan.

– Sinovchilar va ishlab chiquvchilar tomonidan bajarilgan.

– Ichki ishlash to‘liq ma’lum va sinovchi mos ravishda test ma’lumotlarini ishlab chiqishi mumkin.

– Eng ko‘p va ko‘p vaqt talab etiladi.

– Ma’lumotlar domeni va ichki chegaralar yaxshiroq tekshirilishi mumkin.

– Algoritmni sinash uchun mos

Black-Box testi nima?

Qora quti testi, tizim qanday harakat qilishidan qat’iy nazar, faqat tizimning funktsional imkoniyatlarini sinash uchun ishlatiladi. Asosan tizim talablari qoniqtirilganiga ishonch hosil qilish uchun mo‘ljallangan. Bu yopiq qutiga o‘xshaydi, bu erda biz nimani oziqlantirayotganimizni bilamiz va oxir-oqibat u natija beradi, lekin bu qanday ishlab chiqarilganligini bilmaymiz.

Sinov usullari quyidagilarni o‘z ichiga oladi; yuqori darajadagi test o‘tkazish uchun qarorlar jadvalini sinovdan o‘tkazish, davlatga o‘tish jadvallari, ekvivalent qismlarga ajratish va boshqalar. Ushbu sinov oq-qutidagi testlarga qaraganda kamroq vaqt talab etadi, chunki bu tizim berilgan kirishga muvofiq tizim kutilgan natijani beradimi yoki yo‘qligini tekshirish haqida o‘ylaydi.

Sinov holatlari faqat tizim talablariga muvofiq tuziladi. Sinovchining texnik ko‘nikmalari juda kutilmaydi. Agar tizimda xato bo‘lsa, uni kuzatish oson emas, chunki u ichki jarayonni sinab ko‘rmaydi.

Odatda, ushbu ikkala usul ham dasturiy ta’minotni ishlab chiqish muhitida, butun dasturning to‘g’ri ishlashiga ishonch hosil qilish uchun ishlatiladi. Ushbu ikkita sinovni o‘tkazish uchun aniq buyurtma yo‘q va yondashuvlar dasturiy ta’minotni ishlab chiqish hayotiy tsiklining biron bir bosqichiga tegishli emas. Shu bilan birga, qora quti sinovini alohida guruh amalga oshirishi mumkin, oq qutini sinovdan o‘tkazishni esa alohida sinov guruhiga qo‘shimcha ravishda ishlab chiquvchilar yoki dasturchilarning o‘zlari amalga oshirishi mumkin.

White-Box Testing va Black-Box testlari o‘rtasidagi farq nima?

– Oq quti sinovi tizim tizimida sinovlarni amalga oshiradi

– Tizimning talablari qoniqtirilganligini tekshirish uchun qora quti sinovlari

– Oq quti sinovi yuqori texnik sinovchilarga ehtiyoj sezadi
– Qora quti sinovi uchun sinovchining texnik ma’lumotlari unchalik katta emas.
– Oq-qutidagi sinovda ichki xatoliklarni kuzatib borish oson;
– tizim qora quti sinovidan foydalanib qanday ishlashini ko‘rish uchun sinovni bajarish oson

Qora qutini sinovdan o‘tkazishga quyidagicha ta’rif berish mumkin:

– dasturiy ta’minot haqida ichki ma’lumotga ega bo‘linmaydi;
– Ichki dasturlash algoritmi ma’lum emas;
– Ilovaning ichki ishlashi ma’lum bo‘lishi shart emas;
– Yopiq quti, ma’lumotlar boshqaruvi va funktsional test sifatida tanilgan.
– Yakuniy foydalanuvchilar, shuningdek sinovchilar va dasturchilar tomonidan bajariladi.
– Sinov tashqi kutishga asoslangan, dasturning ichki harakati noma’lum.
– Eng kam vaqt sarflaydigan va to‘liq.
– Algoritmni tekshirish uchun mos emas.
– Buni sinov va xato usuli bilan amalga oshirish mumkin.

Kulrang qutini sinovdan o‘tkazish

Ushbu testning maqsadi noto‘g’ri tuzilish yoki noto‘g’ri foydalanish tufayli nosozliklarni aniqlashdir. (Asosan ma’lumotlar bazasini sinovdan o‘tkazish, xalqaro inglizcha imlo: grey-box testi)

– Ichki dasturlash qisman ma’lum.
– Ilovaning ichki ishlashi haqida ozgina ma’lum.
– Shaffof sinov sifatida tanilgan.
– Yakuniy foydalanuvchilar, shuningdek, sinovchilar va dasturchilar tomonidan bajariladi.
– Yuqori darajadagi ma’lumotlar bazasi diagrammalari va ma’lumotlar oqimi diagrammasi asosida.
– Qisman vaqt talab qiladigan va to‘liq.
– Algoritmni tekshirish uchun mos emas.
– Ma’lum bo‘lsa, ma’lumotlar domenlari va ichki chegaralari tekshirilishi mumkin.

Sinov uslubini tanlash

Siz foydalanadigan test yondashuvi bir qator omillarga, shu jumladan baholashga ajratilgan vaqtga, ichki dastur manbalariga kirishga

va test maqsadlariga bog’liq bo‘lishi kerak. Cheklangan resurslarga ega bo‘lgan tajovuzkorlarning qisqa muddatli harakatlarini eng yaxshi taxmin qilish uchun mo‘ljallangan sinovlar qora quti metodologiyasi yordamida o‘tkazilishi mumkin.

Agar test yanada muhim resurslarga ega bo‘lgan tajovuzkorlarning uzoq muddatli harakatlarini aks ettirishga mo‘ljallangan bo‘lsa, kulrang quti testlari buzg’unchilar amaliy guruhlar haqida bilib olishlari mumkinligi to‘g’risida bilimlarni aks ettirishga yordam beradi, bu esa baholash guruhidan mavjud resurslarning to‘liq hajmini sarflashni talab qilmasdan amalga oshiriladi. tajovuzkorlarga. Ilovalar bo‘yicha cheklangan vaqt ichida eng aniq va aniq tavsiyalarni ishlab chiqishi kerak bo‘lgan jamoalar oq yoki aniq quti sinovlaridan foydalanishlari kerak.

Xavfsizlik sinovchilari moslashuvchan bo‘lishi va ushbu stsenariylarning har qandayiga sinov vaqtini rejalashtirish imkoniyatiga ega bo‘lishi kerak, bunda dastur uchun vaqt va manbalar mavjud. Baholash vaqti va sinov manbalarining mavjudligini testning umumiy maqsadlarini hisobga olgan holda, tahlilchilar ushbu cheklovlar doirasida topilgan narsalarning xavfsizlik foydasini oshiradigan sinov usulini tanlashi mumkin. Vaqt va sinov manbalari tabiatda tajovuzkorlarni yoqtirishini tushunganligini hisobga olib, baholash guruhlari o‘zlarining faoliyatlarini shunga mos ravishda optimallashtirishlari kerak. Ishlab chiqarishdagi testlash dasturiy ta’minotni ishlab chiqaruvchilari tomonidan shu jarayonda olib boriladigan barcha testlarni o‘z ichifa oladi. Odatda testlovchi dasturiy ta’minotni ishlab chiqarishda ishtirok etgan dasturchi hisoblanadi. Lekin ba’zan dasturiy ta’minotni ishlab chiqaruvchi jamoa alohida testlovchi va dasturchilardan ham tashkil topgan bo‘lishi mumkin.

13-bob bo‘yicha xulosalar

Testlash - bu dasturiy ta’minotni foydalanishga qo‘yishdan oldin dastur nuqsonlarini topish va ularni to‘g‘irlashga mo‘ljallangan jarayon hisoblanadi. Testlash – bu dasturiy maxsulotning ishlash jarayonida ishlamaydigan qismini topishdir. Testlash dasturiy maxsulotning xatoliklarini tuzatishdagi asosiy usul hisoblanadi. Dasturni qayta tayyorlash dasturiy maxsulotning yetishmovchiligini to‘ldirish jarayonidir. Agar dastur anchagina testlar to‘plamidan keyin ham to‘g‘ri natijalar chiqarib bersa, bu dasturda xato yo‘q degan asos bo‘lishi kerak emas. Bu paytda dasturni to‘g‘rilik darajasida gapirish mumkin.

Verifikatsiya va validatsiya jarayonlari dasturiy ta’minotning funksional imkoniyatlarini ta’minlash va uning shartlariga mos kelishini tekshirish bilan bog‘liq. Bu tekshirish jarayonlari talablar mavjud hollarda va ular ishlab chiqarish jarayonlarining barcha bosqichlarini davom ettirish vaqtida boshlanadi. Verifikatsiyaning maqsadi dasturiy ta’minot unga qo‘yilgan funksional va nofunksional talablarga javob berishini tekshirishdir. Verifikatsiya umumiy jarayon hisoblanadi. Validatsiyaning maqsadi dasturiy ta’minot buyurtmachi kutayotgan natijalarga mosligini ta’minlashdir. Validatsiya muhim rol o‘ynaydi, sababi, ma’lumotlarni tasniflash talablari har doim ham mijozlar va foydalanuvchilar talablari va istaklarini amalga oshiravermaydi. Verifikatsiya va validatsiya jarayonining yakuniy maqsadi esa dasturiy ta’minot tizimi “maqsadga mos kelishi” ga ishonchni o‘rnatishdir.

Dasturni testlash jarayonida muhim shartlardan biri shuki, dasturlashga qancha vaqt sarf bo‘lgan bo‘lsa testlashga ham shuncha vaqt ajratish kerak.

Oq quti testi tizimning tuzilishiga asoslangan dasturiy ta'minot tizimini sinash uchun ishlatiladi. Bu ichkarida nimalar sodir bo‘layotganini ko‘rishimiz mumkin bo‘lgan shaffof qutiga o‘xshaydi. Qora quti testi, tizim qanday harakat qilishidan qat’iy nazar, faqat tizimning funktsional imkoniyatlarini sinash uchun ishlatiladi. Asosan tizim talablari qoniqtirilganiga ishonch hosil qilish uchun mo‘ljallangan. Kulrang qutini sinovdan o‘tkazishning maqsadi noto‘g’ri tuzilish yoki noto‘g’ri foydalanish tufayli nosozliklarni aniqlashdir.

13-bob bo‘yicha nazorat savollari

1. Dasturiy ta’minotni testlash (Software Testing) deganda nimani tushunasiz?
2. Testlashni rejalashtirish (Test Plan) mazmunini aytib bering.
3. Dasturiy ta’minotni testlash darajalari haqida nimalar bilasiz?
4. Testlash jarayoni necha bosqichga bo‘linadi?
5. Testlash jarayonining maqsadlari nimalardan iborat?
6. Verifikatsiya iborasi nimani anglatadi?
7. Validatsiya iborasi nimani anglatadi?
8. Testlashning asosiy tamoyillarini aytib bering.
9. Testlashning asosiy usullarini aytib bering.
10. Oq quti testi mazmunini aytib bering.
11. Qora quti testi mazmunini aytib bering.
12. Kul rang quti testi mazmunini aytib bering.

14-BOB. DASTURIY TA’MINOTNING HIMOYASI

14.1. Dasturiy ta’minot xavfsizligini ta’minlash

Dasturiy ta’minotni himoya qilish - dasturiy ta’minotni ruxsatsiz olish, foydalanish, tarqatish, o‘zgartirish, o‘rganish va analoglarini qayta qurishdan himoya qilishga qaratilgan chora-tadbirlar majmui. Dasturlardan ruxsatsiz foydalanishdan himoya qilish - dasturiy ta’minotdan noqonuniy foydalanishga qarshi kurashishga qaratilgan chora-tadbirlar tizimidir. Himoya qilish uchun tashkiliy, huquqiy, dasturiy va dasturiy va apparat vositalaridan foydalanish mumkin. Nusxa ko‘chirishdan himoya qilish dasturiy ta’minotga kamdan-kam qo‘llaniladi, chunki uni tarqatish va foydalanuvchilarning kompyuterlariga o‘rnatish zarurati. Biroq, dastur uchun litsenziya (jismoniy tashuvchida tarqatilganda) yoki uning individual algoritmlari nusxa ko‘chirishdan himoyalangan bo‘lishi mumkin. Usullarni himoyalangan dasturiy ta’minotni tarqatish usuli va litsenziya tashuvchisi turiga ko‘ra tasniflash mumkin.

Dasturiy ta’minotni lokal himoya qilish

O‘rnatish yoki ishga tushirish paytida seriya raqamini (kalit) kiritish talabi. Ushbu usulning tarixi ilovalar faqat jismoniy tashuvchilarda (masalan, kompakt disklar) tarqatilganda boshlangan. Disk qutisida faqat dasturiy ta’minotning ushbu nusxasiga mos keladigan seriya raqami chop etilgan. Tarmoqlarning ko‘payishi bilan aniq kamchilik tarmoq bo‘ylab disk tasvirlari va seriya raqamlarini tarqatish muammosi edi. Shu sababli, hozirgi vaqtda usul faqat bir yoki bir nechta boshqa usullar (masalan, tashkiliy) bilan birgalikda qo‘llaniladi.

Dasturiy ta’minotni tarmoqda himoya qilish

Dasturiy ta’minotning himoyasini mahalliy tarmoqda tashkil qilish tarmoqni skanerlash bitta mahalliy tarmoq ichidagi ikkita kompyuterda bitta ro‘yxatga olish kaliti bilan ikkita dasturni bir vaqtning o‘zida ishga tushirishni istisno qiladi. Buning kamchiligi shundaki, xavfsizlik devori himoyalangan dasturga tegishli paketlarning o‘tishiga yo‘l qo‘ymaydigan qilib sozlanishi mumkin. To‘g’ri, xavfsizlik devorini o‘rnatish foydalanuvchidan ba’zi ko‘nikmalarni talab qiladi. Bundan tashqari, ilovalar tarmoq orqali o‘zaro ta’sir qilishi mumkin (masalan, tarmoq

o‘yinini tashkil qilishda). Bunday holda, xavfsizlik devori bunday trafikka ruxsat berishi kerak.

Global tarmoq himoyasi

Agar dastur biron bir markazlashtirilgan server bilan ishlasa va usiz foydasiz bo‘lsa (masalan, onlayn o‘yin serverlari, antivirusni yangilash serverlari), u o‘zining seriya raqamini serverga uzatishi mumkin; agar raqam noto‘g’ri bo‘lsa, server xizmatni rad etadi. Kamchilik shundaki, bu tekshiruvni o‘tkazmaydigan serverni yaratish mumkin. Masalan, Battle.net (Blizzard Entertainment-dan) funksionalligi bo‘yicha o‘xshash bo‘lgan battle.da nomli server mavjud edi, lekin foydalanuvchilarga o‘yinlarning ruxsatsiz nusxalarini o‘ynash imkonini berdi. Endi bu server yopildi, lekin ro‘yxatga olish raqamlarini tekshirmaydigan bir nechta PvPGN serverlari mavjud.

CD bilan himoya qilish

Dastur original CDni talab qilishi mumkin. Xususan, bu usul o‘yinlarda qo‘llaniladi. Bunday himoya vositalarining chidamliligi past, chunki kompakt disklarni tasvirga olish uchun asboblar keng tarqalgan. Qoida tariqasida, ushbu himoya usuli bir vaqtning o‘zida kalit bo‘lgan bir xil kompakt diskda yozilgan dasturlarni himoya qilish uchun ishlatiladi. Nusxa ko‘chirishdan himoya qilish uchun quyidagilardan foydalanish mumkin:

- foydalanilmayotgan tarmoqlarda ma’lumotlarni qayd etish;
- "yomon" sektorlarning joylashuvi va mazmunini tekshirish;
- alohida sektorlarni o‘qish tezligini tekshirish.

Dastlabki ikkita usul, tegishli amaliy dastur yordamida diskdan to‘liq tasvirni olib tashlash imkoniyati tufayli amalda foydasizdir. Uchinchi usul yanada ishonchli deb hisoblanadi (xususan, StarForce himoyasida qo‘llaniladi). Ammo ma’lumotlar joylashuvining geometriyasini hisobga olgan holda disklarni taqlid qiladigan va shu bilan ushbu himoyani chetlab o‘tadigan dasturlar mavjud. StarForce-da, boshqa tekshiruvlar qatorida, u kiritilgan diskka yozish qobiliyatini ham tekshiradi. Agar iloji bo‘lsa, disk litsenziyalanmagan deb hisoblanadi. Biroq, agar tasvir CD-R diskiga yozilgan bo‘lsa, unda ko‘rsatilgan tekshirish o‘tadi. CD-R yoki CD-RW oddiy CD-ROM sifatida ko‘rinishi uchun disk turini yashirish mumkin. Biroq, emulyatsiya mavjudligini tekshirish himoya drayveriga o‘rnatilishi mumkin.

Hozirda dunyodagi eng mashhur tizimlar nusxa ko‘chirishdan himoya qilish tizimlari SecuROM, StarForce, SafeDisc, CD-RX va

Tages. Koʻpgina dasturlar uchun koʻrsatilgan himoya usuli mukammal tarqatish usuli tufayli mavjud emas (masalan, Shareware dasturlari).

Elektron kalitlar yordamida himoya qilish

Kompyuter portlaridan biriga (USB, LPT yoki COM interfeysi bilan) oʻrnatilgan elektron kalit (dongle) ishlab chiqaruvchi tomonidan yozilgan litsenziya deb ataladigan kalit maʼlumotlarni oʻz ichiga oladi:

– oʻqish yoki yozish uchun maʼlumot (hozirda u deyarli qoʻllanilmaydi, chunki oʻqilgandan soʻng kalitni taqlid qilish mumkin);

– apparat kriptografik algoritm kalitlari (eng koʻp ishlatiladigan);

– dastur ishlab chiqaruvchisi tomonidan yaratilgan algoritmlar (ixtiyoriy kodni bajarishga qodir mikroprotsessorli elektron kalitlarning paydo boʻlishi tufayli nisbatan yaqinda mavjud boʻlgan usul; hozir u tobora koʻproq foydalanilmoqda).

Elektron kalitlardan foydalangan holda himoya qilishning afzalliklari:

– Kalit dasturni ishga tushirmoqchi boʻlgan har qanday kompyuterga kiritilishi mumkin;

– Kalit diskni talab qilmaydi;

– Elektron kalit kriptografik oʻzgarishlarni amalga oshirishga qodir.

Zamonaviy kalitlar xavfsizlikni ishlab chiquvchi tomonidan joylashtirilgan oʻzboshimchalik kodini bajarishi mumkin (masalan, Guardant Code, Senselock). Xavfsizlikning mustahkamligi asosiy xavfsizlik maʼlumotlari (kriptografik kalitlar, yuklangan kod) u bilan ishlash jarayonida kalitni qoldirmasligiga asoslanadi. Asosiy kamchiliklari:

– Narxi (bir dona uchun 15-30 dollar);

– Kalitni oxirgi foydalanuvchiga etkazish zarurati.

– Kamchiliklar kalitning past tezligi bilan ham bogʼliq boʻlishi mumkin (kompyuter protsessoriga nisbatan). Biroq, zamonaviy dongles 1,25 DMIPS (masalan, HASP, Guardant) koʻrsatkichlariga erishadi va ularning yordami bilan himoya qilish texnikasi dongle bilan doimiy almashinuvni anglatmaydi.

Ilgari maʼlum apparat platformalarida kalitni oʻrnatish bilan bogʼliq muammolar hozirda tarmoq kalitlari (ular himoyalangan dasturning bir yoki bir nechta nusxalari bilan ishlashga qodir, faqat u bilan bir xil mahalliy tarmoqda boʻlishi mumkin) va dasturiy taʼminot yoki apparat "yoʻnaltirish" yordamida hal qilinadi. " USB qurilmalari tarmoq orqali.

Kompyuter sozlamalariga ulanish va faollashtirish

Foydalanuvchi haqidagi ma’lumotlarga bog’lash uning kompyuter komponentlarining seriya raqamlari va keyinchalik dasturiy ta’minotni faollashtirish hozirda keng qo‘llaniladi (masalan: Windows OS). O‘rnatish jarayonida dastur faollashtirish kodini hisoblab chiqadi - o‘rnatilgan kompyuter komponentlari va o‘rnatilgan OT parametrlariga aniq mos keladigan nazorat qiymati. Bu qiymat dastur ishlab chiquvchisiga uzatiladi. Uning asosida ishlab chiquvchi dasturni faqat ko‘rsatilgan mashinada faollashtirish uchun mos bo‘lgan faollashtirish kalitini yaratadi (o‘rnatilgan bajariladigan fayllarni boshqa kompyuterga nusxalash dasturni ishlamay qoldiradi). Afzalligi shundaki, hech qanday maxsus apparat talab qilinmaydi va dastur raqamli tarqatish (Internet orqali) orqali tarqatilishi mumkin.

Buning asosiy kamchiligi: agar foydalanuvchi kompyuterni yangilasa (apparat bilan bog’langan holda), himoya muvaffaqiyatsiz bo‘ladi. Bunday hollarda ko‘plab dasturlarning mualliflari yangi ro‘yxatga olish kodini berishga tayyor. Misol uchun, Windows XP da Microsoft har 120 kunda bir marta yangi ro‘yxatga olish kodini yaratishga imkon beradi (lekin istisno hollarda, faollashtirish xizmatiga qo‘ng’iroq qilib, ushbu muddat tugagandan so‘ng yangi kodni olishingiz mumkin).

Asosan, bog’lovchi sifatida BIOS-ning seriya raqami va qattiq diskning seriya raqami ishlatiladi. Foydalanuvchidan yashirish uchun himoya ma’lumotlari qattiq diskning ajratilmagan qismida joylashgan bo‘lishi mumkin. Yaqin vaqtgacha bunday himoya vositalari dasturiy mahsulotni ishlab chiquvchilar tomonidan ishlab chiqilgan va amalga oshirilgan. Biroq, hozirda dasturiy ta’minot kalitlari bilan ishlash uchun SDKlar mavjud, masalan, Aladdin RD kompaniyasining HASP SL Bundan tashqari, bir vaqtning o‘zida himoya funktsiyasini va faollashtirish litsenziyalash serverini taklif qiluvchi xizmatlar keng tarqalmoqda (masalan, Guardant Onlayn, onlayn himoya).

Dasturni himoya qilishning yana bir sohasi - bu SaaS yondashuvidan foydalanish, ya’ni ushbu dasturlarning funktsiyalarini (to‘liq yoki qisman) xizmat sifatida taqdim etish. Bunday holda, dastur kodi global tarmoqda mavjud bo‘lgan serverda joylashgan va bajariladi. Unga nozik mijoz asosida kirish mumkin. Bu nusxa ko‘chirish himoyasi amalga oshiriladigan kam sonli holatlardan biridir. Kod "ishonchli" tomonda bajariladi, u yerdan nusxa ko‘chirish mumkin emas. Biroq, bu yerda ham bir qator xavfsizlik muammolari paydo bo‘ladi:

– bunday himoyaning kuchi, birinchi navbatda, u bajariladigan serverlarning xavfsizligiga bog’liq (biz Internet xavfsizligi haqida gapiramiz);

– so‘rovlarning maxfiyligini, foydalanuvchi autentifikatsiyasini, resursning yaxlitligini ("issiq" zaxiralash imkoniyati) va umuman yechimning mavjudligini ta’minlash muhimdir.

Xizmatga (shu jumladan yuridik) ishonchga oid savollar ham mavjud, chunki u aslida unga dasturiy ta’minotning o‘zi ham, u qayta ishlaydigan ma’lumotlar ham (masalan, foydalanuvchilarning shaxsiy ma’lumotlari) "ochiq" uzatiladi.

Kodni tahlildan himoya qilish

Bu yerda dastur kodining o‘zini tahlil qilish va boshqa dasturlarda foydalanishdan himoya qilish vositalarini alohida ajratib ko‘rsatish mumkin. Xususan, obfuskatorlar qo‘llaniladi - kodni tahlil qilish, o‘zgartirish va ruxsatsiz foydalanishdan himoya qilish uchun uni xiralashtirish uchun dasturlar kerak.

Mobil platformalarda dasturiy ta’minotni himoya qilish

Mobil platformalar uchun dasturiy ta’minotni nusxalashdan himoya qilish usullari odatda oddiy foydalanuvchi qurilmaning EPROM-da saqlangan ma’lumotlarni o‘qib o‘zgartira olmasligiga asoslanadi. Dasturiy ta’minotni faollashtirish ham ishlatilishi mumkin. BSA Global Software Survey statistik ma’lumotlariga ko‘ra:

– Litsenziyasiz dasturiy ta’minot butun dunyo bo‘ylab shaxsiy kompyuterlarda o‘rnatilgan barcha dasturiy ta’minotning 37% ni tashkil qiladi.

– Soxta dasturiy ta’minotning narxi 46 milliard dollarga baholanmoqda.

– Uchinchi tomon manbalaridan yuklab olingan ko‘plab litsenziyasiz dasturlar korxonalarga zarar yiliga 359 milliard dollarga tushadi.

– Litsenziyalangan dasturiy ta’minotni buzishdan ko‘rilgan zarar har yili taxminan 600 milliard dollarni tashkil qiladi.

14.2. Dasturiy ta’minotni himoyalash turlari

Hammamizga ma’lumki, foydalanuvchilar orasida dasturni halol sotib olib, o‘z maqsadiga ko‘ra ishlatadiganlar, u yoki bu tarzda dasturiy ta’minotni buzib, u bilan ishlayotgan yoki sotayotganlar ham bor. Pullik

mahsulotlarni yaratadigan dasturiy ta’minot ishlab chiqaruvchilari o‘z hayotlarining bir necha yillarini darhol buzilgan va bepul ishlatiladigan dasturga sarflashni xohlamaydilar. Obro‘-e’tiborni yo‘qotish ham muammo bo‘lishi mumkin: masalan, foydalanuvchi shaxsiy kompyuteriga tajovuzkor tomonidan o‘rnatilgan virusni yuqtirgan ma’lum bir kompaniyadan jailbroken dasturiy ta’minotini yuklab olayotganda, jabrlanuvchi hodisa uchun hujumchini emas, balki ishlab chiqaruvchini ayblashi mumkin. Daromad tomonida, BSA ma’lumotlariga ko‘ra, o‘z dasturiy ta’minotini himoya qilishga jiddiy kirishgan biznes daromadning taxminan 11% ga oshishini kutishi mumkin (garchi bu o‘rtacha ekanligini tushunish muhimdir). Ammo dasturiy ta’minotni himoya qilish uchun nima qilishingiz mumkin? Dasturiy ta’minotni litsenziyalash va himoya qilish uchun juda ko‘p turli xil yechimlar mavjud. O‘zingiz uchun birini tanlashdan oldin, bir nechta muhim savollarga javob berishga arziydi. Avvalo, sizning loyihangizga qanday himoya darajasi kerak bo‘lishi mumkinligini hal qilishingiz kerak. Tanlov adekvat bo‘lishi kerak. Ko‘pgina ishlab chiquvchilar haqiqatdan ham kerak bo‘lgandan ko‘ra kuchliroq (va qimmat) himoyadan foydalanishda xato qilishadi.

Ikkinchidan, o‘zingizdan himoya qilish uchun qancha pul sarflashga tayyor ekanligingizni so‘rashingiz kerak. Javob qiyin bo‘lishi mumkin, shuning uchun to‘g’ri tanlov qilish uchun sizga kerak bo‘lgan narsalarni tahlil qilishga arziydi. Keyin, hamma narsani hal qilganingizdan so‘ng, siz dasturiy mahsulotdan foydalanish strategiyasiga asoslangan himoyani tanlashni boshlashingiz mumkin.

Himoyaning asosiy elementlari

Hammasi litsenziyalash tamoyilini tanlashdan boshlanadi: mahsulotingiz qanday to‘lanishini tanlashingiz kerak. Turlari juda ko‘p, ularni to‘rt turga bo‘lish mumkin:

– Bir martalik to‘lov. Siz dasturiy ta’minotingiz uchun bir marta to‘laysiz, shundan so‘ng uni cheksiz muddatga ishlatishingiz mumkin.

– Funktsional cheklovlar. Foydalanuvchi qo‘shimcha haq evaziga qo‘shimcha imkoniyatlar ochishi mumkin.

– Vaqtinchalik litsenziya. Siz "ilovani ijaraga olasiz", ya’ni obuna.

– Ko‘p darajali. Bu nomdagi usullarning kombinatsiyasi. Tegishli to‘lovdan so‘ng foydalanuvchi dasturiy ta’minotning kumush, oltin yoki platinum versiyasini oladi.

Dasturiy ta’minot uchun litsenziya olish jarayonidan so‘ng, dasturiy ta’minotni himoya qilish texnologiyalarini qidirishni boshlash vaqti keladi. Va bu yerda dasturiy ta’minotni Internetga ulash qobiliyati, uning ixtisoslashuvi, dasturiy ta’minot uchun mo‘ljallangan platforma turi va boshqalar kabi muammolarni esga olish kerak. Biz tegishli himoyani tanlash muhimligini yana bir bor ta’kidlaymiz. Agar siz velosipedingizni Fort Noksda qo‘llaniladigan usul bilan himoya qilmoqchi bo‘lsangiz, bu mantiqiy emas. Bundan tashqari, teskari munosabat mavjud: agar siz Fort-Noksni himoya qilmoqchi bo‘lsangiz, buning uchun velosiped qulfidan foydalanmang, bu foydasiz, o‘g’irlik kafolatlanadi. Umuman olganda, litsenziyalash strategiyasi mahsulotning o‘zi narxiga mos kelishi kerak.

Yuqorida aytib o‘tilganidek, dasturiy ta’minotni buzish va nusxalashdan himoya qilishning turli xil variantlari mavjud. Ushbu variantlar xarajat, himoya darajasi va mutaxassislik bo‘yicha farq qilishi mumkin. "Ishonch" bilan himoya qilish. Bu yerda siz foydalanuvchilarning hech qanday muammosiz to‘lashlarini kutasiz. Bitta foydalanuvchi - bitta litsenziya, abadiy. Asosan, siz tomondan deyarli hech qanday xarajatlar yo‘q. Ilova tuzilgandan so‘ng uni tarqatishni boshlashingiz mumkin. Ammo muammo shundaki, agar sizning mahsulotingiz mashhur bo‘lib qolsa, kimdir uni albatta buzadi va berishni boshlaydi. Bunday holda, xakerlikdan himoyalanish yo‘q, u nolga teng.

Dasturiy ta’minotni oflayn himoya qilish

Bu internetga ulanmasdan turib o‘zingizni himoya qilish haqida. Odatda, bunday sxema dasturni tuzishdan so‘ng darhol amalga oshiriladi. Ko‘pincha ma’lum sozlamalarga ega dasturiy ta’minot qobig’i ishlatiladi. Himoyalangan dastur butunlikni tekshirish uchun hech qanday tashqi serverlarga ulanmaydi. Asos sifatida, bunday himoyani hech qanday muammosiz chetlab o‘tish mumkin.

Dasturiy ta’minotni onlayn himoya qilish

Bu yerda biz jiddiyroq usul haqida gapiramiz - litsenziya serveri yordamida litsenziyani tekshirish. Bunday holda, boshida nisbatan yuqori xarajatlar, keyinroq esa takroriy xarajatlar talab qilinadi. Oldingi versiyada bo‘lgani kabi, dasturiy ta’minot qobig’i ishlatiladi, lekin litsenziyalash parametrlari onlayn tarzda tekshiriladi va sozlanadi. Dasturiy ta’minotni tekshirish variantlarini qo‘shish mumkin: u qanday ishlatiladi, litsenziya bor yoki yo‘q. Agar tarmoqqa doimiy ulanish kerak bo‘lsa, unda mahsulot har doim ham, hamma joyda ham ishlashi mumkin

emas. Ushbu himoyaning jiddiyligi o‘rta va yuqori o‘rtasida. Har doim kerakli himoya darajasi va tarmoq samaradorligini tanlash muammosi mavjud. Ba'zi hollarda foydalanuvchilar yoki iste'molchilar xavfsizlik choralarini kirish va samaradorlikni cheklovchi sifatida qabul qilishlari mumkin. Biroq, kriptografiya kabi vositalar foydalanuvchining ma'lumotlarga kirishini cheklamasdan himoya darajasini sezilarli darajada oshirishi mumkin.

Apparatli himoya

Boshqa barcha strategiyalarning afzalliklarini birlashtirgan eng ishonchli usullardan biri. USB dongle litsenziyalash uchun javobgardir, bu tarmoq ulanishini talab qilmaydi. Ishlab chiquvchi uchun har bir kalitning narxi past, davriy qo‘shimcha xarajatlar yo‘q. U API va dasturiy ta’minot qobig’i yordamida amalga oshirilishi mumkin. Ushbu usulning afzalligi shundaki, litsenziyani operatsion tizimdan tashqarida olib tashlash mumkin, kalit shaxsiy kompyuterdan tashqarida saqlanadi. Kalitni nusxalash juda qiyin yoki umuman imkonsiz. Uskuna kaliti bilan himoyalangan dasturiy ta’minot tarmoq ulanishi bo‘lmagan tizimlarda ishlatilishi mumkin. Bular, masalan, davlat muassasalari yoki sanoatdir. Yana bir ortiqcha narsa shundaki, elektron kalit turli xil dasturiy muhitlar uchun turli xil echimlarni talab qilmaydi va litsenziyalash imkoniyatlari juda moslashuvchan.

Apparat kalitga asoslangan himoya vositalari bir necha daqiqada joylashtirilishi mumkin va ular operatsion tizimning deyarli barcha versiyalarida qo‘llab-quvvatlanadi. Himoya vositalari provayderi (agar siz o‘zingiz elektron kalit yarata olmasangiz) hamma narsani tezda bajarishi kerak, shunda elektron kalitlar to‘plamini kutishning hojati yo‘q va shunga mos ravishda dasturiy ta’minot sotuvi boshlanishini keyinga qoldirish mumkin. Sotuvchi, shuningdek, tez tarqaladigan oddiy va samarali yechimni taqdim etishi kerak. Albatta, siz etkazib beruvchiga ishonishingiz kerak - aks holda siz ularning xizmatlaridan foydalanmasligingiz kerak. Konstruktsiyalash bosqichida ham dasturiy ta’minotni himoya qilish haqida o‘ylashingiz kerak: loyiha qisman yoki to‘liq tayyor bo‘lgach, biror narsani o‘zgartirish oson bo‘lmaydi. ma'lumotlarni himoya qilish muammolarini tubdan hal qilish faqat ma'lumotlarni xavfsiz avtomatlashtirilgan qayta ishlash va uzatishning eng muhim muammolarini hal qilishga imkon beruvchi kriptografik usullardan foydalanish orqali erishish mumkin. Shu bilan birga, kriptografik o‘zgartirishning zamonaviy yuqori tezlikdagi usullari

avtomatlashtirilgan tizimlarning dastlabki ishlashini saqlab qolish imkonini beradi.

14.3. Dasturiy ta’minot ishonchliligi va xavfsizligi

Dasturiy ta’minot tizimlarining hajmi va murakkabligi oshib borgani sari, dasturiy injiniring sohasida uchraydigan eng muhim talab bu - biz tizimga ishonishimiz mumkinligini ta’milash ekanligi oydinlashmoqda. Biz biror tizimga ishonishimiz uchun bu tizim talab qilingan ishga mos kelishi va bu ishni to‘g’ri bajarishi kafolatlanmog’i lozim. Buning ustiga tizim xavfsiz bo‘lsin, ya’ni bizning dasturiy ta’minotlarimiz yoki ma’lumotlarimiz bu tizim orqali xavf ostida qolmasin. Bizning ushbu ma’ruzamiz ishonchlilik va xavfsizlik borasidagi muhim ma’lumotlarni o‘z ichiga oladi[38].

Axborot tizimlari shaxsiy hayotimiz hamda ishlarimizga chuqur kirib borgani sari tizim va dasturiy ta’minot nosozligi oqibatida kelib chiqadigan muammolar ham ortib bormoqda. Masalan, elektron tijorat bilan shug’ullanuvchi kompaniyaning serveri dasturiy ta’minotida paydo bo‘lgan nosozlik ko‘p miqdorda yillik daromad boy berilishi, mijozlarning yo‘qotilishiga sabab bo‘ladi.

Hozirda dasturiy ta’minot intensiv tizimlari hukumat, kompaniyalar va jismoniy shaxslar uchun juda ham zarur, shuning uchun keng qo‘llanadigan dasturiy ta’minotlarga qo‘yiladigan eng muhim talablardan biri bu ishonchlilik bo‘ladi. Dasturiy ta’minot talab qilingan vaqtda javob berishi, vazifani to‘g’ri bajarishi hamda autorizatsiyalanmagan ma’lumotlarni oshkor etilishi kabi ishning maqsadiga to‘g’ri kelmaydigan ta’sirlardan yiroq bo‘lishi kerak. ‘Dependability’ ya’ni ‘ishonchlilik’ termini 1995 - yilda Lepray tomonidan tizimining tayyorlik, mustahkamlik, xavfsizlik va himoyalanganlik xususiyatlarini qamrab oluvchi atama sifatida taklif etilgan edi. Quyidagi sabablarda ko‘ra tizimlar ishonchliligi hozirda ularning barcha funsionalligidan ko‘ra muhimdir:

1. Tizim nosozligi ko‘p sondagi insonlarga zarar keltiradi. Ko‘pgina tizimlarning ichki funksionalligi kam ishlatiladi. Agar bu funksionalliklar ya’ni xizmatlardan biri tizimdan olib tashlansa oz miqdordagi foydaanuvchilar zarar ko‘radi. Tizim yaroqliligiga zarar yetkazuvchi nosozlik esa bu tizimdan foydalanayotgan barcha iste’molchilarga zarar ketirishi mumkin. Nosozlik vaqtida normal ish yuritib bo‘lmay qoladi.

[38] “Software Engineering”, by Ian Sommerville, pages 290-295

2. Foydalanuvchilar odatda mustahkam bo‘lmagan, himoyalanmagan yoki xavfsiz bo‘lmagan tizimlarni rad etadilar. Agar foydalanuvchilar tizimni ishonchsiz va himoyalanmagan deb topsalar, uni ishlatishdan bosh tortadilar. Bu narsa esa keyinchalik ushbu tizimni ishlab chiqargan kompaniyaning boshqa mahsulotlariga nisbatan ham foydalanuvchilar ishonchining so‘nishiga olib keladi.

3.Tizim nosozligi juda qimmatga tushishi mumkin. Yadro reaktorini nazorat qilish yoki airoplanlarni boshqaruvchi tizimlarga o‘xshash tizimlarda paydo bo‘ladigan nosozliklardan keladiga zarar, ularni boshqarishga sarflangan xarajatlar qiymatidan ham o‘tib tushadi.

4. Ishonchsiz tizimlar axborot yo‘qotilishiga sabab bo‘lishi mumkin. Gohida hisoblash tizimiga joylashtirilgan ma’lumotlar shu tizimning o‘zidan-da qimmat bo‘ladi. Yo‘qotilgan ma’lumotlarni qayta tiklash esa yanada qimmat turadi.

Ishonchli tizimni loyihalashda quyidagilarni e’tiborga olmoq lozim:

1. Apparat ta’minotidagi nosozliklar. Tizim apparat ta’minoti o‘zining qurilishidagi xatolar yoki biror ehtiyot qismning o‘z vazifasini o‘tab bo‘lgani sababidan nosozlikka duchor bo‘ladi.

2. Dasturiy ta’minot nosozligi. Tizim dasturiy ta’minoti uning tavsifidagi, loyihasidagi yoki realizatsiyasidagi xatoliklar tufayli nosoz bo‘lib qolishi mumkin.

3. Faoliyatdagi nosozlklar. Insonlar tizimdan to‘g’ri foydalanishda va uni to‘g’ri qo‘llashda xato qilishlari mumkin. Apparat va dasturiy ta’minotlar ancha mustahka bo‘lgan hozirgi davrda faoliyatdagi nosozliklar tizim nosozlilarining ko‘pchiligini tashkil etadi deyish mumkin.

Bu nosozliklar ko‘pincha bir-biriga bo‘g’langan bo‘ladi: nosoz apparat ta’minoti tizim operatorlariga qo‘shimcha ishlar yuklashi ularni qiyin ahvolga tushirib qo‘yishi mumkin. Bu narsa esa ulani asabiylashishiga sabab bo‘ladi, insonning asabiylashganida xato qilishi esa tabiiy holdir. Nosoz dasturiy ta’minot bilan ham shunday holatni kuzatish mumkin.

Hisoblash tizimi ishonchliligi - bu tizimga qanchalik ishonish mumkinligini o‘zida aks ettiruvchi xususiyatdir. Bu bilan ishonchlilikni raqamlarda ifodalashni ko‘zda tutmayapmiz. Balki nu o‘rinda "ishonchsiz", "ishonchli", "juda ishonchli" kabi atamalar tzizm ishonchliligini aks ettirish uchun qo‘llaniladi.

Ishonchlilikning to‘rtta asosiy qismi bor:

1. Tayyorlik. Bu xususiyat tizim foydalanuvchi talab qilgan har

qanday vaqtda oz’ xizmatlarini taqdim eta olishidir.

2. Mustahkamlik. Bu xususiyat tizim o‘ziga berilgan vazifani bexato, tavsiflarda keltirilganidek bajarishidir.

3. Himoyalanganlik. Bu tizim kishilarga yoki o‘z muhitiga qanchalik ziyon yekazishi mumkinligini ko‘rsatadigan xususiyat.

4. Xavfsizlik. Bu tizimning qasddan qilingan yoki tasodifiy tahdidlarga qanchalik qarshilik ko‘rsata olishini aks ettiradigan xususiyatdir.

Bu asosiy qismlarga qo‘shimcha ravishda quyidagi xususiyatlarni ham ishonchlilikning tarkibiga kiritish mumkin:

1. Tuzatilish. Tizim nosozliklari muqarrar hodisadir, lekin nosozlik natijasida kelib chiqqan buzilish agar tizimni tezda tuzatish imkoniyati bo‘lsa minimallashtirilishi mumkin. Ochiq kodli dasturiy ta’minotlarda bu ish ancha oson, lekin komponentalarni qayta qo‘llayverish buni qiyinlashtirishi mumkin.

2. Qo‘llab - quvvatlanish. Tizim ishlatilgani sari unga yangi talablar qo‘tib boriladi, shuning uchun talablar asosida tizimning yangi versiyalari ishlab chiqarilishi orqali uni qo‘llab-quvvatlash muhimdir.

3. Saqlanib qolish. Bu Internetga asoslangan tizimlar uchun muhim xossadir. Saqlanib qolish bu - tizimning biror hujum ostida, hatto biror qismi o‘chirib qo‘yilganda ham ishda davom eta olish xususiyatidir. Albatta bunda minimal xizmat ko‘rsata olish nazarda tutilmoqda. Saqlanib qolishni kuchaytirish uchun 3 ta strategiya qo‘llaniladi - hujumga qarshilik qilish, hujumni aniqlash va hujum natijasida ko‘rilgan ziyondan qayta tiklanish.

4. Xatolarga chidamlilik. Bunda ko‘pincha foydalanuvchi xato ma’lumotlar kiritganida, iloji bo‘lsa ularni tuzatish yo‘qsa foydalanuvchiga bu haqdagi xabarni yetkazish tushiniladi

Dasturiy ta’minot xavfsizligi

Biz yuqorida xavfsizlik bu tizimning tashqi qasddan uyushtirilgan yoki tasodifiy hujumlardan o‘zin himoyalay olish xususiyati ekanligi haqida so‘z yuritgan edik. Bu tashqi hujumlar muqarrardir, chunki ko‘pchilik kompyuterlar hozirda internetga ulanadi va bu bilan tashqi tomondan nishonga aylanishi hech gap emas. Bunday hujumlarga viruslar tushishi, tizim xizmatlaridan ruxsatsiz foydalanish, tizimga uatorizatsiyasiz ulanib uning ma’lumotlarini o‘zgartirish kabilarni misol tariqasida keltirish mumkin. Agar siz haqiqatdan xavfsiz tizimda ishlashni xohlasangiz, unda yaxshisi internetga ulanamay qo‘ya qoling. Shunda

agar autorizatsiya qilingan foydalanuvchilar ishonchli bo‘lsa sizning xavfsizlik bo‘yicha muammolaringiz o‘z yechimini topadi. Amalda esa katta tizimlar online rejimida ishlagani uchun yuqori darajada foyda ko‘radilar, internetdan uzilish ular uchun daromadlarning keskin pasayishiga sabab bo‘ladi.

Ko‘pgina tizimlar uchun xavfsizlik bu ishonchlilikning asosiy mezonidir. Harbiy tizimlar, elektron savdo uchun yaratilgan tizimlar hamda o‘ta maxfiy ma’lumotlarga ishlov berish bilan shug’ullanuvchi tizimlar yuqori darajada xavfsizlik ta’minlangan holda loyihalashtirilishi zarur. Masalan, agar havo tarnsportlariga chiptalarni buyurtma qiluvchi tizimda tayyorlik xususiyati past bo‘lsa, bu ishonchning yo‘qolishi hamda ba’zi chiptalardagi kechikishga sabab bo‘lishi mumkin. Agar bu tizim xavfsizligi past bo‘lsa unda hujum qiluvchilar unga kirib barcha buyurtmalarni o‘chirib tashlashlari, buning natijasida esa normal havo yo‘llari harakatlarini davom ettirishga imkoniyat bo‘lmay qolishi mumkin.

Ishonchlilikning boshqa qismlari kabi xavfsizlik ham o‘zining maxsu atamalariga ega.

Pflegeer tomonidan muhim atamalar quyidagicha ta’riflanadi:

Mulk (asset) - himoyalanadigan va biror qiymatga ega bo‘lgan narsa. Mulk bu dasturiy ta’minot tizimining o‘zi yoki bu tizim tomonidan ishlatiladigan ma’lumot bo‘lishi mumkin.

Zararlanish (exposure) - Hisoblash tizimi zaralanishi yoki undagi elementlar yo‘qotilishi bo‘lishi mumkin. Bunda zarar yoki yo‘qotish ma’lumotlarda, vaqtda yoki xavfsizlik buzilganda keyingi tiklash ishlariga ketgan mehnatda ko‘rinadi.

Zaif himoyalanganlik (vulnerability) - Hisoblash tizimi zaifligi, bundan foydalanib tizimga zarar yetkazilishi mumkin.

Hujum (attack) - Tizimning himoyasi zaifligidan foydalanib qolish. Odatda bu tashqi tarafdan bo‘ladi va bunda zarar qasddan yetkazilishi nazarda tutiladi.

Tahdidlar (threats) - zarar yetkazishi mumkin bo‘lgan holatlar, vaziyat va sharoitlar. Bularga tizimga hujum uchun yo‘l ochib beruvchi zaif himoyaga qaragandek qarash lozim.

Nazorat (Control) - tizim himoyasi zaifligini ketkazuvchi chora. Bunga shifrlashni misol qilib keltirish mumkin.

Ixtiyoriy tarmoqqa ulangan tizimda, uch xil asosiy xavfsizlikka qilinadigan tahdidlar uchraydi:

1. Tizim va uning ma’lumotlari maxfiyligiga tahdidlar. Bular

axborotlarning autorizatsiyadan o‘tmagan shaxslar yoki dasturlarga ochilishiga sabab bo‘lishi mumkin.

2. Tizim va uning ma’lumotlari sofligiga tahdid. Bu tahdidlar dasturiy ta’minot yoki ma’lumotlarga zarar yekazishi, ularni buzishi mumkin.

3. Tizim va uning ma’lumotlari tayyorligiga tahdidlar. Bu tahdidlar autorizatsiyadan o‘tgan foydalanuvchilarga ruxsatlarni chegaralab qo‘yishi mumkin.

Albatta bu tahdidlar o‘zaro ichki bog’lanishga ega Agar hujum tizim tayyorligiga zarar yetkazsa, unda siz vaqt o‘tishi bilan o‘zgarib turadigan axborotlarni yangilay olmaysiz. Bu o‘z navbatida tizim sofligini yo‘qqa chiqaradi. Shunday qilib zararlar bir - biriga ulanib ketadi. Amalda, sotsialtexnik tizimlardagi ko‘pchilik himoya zaifligi texnik muammolardan ko‘ra ko‘proq insonlarning xatolari natijasida paydo bo‘ladi. Odamlar oson parollar tanlaydilar, yoki parollarini topib olish oson bo‘lgan joylarga yozib qo‘yadilar, tizim administratorlari ruxsatlarni belgilashda yoki fayllarni joylashtirishda xato qiladilar bundan tashqari foydalanuvchilar himoyalovchi dasturiy ta’minotlarni qo‘llamaydilar.

Siz tizim xavfsizligini kuchaytirish uchun qo‘yishingiz mumkin bo‘lgan nazoratlar quyidagilardir:

1. Himoya zaifligidan chetlanish. Qilinayotgan hujumlar muvaffaqiyatsiz bo‘lishiga ishonch hosil qilish uchun qo‘yiladigan nazoratlar. Bu yerda strategiya tizimni xavfsizlikka oid muammolardan chetda loyihalashdan iborat. Masalan, harbiy tizimlar mahalliy tarmoqlarga ulanmagan bo‘ladi, shuning uchun ularga tashqi kirish yo‘llari berkdir. Ma’lumotlarni shifrlashni ham bu turdagi nazoratlarga kiritish mumkin. Shifrlangan ma’lumotga har qanday autorizatsiyasiz kirishda, bu ma’lumot hujumchilar tomonidan o‘qib bo‘lmaydigan ko‘rinishda bo‘ladi. Amalda, kuchli shifrlangan ma’lumotlarni deshifrlash ko‘p vaqt talab qiladi va qimmatga tushadi.

2. Hujumni aniqlash va uni bartaraf etish. Bu turdagi nazoratlar hujumlarni aniqlab ularni yo‘q qilishga mo‘ljallangan. Bu nazoratlar tizimda bajarilayotgan amallarni kuzatib turadi va g’ayrioddiy holatni aniqlaganda chora ko‘radi: tizimning ushbu qismini o‘chirib qo‘yishi yoki aniqlangan foydalanuvchiga kirish yo‘lini yopib qo‘yishi mumkin.

3. Chegaralar qo‘yish va tiklash. Bu nazoratlar muammolardan keyin qayta tiklanishni qo‘llab-quvvatlaydi.

Talabga javob beradigan xavfsizliksiz, biz tizimning tayyorligi, mustahkamligi hamda himoyalanganligiga ishonolmaymiz.

Tizimni ishlab chiqarishdagi xatoliklar keyinchalik xavfsizlikni aylanib o‘tilishiga olib kelishi mumkin. Agar tizim ko‘zda tutilmagan kiruvchi parametrlarga javob bermasa yoki kiritilayotgan massiv ko‘riniishidagi ma’lumotlarning chegarasi aniqlanmasa, hujumchilar bu zaifliklardan tizimga ruxsatsiz kirish uchun foydalanishlari mumkin. Asosiy xavfsizlik buzilish hodisalari ushbu zaifliklar orqali kelib chiqadi. C# tilida tuzilgan dasturlar massiv chegarasini tekshirishni o‘z ichiga olmaydi, bu esa tizimga ruxsatsiz kirish orqali xotiraning biror qismini qayta yozishga imkon yaratadi. Himoya kompyuter tizimlarini ruxsatsiz kirishdan himoya qilish sohasida paydo bo‘layotgan yangi muammolar nisbatan yuqori hisoblash murakkabligiga ega bo‘lgan mexanizmlar va protokollardan foydalanishni talab qilishi va kompyuter resurslaridan foydalangan holda samarali hal qilinishida namoyon bo‘ladi.

14.4. Tizim ishonchliligi va xavfsizligining xususiyatlari

Tizimning ishonchliligi apparat ta’minot ishonchliligi, dasturiy ta’minot ishonchliligi hamda tizim operatorlari ishonchliligiga bog’liq. Tizim dasturiy ta’minoti bu yerda alohida o‘rin tutadi. Bu o‘z ichiga dasturiy ta’minot nosozligini qoplaydigan talablarni olish bilan birga operator hamda apparat ishonchlilik talablariga bog’liq bo‘lib apparatdagi nosozliklar hamda operator xatolarini aniqlashda yordam berishi mumkin.

Ishonchlilik xavfsizlik hamda himoyalanganlikdan farqli ravishda tizimning o‘lchasa bo‘ladigan xususiyatdir. Tizim ishonchlilik darajasini tavsiflash mumkin, biror vaqt daomida tizim amallari kuzatib turiladi, hamda talab qilingan ishonchlilikka erishilgan bo‘lsa bu belgilab qo‘yiladi. Masalan, ishonchlilikka quyidagicha talab qo‘yish mumkin: tizim qayta yuklanishiga sabab bo‘ladigan tizim nosozligi haftada bir martadan ortiq ro‘y bermasin. Aytilgan nosozlik har ro‘y berganda siz bundan xabar topasiz hamda belgilab qo‘yasiz, shunday qilib talab etilgan ishonchlilikka erishildimi yo yo‘qmi bilib olasiz. Agar erishilmagan bo‘lsa ishonchlilik talablarini qayta ko‘rib chiqasiz yoki tizimdagi muammolarni tuzatishga kirishasiz. Siz past darajadagi ishonchlilikka ham rozi bo‘laverishingiz mumkin, chunki ishonchlilikni oshirish uchun tizimga kiritiladigan o‘zgartirishlar juda qimmatga tushishi mumkin.

Ishonchlilik talablari ikkiga bo‘linadi:

1. Nofunksional talablar. Bular tizim normal ishlab turganida yoki tizim ishga tayyor bo‘lmaganida qabul qilinishi mumkin bo‘lgan nosozliklar sonini aniqlaydi.

2. Funksional talablar. Bular tizim va dasturiy ta’minotning dasturiy ta’minot nuqsonlarini chetlatish, aniqlash va ularga bardosh berish funksiyalarini aniqlaydi va bu nuqsonlar tizim nosozligiga olib kelmasligini ta’minlaydi.

Ishonchlilik talablari shunga bogliq bo‘lgan funksional tizim talablariga yo‘l ochadi. Biror darajadagi ishonchlilikka erishish uchun bu tizimning funksional hamda loyihaviy talablari aniqlanadigan xatoliklarni hamda ular tizim nosozligiga olib kelmasligini ta’minlash uchun ko‘riladigan choralarni tavsiflamog’i lozim.

Umuman olganda tizim ishonchliligini o‘sha tizim biror operatsion muhitda ishlatilganida tizim nosozligi ro‘y berishi ehtimoli bilan tavsiflash mumkin. Maslan 1000 ta ixtiyoriy xizmatdan bittadida nosozlik ro‘y bersa unda nosozlik ehtimolligi 0.001 bo‘ladi. Albatta bu har 1000 ta amalda, aniq bitta nosozlik uchraydi degani emas. Bu agar siz 1000 * N ta amalni kuzatsangiz shunda nosozliklar soni N atrofida bo‘ladi degan ma’noni anglatadi.

Ishonchlilikni tavsiflash uchun ikkita asosiy miqdordan va bunga qo‘shimcha ravishda ishonchlilikka bo‘gliq bo‘lga xususiyat tayyorlikni tavsiflash uchun yana bitta miqdordan foydalaniladi:

1.Talab qilingan nosozlik ehtimolligi - Probability of failure on demand (POFOD). Agar siz bu miqdorni qo‘llasangiz, unda tizim tomnidan biror xizmat uchun belgilangan tizim nosozligi ehtimolligini natija sifatida olasiz.Shunday qilib POFOD = 0.001 ifoda talab bajarilganida nosozlik ro‘y berishi imkoniyati 1/1000 ga bo‘lishini ko‘rsatadi.

2. Nosozliklar sodir bo‘lish darajasi - Rate of occurrence of failures (ROCOF). Bu biror vaqt davomida yoki biror sondagi amallar bajarilish jarayonini kuzatish davomida qayd etilgan nosozliklar soni bilan belgilanadi. Masalan bir soatda ikkita nosozlik yuz bersa unda nosozlik yuz berish oralig’i yarim soat bo‘ladi.

3.Tayyorlil -Availability(AVAIL). Tizimning tayyorligi so‘rovlar jo‘natilganida xizmatlarni yetkazib berishida aks etadi. Masalan AVAIL = 0.9999 bu tizim har vaqt amallarni bajarishga 99.99% tayyor degani.

Tizimlarning xavfsizligiga qo‘yiladigan talablar tavsifi bir jihatdan olib qaraganda himoyalanganlik talablari bilan umumiydir. Shunday bo‘lsa ham xavfsizlik himoyalanganlikka qaraganda muhimroq muammodir. Buning sabablari quyidagicha:

1. Himoyalanganlikni olib qaraydigan bo‘lsak, siz tizim o‘rnatilgan muhitni "dushman" sifatida qaramasligingiz mumkin. Hech kim

himoyalanganlik tomonidan muammo chiqarishga urinib ko‘rmaydi. Ammo xavfsizlik tarafdan yondashuv mutlaqo boshqa natijaga olib keladi. Bunda tashqaridan tizimning zaif nuqtalaridan yaxshigina xabardor qandaydir g’arazli kimsa tizimga tashqi tomondan ta’sir o‘tkazishga harakat qiladi.

2. Agar nosozlik himoyalanganlikdagi tavakkalchilikdan kelib chiqsa, siz nosozlikka sabab bo‘lgan xatolarni va bo‘shliqlarni ko‘rishingiz mumkin. Tashqi hujumlar tizim nosozligini keltirib chiqarganida esa, ildizni toppish qiyinlashib ketadi. Chunki hujumchilar nosozlik sababini yashirishga intiladilar.

3. Odatda tizimni o‘chirib qo‘yish, yoki uning biror xizmatlarini to‘xtatsih himoyalanganlik buzilishi natijasida paydo bo‘lgan nosozliklar uchun eng ma’qul yechimlardan hisoblanadi. Tashqi hujumlar esa ko‘pincha tizimni o‘chirib qo‘yishga yo‘naltirilgan bo‘ladi. Tizim o‘chirib qo‘yilsa hujum muvaffaqiyatli yakunlanibdi deb hisoblayvering.

4. Himoyalanganlik bilan bo‘gliq harakatlar aqlli "dushman" tomonidan amalga oshirilmaydi. Tashqi tomondan hujum qiluvhchi shaxs esa bir qancha tizimlarga hujum qilib tajriba orttirgan bo‘lishi, tizim va uning javoblari haqida ega bo‘lgan bilimlarini qo‘llab o‘z hujumlarini o‘zgartirib turishi mumkin.

Yuqoridagilardan xavfsizlikka qo‘yiladigan talablarrning himoyalanganlik talablariga nisbatan nechog’lik keng miqyosda bo‘lishini ko‘rishimiz mumkin. Qiyoslashlar natijasida, tizimga duch keladigan turli tahdidlarni o‘z ichiga qamrab oluvchi bir necha tur xavfsizlik talablarini keltirish mumkin. Firesmith (2003 – yili) tizim tavsifida mavjud bo‘lishi mumkin bo‘lgan 10 ta xavfsizlik talablarini keltiradi:

1. Identifikatsita talablari. Tizim o‘z foydalanuvchilari bilan muloqotga kirishishdan oldin ular identifikatsiya qilingan yo qilinmaganini tavsiflaydi.

2. Autentifikatsiya talablari. Foydalanuvchilar qanday identifikatsiya-langanini tavsiflaydi.

3. Avtorizatsiya talablari. Identifikatsiyalangan foydalanuvchining imtiyoz-lari va kirish ruxsatlarini tavsiflaydi.

4. Qarshi turish (immunitet) talabari. Tizim viruslar, vormlar va shunga o‘xshash tahdidlardan o‘zninqanday himoyalashini ko‘rsatadi.

5. Soflik talablari. Ma’lumotlar buzilishidan qanday saqlanish mumkinligini ko‘rsatadi.

6. Ruxsatsiz kirishni aniqlash talablari. Tizimga qilinayotgan

hujumlarni aniqlashda qanday mexanizm qo‘llanishini ko‘rsatadi.

7. Rad etilmaslik talablari. Biror xizmat a’zolaridan biri xizmatning o‘ziga tegishli qismini inkor etmasligini ko‘rsatadi.

8. Sir saqlash talablari. Ma’lumotlarni sir saqlash qanday qo‘llab-quvvatlanishini ko‘rsatadi.

9. Xavfsiz nazorat talablari. Tizimdan foydalanish qanday kuztilishi va tekshirilishi mumkinligini ko‘rsatadi.

10. Tizimni qo‘llab-quvvatlashning xavfsizligi talablari. Ilova qanday qilib xavfsizlik mexanizmidagi tasodifiy muvaffaqiyatsizlikdan so‘ng autorizatsiyada yuzga keladigan o‘zgarishlardan saqlanishini ko‘rsatadi.

Albatta siz yuqoridagi xavfsizlik talablarning har birini barcha tizimlarda ham uchratavermaysiz. Bu talablarning qo‘llanishi tizim turiga, undan foydalanish holatiga va undan foydalanishi mumkin bo‘lgan iste’molchilarga bo‘gliqdir.

Dasturiy ta’minot xavfsizligining ehtimolligini boshqarish

Dasturiy ta’minot xavfsizligining ehtimolligini baholash va boshqarish samarali xavfsiz mexanizm qurishda juda muhimdir. Ehtimollikni boshqarish tizim mulkiga hujum natijasida kelib chiqishi mumkin bo‘lgan yo‘qotishlarni baholash hamda bu yo‘qotishlarni ularni yo‘qotishi mumkin bo‘lgan xavfsizlik xizmati qiymati bilan muvozanatlashtirish ishlari bilan ham uzviy bog’liqdir. Kredit kartochka kompaniyalari buni har doim amalga oshiradilar. Kredit kartochkalardagi tovlamachiliklarni yo‘qotish uchun yangi texnologiyalar ishlab chiqish nisbatan oson ishdir. Shunga qaramay, tovlamachilikka yo‘l qo‘ymaydigan tizimni sotib olish va uni o‘rnatishdan ko‘ra kompaniyalarga foydalanuvchilarining zararlarini qoplash uchun tovlamachilikdan ko‘rilgan ziyonni qoplab berish arzonga tushadi. Yo‘qotish va hujumlar qiymatiga qarab bu muvozanat buzilishi mumkin.

Ehtimollikni boshqarish texnik ishlardan ko‘ra tadbirkorlik ishlariga yaqinroqdir. Dasturiy ta’minot muhandislari tizimda qanday nazoratlar o‘rnatilishi haqida qaror qabul qilmaydilar. Xavfsizlik tizimi qiymatini qabul qilish yoki natijalarni xavfsizlik xizmatlaridan mahrum qilish bo‘yicha qarorni yuqori menejment qabul qiladi. Shunga qaramasdan dasturiy ta’minot muhandislarining texnik yo‘riqnomalar berish hamda xavfsizlik muammolarini hal qilish yo‘llarini ko‘rsatishdagi rollari beqiyosdir. Shuning uchun ham ular ehtimolli holatlarni boshqarish jarayonida asosiy ishtirokchilardan bo‘lib qolveradilar.

Ehtimollikni baholash tizim qurilishidan oldin boshlanadi, tizim

ishlab chiqarish jarayonida hamda tizim iste’molga chiqarilganda ham u davom etaveradi. Ehtimollikni baholash uch bosqichdan iboratdir:

1. Dastlabki ehtimollikni baholash. Preliminary risk assessment. Bu bosqichda hali tizimga qo‘yilgan barcha talablar, tizim arxitekturasi yoki realizatsiyasi borasida hali qaror qabul qilinmagan bo‘ladi.

2. Yashash siklidagi ehtimollikni baholash. Life-cycle risk assessment. Bu ehtimollikni baholash tizimni ishlab chiqish yashash sikli davomida amalda bo‘ladi hamda tizimning texnik arxitekturasi va realizatsiya haqidagi qarorlardan ma’lumotlarni qabul qiladi.

3. Faoliyatdagi ehtimollikni baholash. Operational risk assessment. Tizim ishlab chiqarilgani va iste’molga chiqarilganidan so‘ng, foydalanuvchilarga tizim qanday qo‘llanishini ko‘rsatish hamda yangi va o‘zgargan talablarga mos ravishda takliflar kiritish uchun kerak. Faoliyatdagi talablarga doir taxminlar tizim noto‘g’ri tavsiflanganda qilinadi. Tashkiliy o‘zgarishlar tizim asl rejadan tashqari maqsadlarda qo‘llanilayotganini ko‘rsatadi. Faoliyatdagi ehtimolliklarni baholash tizim rivojlangani sari unga yangi xavfsizlik talablarini qo‘yishga olib keladi.

Ehtimolliklarni baholash uchun sizi avvalo tizimga duch kelishi mumkin bo‘lgan tahdidlarni aniqlab olishingiz lozim. Buni amalga oshirishning bir yo‘li "noto‘g’ri holatlar" (misuse cases) to‘plamini ishlab chiqishdir. "Noto‘g’ri holatlar" bu tizim bilan tizimga zararli bo‘lgan o‘zaro ta’sirlarga kirishdigan holatlardir. Siz bu holatlarni mumkin bo‘lgan tahdidlarni aniqlash va ularni muhokama qilishda qo‘llashingiz, binobarin bundan tizim xavfsizligiga qo‘yiladigan talablarni ishlab chiqishda foydalanishingiz mumkin. Pflegeer tahdidlarni mumkin bo‘lgan noto‘g’ri holatlarni aniqlashda boshlang’ich nuqta bo‘lishi mumkin bo‘lgan to‘rtta bo‘lim ostida aks ettiradi. Bular quyidagilar:

1. Ko‘rib olish hujumlari. Hujum qiluvchiga tizim va uning ma’lumotlariga

kirish yo‘lini ochadi.

2. Uzib qo‘yish hujumlari. Bular hujum qiluvchilarga tizimning biror qismini tayyorlik holatidan chiqarish imkonini beradi.

3. O‘zgartirish hujumlari. Bu hujumlar natijasida hujum qiluvchi tizimning muhim ma’lumotlarini o‘zgartirish imkoniyatiga ega bo‘lishi mumkin.

4. Soxtalashtirish hujumlari. Bular hujum qiluvchiga tizim ichiga noto‘g’ri axborotlar kiritish imkonini beradi.

Faoliyat davomida ehtimolliklarni baholash. Dasturiy ta’minot

xavfsizligi ehtimolligini baholash va uni boshqarish, bu mahsulot yashash siklidan keyin ham davom etishi mumkin. Chunki vaqt o‘tishi bilan yangi ehtimolli holatlar paydo bo‘ladi va tizim ular bilan kurasha olishi uchun unga o‘zgartirish kiritilishi mumkin.

Mana shu jarayon faoliyat davomida ehtimolliklarni boshqarish deyiladi. Yangi ehtimolli holatlar tizimga qo‘yilgan talablar o‘zgarishi natijasida kelib chiqishi mumkin. Chunki tizimga qo‘yilgan talablar o‘zgarishi tizim infrastrukturasi o‘zgarishiga yoki tizim qo‘llanilayotgan muhitning o‘zgarishiga sabab bo‘ladi.

Faoliyat davomida ehtimolli holatlarni boshqarish, yashash siklida ehtimolliklarni boshqarishga o‘xshaydi, biroq qo‘shimcha ravishda tizim ishlatilayotgan muhit haqidagi axborotlarni ham o‘z ichiga oladi. Muhitning xususiyatlarini bilish juda muhim ahamiyatga ega. Chunki ular yangi ehtimolli holatlarni keltirib chiqarishi mumkin.

14-bob bo‘yicha xulosalar

Dasturlardan ruxsatsiz foydalanishdan himoya qilish - dasturiy ta’minotdan noqonuniy foydalanishga qarshi kurashishga qaratilgan chora-tadbirlar tizimidir. Himoya qilish uchun tashkiliy, huquqiy, dasturiy va dasturiy va apparat vositalaridan foydalanish mumkin. Dasturiy ta’minotning himoyasini mahalliy tarmoqda tashkil qilish tarmoqni skanerlash bitta mahalliy tarmoq ichidagi ikkita kompyuterda bitta ro‘yxatga olish kaliti bilan ikkita dasturni bir vaqtning o‘zida ishga tushirishni istisno qiladi. Hammamizga ma’lumki, foydalanuvchilar orasida dasturni halol sotib olib, o‘z maqsadiga ko‘ra ishlatadiganlar, u yoki bu tarzda dasturiy ta’minotni buzib, u bilan ishlayotgan yoki sotayotganlar ham bor. Pullik mahsulotlarni yaratadigan dasturiy ta’minot ishlab chiqaruvchilari o‘z hayotlarining bir necha yillarini darhol buzilgan va bepul ishlatiladigan dasturga sarflashni xohlamaydilar.

Dasturiy ta’minotni litsenziyalash va himoya qilish uchun juda ko‘p turli xil yechimlar mavjud. dasturiy ta’minotni buzish va nusxalashdan himoya qilishning turli xil variantlari mavjud. Ushbu variantlar xarajat, himoya darajasi va mutaxassislik bo‘yicha farq qilishi mumkin. Dasturiy ta’minot tizimlarining hajmi va murakkabligi oshib borgani sari, dasturiy injiniring sohasida uchraydigan eng muhim talab bu - biz tizimga ishonishimiz mumkinligini ta’milash ekanligi oydinlashmoqda. Biz biror tizimga ishonishimiz uchun bu tizim talab qilingan ishga mos kelishi va bu ishni to‘g’ri bajarishi kafolatlanmog’i lozim.

Axborot tizimlari shaxsiy hayotimiz hamda ishlarimizga chuqur kirib borgani sari tizim va dasturiy ta’minot nosozligi oqibatida kelib chiqadigan muammolar ham ortib bormoqda. Hozirda dasturiy ta’minot intensiv tizimlari hukumat, kompaniyalar va jismoniy shaxslar uchun juda ham zarur, shuning uchun keng qo‘llanadigan dasturiy ta’minotlarga qo‘yiladigan eng muhim talablardan biri bu ishonchlilik bo‘ladi. Dasturiy ta’minot talab qilingan vaqtda javob berishi, vazifani to‘g’ri bajarishi hamda autorizatsiyalanmagan ma’lumotlarni oshkor etilishi kabi ishning maqsadiga to‘g’ri kelmaydigan ta’sirlardan yiroq bo‘lishi kerak.

Ko‘pgina tizimlar uchun xavfsizlik bu ishonchlilikning asosiy mezonidir. Harbiy tizimlar, elektron savdo uchun yaratilgan tizimlar hamda o‘ta maxfiy ma’lumotlarga ishlov berish bilan shug’ullanuvchi tizimlar yuqori darajada xavfsizlik ta’minlangan holda loyihalashtirilishi zarur. Dasturiy ta’minot xavfsizligi ehtimolligini baholash va uni boshqarish, bu mahsulot yashash siklidan keyin ham davom etishi mumkin. Chunki vaqt o‘tishi bilan yangi ehtimolli holatlar paydo bo‘ladi va tizim ular bilan kurasha olishi uchun unga o‘zgartirish kiritilishi mumkin.

14-bob bo‘yicha nazorat savollari

1. Tizim ishonchliligining asosiy olti asosiy xususiyatini ayting.
2. Tizimda uchrashi mumkin bo‘lgan nosozliklar turlarini sanang.
3. Nima uchun mustahkamlik talablari oshgani sari tizim ishonchliligini

ta’minlashning qiymati favqulodda tez ko‘tarilib ketadi?

4. Tizim mustahkamligiga qanday erishiladi?
5. Mustahkamlik va xavfsizlik o‘zaro bo‘g’liq bo‘lgan ishonchlilik xususiyatlaridir, ayni paytda ular alohida xususiyatlardir. Ular orasidagi eng muhim farqni bayon qiling.
6. Xavfning jiddiyligi qanday baholanadi?
7. Himoyalanganlik i qanday mezonlar asosida baholanadi?
8. Himoyalanganlikdagi tahdid va hujum tushunchalari qanday farqlanadi?
9. Ilova darajasidagi hamda infrastruktura darajasidagi himoyalanganlik bi biridan qanday farq qiladi?
10. Tizimda yuzaga keladigan nosozliklarning katta qismi qaysi turdagi tizim nosozliklari tashkil etadi?

15-BOB. DASTURIY TA’MINOTNI RIVOJLANTIRISH TENDENSIYALARI

15.1. Dasturiy ta’minotning evolutisiyasi

Dasturiy ta’minot quyidagi holatlarda o‘zgarishi muqarrardir:

- Dasturiy maxsulotdan foydalanilayotganda yangi talablar yuzaga keladi;
- Biznes muhit o‘zgadi;
- Xatoliklarning tamirlanish majburiyati;
- Yangi kompyuter va jihozlarning sistemaga qo‘shilishi;
- Sistemaning ish bajarishi yoki ishonchliligini oshirishga majbur bo‘lish.

Barcha tashkilotlar uchun asosiy muammo ularning mavjud dasturiy taminoti uchun o‘zgarishlarni amalga oshirish va boshqarishdir. Evolutsiyaning ahamiyati:

– Tashkilolarning dasturiy taminot tizimlarida juda katta investitsiyasi bo‘lishi bu katta mulkdir;

– Bu mulkni biznesda qiymatini saqlab qolish uchun ular o‘zgartirilishi va yangilanib borishi lozim;

– Katta kompaniyalardagi dasturiy maxsulot mablag’ining katta qismi yangi dasturiy taminot yaratgandan ko‘ra mavjjud dasturiy taminotni rivojlantirish va o‘zgartirishga sarflanadi.

Evolutsiya va servis xizmat

Evolutsiya - bu dasturiy taminot hayot siklining shunday bosqichiki bunda u tezkor o‘zgarishda bo‘ladi va taklif qilingan yangi talablar bosqichma – bosqich shaklanadi, hamda sistemada amalga oshiriladi.

Servis xizmat ko‘rsatish - bu bosqichda dasturiy maxsulot foydali bo‘lib qoladi lekin faqatgina o‘zgarishlar uning tezligini oshirish maqsadida qo‘shiladi ya’ni dasturiy taminotda muhitida xatolarni to‘grilash va o‘zo‘zgarishlarni tasvirlash amalga oshiriladi. Yangi funksiyalar esa qo‘shilmaydi.

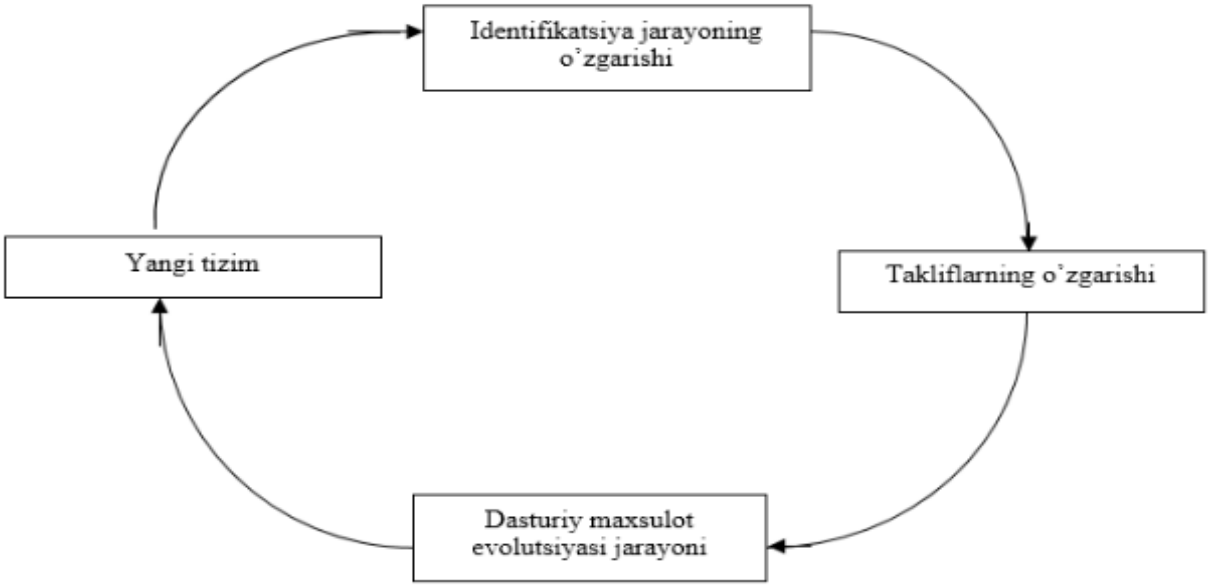
Bosqichdan chiqish - dasturiy taminot haligacha ishlatiladi lekin uni hosil qilish uchun yangi o‘zgartirishlar kiritilmaydi.

Dasturiy maxsulot evolutsiyasi quyidagilarga bog’liq:

- Saqlanib kelayotgan dasturiy maxsulotning turi;
- Foydalanilayotgan qurilish jarayoni;
- Loyihaga jalb qilingan ishchilarning tajribasi va qobilyati.

O‘zgarishlar uchun takliflar tizim evolutsiyasi uchun asosiy hisoblanadi. Inedtifikatsiya va evolutsiyaning o‘zgarishi tizimning butun hayot sikli mobanida davom etadi.

Evolutsiya va identifikatsiya o‘zgarish jarayonlari



15.1-rasm. Evolutsiya va identifikatsiya o‘zgarish jarayonlari[39].

Shakllantirilgan amalga oshirilgan va test qilingan sistemani takrorlantiradigan yaratilish jarayonining qaytishi(takrorlanishi). Muhim farq shundaki amalga oshirish o‘zgarishining birinchi bosqichi o‘z ichiga dasturni tushunishni oladi. ayniqsa bu holathaqiqiy tizim tashkilotchilari amalga oshirishga javobgar bo‘lmaganlaridasodir bo‘ladi. Dastruni tushunish bosqichi davomida siz dasturning tuzilish strukturasini, dasturga yaxshi tasir qiladigan o‘zgartirishlarni qanday taklif qilishni va uning funksiyonalligini qanday ta’minlab berishni tushunishingiz shart. Zarur o‘zgarishlar dasturiy inginiring jarayonining barcha bosqichlarida ham amalga oshirish shart bo‘lmasligi mumkin:

– Jiddiy tizim xatosi normal amaliyot davom ettirishga imkon berishni tuzatish kerak bo‘lganda;
– Agar sistema muhiti uchun kutilmagan tasirlar bo‘lsa;
– Agar tezda javob qaytarishni talab qiladigan talablar bo‘lsa.

Dasturiy ta’minot evolyutsiyasi - bu o‘zgaruvchan manfaatdor tomonlar va bozor talablariga javob beradigan dasturiy ta’minotning dastlabki chiqarilganidan keyin uzluksiz rivojlanishi. Dasturiy ta’minotning evolyutsiyasi muhim ahamiyatga ega, chunki tashkilotlar

[39] Романов А.А. Конструирование программного обеспечения: учебное пособие. – Ульяновск: УлГТУ, 2016.

o‘z dasturiy ta’minotiga katta miqdorda mablag’ sarflaydi va to‘liq ushbu dasturiy ta’minotga bog’lanib qoladi. Dasturiy ta’minot evolyutsiyasi dasturiy ta’minotni o‘zgaruvchan biznes talablariga moslashishga, nuqsonlarni tuzatishga va dasturiy ta’minot tizimlari muhitida boshqa o‘zgaruvchan tizimlar bilan integratsiyalashuviga yordam beradi. Fred Bruks o‘zining "Afsonaviy odam-oy" kitobida odatdagi tizim narxining 90% dan ortig’i xizmat ko‘rsatish bosqichiga to‘g’ri kelishini va har qanday muvaffaqiyatli dasturiy ta’minot muqarrar ravishda xizmat ko‘rsatilishishini ta’kidlaydi. Aslida, Agile usullari veb-texnologiyalarda va uning atrofida texnik xizmat ko‘rsatishga o‘xshash faoliyatga asoslanadi, bu yerda imkoniyatlarning asosiy qismi ramkalar va standartlardan kelib chiqadi.

Dasturiy ta’minotga texnik xizmat ko‘rsatish xatolarni tuzatish va kichik yaxshilanishlarga qaratilgan bo‘lsa, dasturiy ta’minotni rivojlantirish moslashish va boshqa tizimga ko‘chirishga qaratilgan. Dasturiy ta’minot texnologiyalari rivojlanishda davom etadi. Ushbu o‘zgarishlar yangi qonunlar va nazariyalarni yaratish va asoslashni talab qiladi. Ba’zi modellar kelajakdagi dasturlarni ishlab chiqishda qo‘shimcha mulohazalarni ham talab qiladi. Innovatsiyalar va yaxshilanishlar haqiqatan ham dasturiy ta’minotni ishlab chiqishning kutilmagan shaklini oshirmoqda. Kelajakdagi dasturiy ta’minot evolyutsiyasiga moslashish uchun texnik xizmat ko‘rsatish muammolari ham o‘zgarishi mumkin.

Dasturiy ta’minot jarayonlari o‘z-o‘zidan rivojlanib, o‘qitish va takomillashtirishdan o‘tib, ularning samaradorligi va natijaviyligini doimo oshiradi. Dasturiy ta’minot evolyutsiyasiga bo‘lgan ehtiyoj hech kim foydalanuvchi talablari qanday rivojlanishini oldindan aytib bera olmasligidan kelib chiqadi. Boshqacha qilib aytganda, mavjud tizimlar hech qachon tugallanmagan va rivojlanishda davom etadi. Tizimlar rivojlanib borar ekan, bu muammolarni hal qilish uchun yaxshiroq yechim topilmasa, tizimlarning murakkabligi oshadi. Dasturiy ta’minot evolyutsiyasining asosiy vazifalari tizimning funktsional dolzarbligi, ishonchliligi va moslashuvchanligini ta’minlashdan iborat. Dasturiy ta’minot evolyutsiyasi to‘liq qo‘lda (dasturiy ta’minot muhandislari tomonidan kiritilgan o‘zgarishlar asosida), qisman avtomatlashtirilgan (masalan, refaktoring vositalaridan foydalangan holda) yoki to‘liq avtomatlashtirilgan (avtonom sozlash yoki evolyutsiya bilan) bo‘lishi mumkin. Dasturiy ta’minotning rivojlanishiga Internet katta ta’sir ko‘rsatdi:

– World Wide Web va Internet resurslarining tez o‘sishi foydalanuvchilar va muhandislarga tegishli ma’lumotlarni topishni osonlashtiradi.

– har kim manba kodlarini yuklab olishi va shuning uchun ularni o‘zgartirishi mumkin bo‘lgan ochiq kodli ishlab chiqish tez va parallel evolyutsiyani ta’minladi.

15.2. Dasturiy ta’minotga xizmat ko‘rsatish turlari

E.B.Swanson dastlab xizmatning uchta toifasini aniqladi: tuzatuvchi, moslashuvchi va mukammal. Keyin Lientz va Swanson (1980) dasturiy ta’minotning to‘rtta toifasini katalogga kiritdilar. O‘shandan beri ular ISO/IEC 14764:2006 da yangilangan va xalqaro miqyosda standartlashtirilgan:

– **Tuzatuvchi xizmat ko‘rsatish**: aniqlangan muammolarni bartaraf etish uchun yetkazib berilgandan so‘ng dasturiy mahsulotni reaktiv modifikatsiya qilish;

– **Moslashuvchan xizmat ko‘rsatish**: o‘zgaruvchan yoki o‘zgaruvchan muhitda dasturiy mahsulotdan foydalanish imkoniyatini saqlab qolish uchun dasturiy mahsulotning yetkazib berishdan keyingi modifikatsiyasi;

– **Benuqson texnik xizmat ko‘rsatish**: samaradorlik yoki xizmat ko‘rsatish qobiliyatini yaxshilash uchun dasturiy mahsulotning yetkazib berishdan keyingi modifikatsiyasi;

– **Profilaktik xizmat ko‘rsatish**: dasturiy mahsulotdagi yashirin xatolarni haqiqiy xatoga aylanishidan oldin aniqlash va tuzatish uchun dasturiy ta’minot mahsulotini yetkazib berishdan keyingi modifikatsiyasi.

Yuqorida aytilganlarning barchasi ma’lum bo‘lgan o‘zgartirish zarurati mavjud bo‘lganda sodir bo‘ladi. Garchi bu toifalar Warren (1999), Chapin (2001), va boshqalar kabi ko‘plab mualliflar tomonidan kengaytirilgan bo‘lsa-da, xalqaro standart ISO/IEC 14764:2006 asosiy to‘rtta toifani saqlab qoldi. Yaqinda dasturiy ta’minotga xizmat ko‘rsatish va rivojlantirish tavsiflari ontologiyalar (Kitchenham (1999), Deridder (2002), Vizcaíno (2003), Dias (2003), va Ruiz (2004)) yordamida ifodalangan, bu ko‘plab evolyutsion jarayonlarning tavsifini boyitdi.

Joriy tendentsiyalar va usullar bosqichli model deb ataladigan yangi dasturiy ta’minot evolyutsiyasi modeli yordamida prognoz qilinadi. Bosqichli model zamonaviy dasturiy ta’minotni ishlab chiqish uchun unchalik mos bo‘lmagan va dasturiy ta’minot evolyutsiyasiga hissa

qo‘shish qiyin bo‘lgan murakkabliklari tufayli tez o‘zgarib turadigan an’anaviy tahlil o‘rniga joriy qilingan. Oddiy bosqichli modelda beshta alohida bosqich (dastlabki ishlab chiqish, rivojlantirish, xizmat ko‘rsatish, bosqichma-bosqich rad etish va yopish) mavjud.

– KHBennett va VT Rajlich fikrlariga ko‘ra, asosiy hissa "xizmat ko‘rsatish" bosqichini evolyutsiya fazalariga bo‘lish, keyin esa xizmat ko‘rsatish va bosqichma-bosqich rad etish bosqichlari hisoblanadi. Ba’zi funksiyalari mavjud bo‘lmagan dasturiy ta’minot tizimining birinchi versiyasi dastlabki ishlab chiqish jarayonida ishlab chiqiladi va alfa bosqichi deb ham ataladi. Biroq, ushbu bosqichda yaratilgan arxitektura kelajakda har qanday o‘zgartirish yoki tuzatishlarga duchor bo‘ladi. Ushbu bosqichdagi havolalarning aksariyati stsenariylarga yoki amaliy tadqiqotlarga asoslanadi. Olingan bilim dastlabki rivojlanishning yana bir muhim natijasi sifatida belgilanadi. Bunday bilimlar, jumladan domen haqidagi bilimlar, foydalanuvchi talablari, biznes qoidalari, siyosatlari, qarorlari, algoritmlari va boshqalar, dasturiy ta’minotni rivojlantirish uchun asos bo‘ladi. Bilim ham evolyutsiyaning keyingi bosqichi uchun muhim omil bo‘lib ko‘rinadi.

– Oldingi bosqich muvaffaqiyatli yakunlangandan so‘ng (va keyingi bosqichga o‘tishdan oldin uni muvaffaqiyatli yakunlash kerak), keyingi bosqich evolyutsiya bosqichi bo‘ladi. Foydalanuvchilar o‘z talablarini o‘zgartirib turadilar va shuningdek, ba’zi yaxshilanishlar yoki o‘zgarishlarni ko‘rishni afzal ko‘rishadi. Ushbu omil tufayli dasturiy ta’minot sanoati tez o‘zgaruvchan muhit muammolariga duch kelmoqda. Shu sababli, evolyutsiyaning maqsadi dasturni doimiy o‘zgaruvchan foydalanuvchi talablari va operatsion muhitga moslashtirishdir. Oldingi bosqichda yaratilgan ilovaning birinchi versiyasida ko‘plab xatolar bo‘lishi mumkin va bu xatolar amaliy tadqiqotlar yoki stsenariylar tufayli o‘ziga xos va aniq talablar asosida evolyutsiya bosqichida tuzatiladi.

– Dasturiy ta’minot evolyutsiya davridan chiqmaguncha doimiy ravishda rivojlanib boradi va undan keyin xizmat ko‘rsatish bosqichiga o‘tadi. Bu bosqich dasturiy ta’minotning yetukligi deb ham ataladi. Ushbu bosqichda faqat kichik o‘zgarishlar kiritiladi.

– Keyingi bosqich, ya’ni bosqichma-bosqich rad etish, endi ushbu maxsus dasturiy ta’minot uchun xizmat ko‘rsatmaydi. Biroq, dasturiy ta’minot hali ham ishlab chiqarish jarayonida bo‘ladi.

– Nihoyat, to‘xtatish bosqichi. Dasturiy ta’minotdan foydalanish o‘chiriladi yoki to‘xtatiladi va foydalanuvchilar almashtirishga yuboriladi.

15.3. Dasturiy ta’minot evolyutsiyasining Lehman qonunlari

1972 yildan 2002 yilgacha London Imperial kollejida ishlagan professor Meir M.Lehman[40] va uning hamkasblari dasturiy ta’minot evolyutsiyasidagi xatti-harakatlar to‘plamini aniqladilar. Ushbu xatti-harakatlar (yoki kuzatishlar) Leman qonunlari sifatida tanilgan. U E-tipli tizimlarga ishora qiladi, xuddi ular qandaydir haqiqiy ishlarni bajarish uchun yozilgan. Bunday tizimlarning xatti-harakati ular faoliyat ko‘rsatayotgan muhit bilan chambarchas bog’liq va bunday tizim ushbu muhitdagi o‘zgaruvchan talab va sharoitlarga moslashishi kerak. Sakkizta qonun quyidagilardan iborat:

1. (1974) "Doimiy o‘zgarish" - E tipidagi tizim doimo moslashishi kerak, aks holda u talablarni kamroq qoniqtiradigan bo‘lib qoladi.
2. (1974) "Murakkablikning ortishi" - E tipidagi tizim rivojlanib borar ekan, uni qo‘llab quvvatlash yoki murakkablikni kamaytirish uchun ish qilinmasa, uning murakkabligi ortadi.
3. (1980) "O‘z-o‘zini tartibga solish" - E tipidagi tizimning evolyutsiya jarayonlari mahsulotlarning taqsimlanishi va jarayon ko‘rsatkichlari normaga yaqin bo‘lgan holda o‘z-o‘zini tartibga soladi.
4. (1978) "Tashkiliy barqarorlikni saqlash (invariant ish darajasi)" - rivojlanayotgan E-tipdagi tizimda o‘rtacha samarali global faoliyat darajasi mahsulotning butun hayoti davomida o‘zgarmasdir.
5. (1978) "Bilimlarni saqlash" - elektron turdagi tizim rivojlanib borar ekan, u bilan bog’liq bo‘lgan barcha shaxslar, masalan, ishlab chiquvchilar, savdo xodimlari va foydalanuvchilar qoniqarli rivojlanishga erishish uchun uning mazmuni va xatti-harakatlarini o‘zlashtirishlari kerak. Haddan tashqari o‘sish bu mahoratni pasaytiradi. Shu sababli, o‘rtacha daromad tizim rivojlanishi bilan bir xil bo‘lib qoladi.
6. (1991) "Doimiy o‘sish" - elektron turdagi tizimning funktsional mazmuni uning butun hayoti davomida foydalanuvchi qoniqishini saqlab turish uchun doimiy ravishda kengayib borishi kerak.
7. (1996 y.) “Sifatning pasayib borishi” - E-tipli tizimning sifati, agar u ehtiyotkorlik bilan xizmat ko‘rsatilmasa va ish muhitidagi o‘zgarishlarga moslashtirilmasa, uning sifati yomonlashadi.
8. (1996) "Qayta aloqa tizimi" (birinchi marta 1974 yilda tuzilgan, 1996 yilda qonun sifatida rasmiylashtirilgan) - Elektron turdagi evolyutsiya jarayonlari ko‘p bosqichli, ko‘p davrli, ko‘p agentli qayta

[40] https://www.wikiwand.com/en/Manny_Lehman_(computer_scientist)

aloqa tizimlari bo‘lib, ular har qanday asosga qaraganda sezilarli yaxshilanishga erishish uchun shunday deb hisoblanishi kerak.

Shuni ta’kidlash kerakki, ushbu qonunlarning barcha turdagi dasturiy ta’minot tizimlari uchun qo‘llanilishi bir qancha tadqiqotchilar tomonidan o‘rganilgan. Misol uchun, Nanjangud C Narendra taqdimotiga qarang, u yerda u Lemanning dasturiy ta’minot evolyutsiyasi qonunlari nuqtai nazaridan korxona Agile loyihasi misolini tasvirlaydi.

Ochiq kodli dasturiy ta’minotni ishlab chiqishni o‘rganish bo‘yicha ba’zi empirik kuzatishlar ba’zi qonunlarga qarshi ko‘rinadi. Qonunlar dasturiy ta’minot tizimidagi funktsional o‘zgarishlar zarurati to‘liq yoki noto‘g’ri talablarni tahlil qilish yoki noto‘g’ri dasturlash natijasi emas, balki muqarrar ekanligini taxmin qiladi. Ular dasturiy ta’minotni ishlab chiqish guruhi o‘zgarishlar va yangi xususiyatlarni xavfsiz amalga oshirish nuqtai nazaridan erishish mumkin bo‘lgan cheklovlar mavjudligini ta’kidlaydilar.

Dasturiy ta’minot evolyutsiyasiga xos yetuklik modellari jarayonlarni takomillashtirish va dasturiy ta’minotni iterativ ravishda rivojlanib borishi bilan doimiy ravishda yangilanishini ta’minlash uchun ishlab chiqilgan. Ko‘pgina manfaatdor tomonlar (masalan, ishlab chiquvchilar, foydalanuvchilar, ularning menejerlari) amalga oshiriladigan "global jarayon" ko‘plab qayta aloqa zanjirlariga ega. Evolyutsiya tezligi teskari aloqa zanjirining tuzilishiga va global tizimning boshqa xususiyatlariga bog’liq.

Tizim dinamikasi kabi jarayonlarni modellashtirish usullari bunday global jarayonlarni tushunish va boshqarishda foydali bo‘lishi mumkin. Dasturiy ta’minot evolyutsiyasi darvincha, lamarkcha yoki boldvincha bo‘lishi dargumon, lekin o‘ziga xos muhim hodisadir. Jamiyat va iqtisodiyotning barcha darajalarida dasturiy ta’minotga tobora ortib borayotgan qaramlikni hisobga olgan holda, dasturiy ta’minotning muvaffaqiyatli evolyutsiyasi tobora muhim ahamiyat kasb etmoqda. Bu muhim tadqiqot mavzusi bo‘lib, unga yetarlicha e’tibor berilmagan.

Lehmanning ta’kidlashicha, dasturiy ta'minot evolyutsiyasi tabiati tabiiy o'zgarishlarni aks ettiradi, masalan, shaharlarning vaqt o'tishi bilan kengayishi, harbiy tuzilmalarning qurol tizimlarini bosqichma-bosqich takomillashtirishi va boshqalar. Jarayonning dastlabki uchta qonuni davom etayotgan o‘zgarishlarning detallashganligi, ortib borayotgan murakkabligi va yirik dasturlarning evolyutsiyasi deb ataladigan tendentsiyalarni taqlid qiladi. Davom etilayotgan o‘zgarishlar dasturning hozirgi real dunyo biznes sharoitlariga moslashishi kerakligini anglatadi

va bu tobora ortib borayotgan murakkablikni aks ettiradi, chunki dastur tobora ortib borayotgan turli xil kutilmagan ehtiyojlarni qondirishi kerak. Ajoyib dastur evolyutsiyasi bozor talablari bilan uzviy bog‘liq bo‘lgan xatolarni tuzatish va yangi dastur relizlariga bo‘lgan ehtiyojni anglatadi.

15-bob bo‘yicha xulosalar

Evolutsiya - bu dasturiy taminot hayot siklining shunday bosqichiki bunda u tezkor o‘zgarishda bo‘ladi va taklif qilingan yangi talablar bosqichma – bosqich shaklanadi, hamda sistemada amalga oshiriladi. Dasturiy ta’minot evolyutsiyasi - bu o‘zgaruvchan manfaatdor tomonlar va bozor talablariga javob beradigan dasturiy ta’minotning dastlabki chiqarilganidan keyin uzluksiz rivojlanishi. Dasturiy ta’minotning evolyutsiyasi muhim ahamiyatga ega, chunki tashkilotlar o‘z dasturiy ta’minotiga katta miqdorda mablag’ sarflaydi va to‘liq ushbu dasturiy ta’minotga bog’lanib qoladi. Dasturiy ta’minot evolyutsiyasi dasturiy ta’minotni o‘zgaruvchan biznes talablariga moslashishga, nuqsonlarni tuzatishga va dasturiy ta’minot tizimlari muhitida boshqa o‘zgaruvchan tizimlar bilan integratsiyalashuviga yordam beradi. Dasturiy ta’minot evolyutsiyasining asosiy vazifalari tizimning funktsional dolzarbligi, ishonchliligi va moslashuvchanligini ta’minlashdan iborat.

Dasturiy ta’minot evolyutsiyasiga xos yetuklik modellari jarayonlarni takomillashtirish va dasturiy ta’minotni iterativ ravishda rivojlanib borishi bilan doimiy ravishda yangilanishini ta’minlash uchun ishlab chiqilgan. Ko‘pgina manfaatdor tomonlar (masalan, ishlab chiquvchilar, foydalanuvchilar, ularning menejerlari) amalga oshiriladigan "global jarayon" ko‘plab qayta aloqa zanjirlariga ega. Evolyutsiya tezligi teskari aloqa zanjirining tuzilishiga va global tizimning boshqa xususiyatlariga bog’liq.

Ba’zi funksiyalari mavjud bo‘lmagan dasturiy ta’minot tizimining birinchi versiyasi dastlabki ishlab chiqish jarayonida ishlab chiqiladi va alfa bosqichi deb ham ataladi. Biroq, ushbu bosqichda yaratilgan arxitektura kelajakda har qanday o‘zgartirish yoki tuzatishlarga duchor bo‘ladi. Ushbu bosqichdagi havolalarning aksariyati stsenariylarga yoki amaliy tadqiqotlarga asoslanadi. Olingan bilim dastlabki rivojlanishning yana bir muhim natijasi sifatida belgilanadi. Bunday bilimlar, jumladan domen haqidagi bilimlar, foydalanuvchi talablari, biznes qoidalari, siyosatlari, qarorlari, algoritmlari va boshqalar, dasturiy ta’minotni

rivojlantirish uchun asos bo‘ladi. Bilim ham evolyutsiyaning keyingi bosqichi uchun muhim omil bo‘lib ko‘rinadi.

15-bob bo‘yicha nazorat savollari

1. Dasturiy maxsulotning evolutisiyasi deganda nimani tushunasiz?
2. Dasturiy maxsulotni qanday holatlarda o‘zgarishi muqarrardir?
3. Dasturiy ta’minotga servis xizmat ko‘rsatish haqida nimalar bilasiz?
4. Dasturiy maxsulot evolutsiyasi nimalarga bog’liq?
5. Dasturiy ta’minotning rivojlanishiga Internetning roli nimalardan iborat?
6. Dasturiy ta’minotga xizmat ko‘rsatish turlarini aytib bering.
7. Tuzatuvchi xizmat ko‘rsatish iborasi nimani anglatadi?
8. Moslashuvchan xizmat ko‘rsatish iborasi nimani anglatadi?
9. Profilaktik xizmat ko‘rsatish iborasi nimani anglatadi?
10. Dasturiy ta’minot evolyutsiyasining Lehman qonunlari aytib bering.

GLOSSARIY

Quyi darajadagi konstruktsiyalash - bu dasturiy ta’minot arxitekturasini yanada batafsil ishlab chiqish

Kodlash - bu dastur kodini yozish tartibi.

Murakkablikni pasaytirish - bu konstruktsiyalashdagi murakkablikni minimallashtirish, kamaytirish va alohida qismlarga bo‘lish.

Dasturlash — kompyuterlar va boshqa mikroprotsessorli kompyuterlar uchun dasturlar tuzish, sinash va o‘zgartirish jarayonidan iborat.

Arxitekturaviy konstruktsiyalash - tizimning eng yuqori abstract versiyasi.

Modullashtirish - bu dasturiy ta’minot tizimini vazifalarni mustaqil bajarishga qodir bo‘lgan bir nechta diskret va mustaqil modullarga bo‘lish usuli.

Dasturiy ta’minotning hayot davri modeli - hayot tsikli davomida bajarish ketma-ketligini va jarayonlar, harakatlar va vazifalarning o‘zaro bog’liqligini belgilaydigan tuzilma.

Algoritmning samaradorlik xususiyati - cheklangan sonli operatsiyalarni bajargandan so‘ng natija olish imkoniyatini anglatadi.

Algoritmning aniqlik xususiyati - foydalanuvchidan va qo‘llaniladigan texnik vositalardan qat’iy nazar olingan natijalar tasodifidan iborat.

Algoritmning diskretlik xususiyati - algoritm tomonidan belgilangan hisob-kitoblar jarayonini alohida bosqichlarga bo‘lish imkoniyati, ma’lum bir tuzilishga ega dastur bo‘limlarini tanlash qobiliyati.

Standartlashtirish – haqiqatda mavjud yoki potentsial vazifalarga nisbatan umumiy va ko‘p karrali foydalanish uchun qoidalarni belgilash vositasida muayyan sohada tartibga keltirishning optimal darajasiga erishishga qaratilgan ilmiy-texnik faoliyat.

Milliy standartlashtirish - bu muayyan bir mamlakat doirasida o‘tkaziladigan standartlashtirish faoliyatidir.

Komplekslilik – bu murakkab sistemadagi barcha elementlarning muvofiqligidir.

Standartlashtirishning samaradorligi – bu normativ hujjat qo‘llanilganda iqtisodiy yoki ijtimoiy samara berishidir, yahni standartlashtirishga sarflangan bir so‘mning o‘n so‘m foyda keltirishidir.

Standartlarni ishlab chiqishning ustivorligi - bu ustivorlik tovar va xizmatlarning o‘zaro muvofiqligi, almashuvchanligi va xavfsizligini tahminlashdir.

Uyg’unlashtirish tamoyili - bu xalqaro savdoda to‘siq bo‘lmaydigan uyg’unlashgan standartlarni ishlab chiqishdir.

Unifikatsiya - standartlashtirish shakli bo‘lib, ikki yoki undan ortiq hujjatlarni (texnik shartlarni) birlashtirishdir.

Klassifikatsiya- bu buyum, hodisa va tushunchalarni klasslar, razryadlarga, ularning umumiy belgilariga bog’liq holda joylanishidir.

Simplifikatsiya - standartlashtirishning shakli bo‘lib, buyumlarning turi va boshqa turli ko‘rinishlarining mikdorini qarayotgan vaqtdagi mavjud talablargacha kamaytirishdir.

Tipizatsiya - bu ko‘p sonli va ko‘p turdagi mashina, pribor, instrument va boshqalardan o‘z sifati bo‘yicha ratsional bo‘lgan va o‘z tartibida umumiy detallar bo‘lganlarini tanlashdir.

Selektsiya - standartlashtirish obyektlarini tanlash**.**

Optimallashtirish - bu optimal bosh parametrlarni hamda sifat va tejamkorlikning boshqa barcha kiymatlarini topishdir.

Agregatlash - bu alohida unifikatsiyalashgan standart uzellaridan mashina, pribor va jihozlarni yaratish usulidir.

Metrologiya – O‘lchashlar, ularning birliligini ta’minlash metodlari va vositalari va talab etilgan aniqlikka erishish usullari to‘g’risidagi fan.

O‘lchashlar birliligi – O‘lchashlarning natijalari qonunlashtirilgan birliklarda ifodalangan va o‘lchashlarning xatoliklari berilgan ehtimollik bilan ma’lum bo‘lgan holat.

O‘lchash – maxsus texnik vositalar yordamida fizik kattaliklar qiymatlarini tajriba yo‘li bilan topish.

O‘lchash vositasi – normalashtirilgan metrologik tavsifga ega bo‘lgan o‘lchash asbobidir.

O‘lchash asbobi – kuzatuvchi idrok qilishi uchun qulay shakldagi o‘lchov informasiyasi signalini ishlab chiqishga xizmat qiladigan o‘lchash vositasi.

O‘lchov – berilgan o‘lchamli fizik kattaliklarni qayta tiklash uchun mo‘ljallangan o‘lchash vositasi.

Birlik etaloni – Fizik kattalikning o‘lchamini boshqa o‘lchash vositasiga berish maqsadida fizik kattalik birligining o‘lchamini qayta tiklash va saqlash uchun mo‘ljallangan o‘lchashlar vositasi.

Davlat etaloni – O‘zbekiston Respublikasi hududida kattalik

birligining o‘lchamini o‘rnatish uchun milliy idora vakilining qarori bilan boshlang’ich ztalon sifatida tan olingan etalon.

O‘lchash o‘zgartkichi - o‘lchashga doir axborotni uzatish, o‘zgartirish, ishlov berish va saqlash uchun qulay bo‘lgan, ammo kuzatuvchi bevosita idrok qilishi mumkin bo‘lmaydigan shakldagi signalni ishlab chiqish uchun xizmat qiladigan o‘lchash vositasidir.

Etalonlar deb, fan va texnikaning eng yuksak saviyasida aniqlik bilan ishlangan namunaviy o‘lchovlarga aytiladi.

O‘lchov birligi o‘lchash natijasi ko‘rsatilgan birlikda ifodalangan va o‘lchash xatoligi berilgan ehtimollikda ma’lum bo‘lgan o‘lchash holatidir.

O‘lchash aniqligi – bu o‘lchash natijalarini va o‘lchanayotgan kattalik haqiqiy qiymatining mos kelish darajasidir.

O‘lchash xatoligi o‘lchash natijasining o‘lchanayotgan kattalikning asl qiymatidan farqlanishidir.

Qonunlashtiruvchi metrologiya – bu metrologiyaning1 bir qismi bo‘lib metrologiya bo‘yicha milliy organ tomonidan amalga oshiriladigan faoliyatga taalluqli va birliklar, o‘lchashvositalari, o‘lchash laboratoriyalariga doir davlat talablariga ega.

Metrologiya xizmati – Davlat idoralari va yuridik shaxslarning metrologik xizmatlari tarmoqlari hamda ularning o‘lchashlar birliligini ta’minlashga yo‘naltirilgan faoliyati.

O‘lchash vositalarining qiyoslanishni olib borish uchun metrologik xizmatni akkreditlash – bu qiyoslashishlarini bajarishga davlat tomonidan, davlatning ishonchli vakili tomonidan rasmiy tan olinishi.

O‘lchash vositasini qiyoslash usuli – bu qiyoslash sxemasi bo‘yicha yuqoridagi o‘lchash vositalaridan quyidagi o‘lchash vositalariga birlikning o‘lchamini uzatish usuli.

O‘lchash vositalarining qiyoslash natijalarini rasmiylashtirish - bu o‘lchash vositalarini qiyoslash natijalari bo‘yicha rasmiy hujjatni tuzish va o‘lchash vositasini yaroqliligini ko‘rsatish.

O‘lchash vositalarini taqqoslash – bu sistematik xatoliklarni aniqlash uchun o‘lchash vositasini etalon yoki o‘sha turdagi namuna o‘lchash vositasiga solishtirish, ya’ni o‘lchovni o‘lchov bilan, priborni pribor bilan.

Bazaviy (asos) metrologiya xizmati – bu aloqa va axborotlashtirish sohasidagi xizmat bo‘lib, biriktirilgan xo‘jalik yurituvchi sub’ektlarning metrologik ta’minot masalalari bo‘yicha ish

faoliyatini muvofiqlashtiruvchi xizmat.

Davlat metrologiya nazorati – bu o‘lchash vositalarining turi va qiyoslanishi, sotilishi va ularning prokatini lisenziyalash bo‘yicha davlat metrologiya xizmati organi tomonidan amalga oshiriladigan faoliyatdir.

Davlat metrologiya tekshiruvi - bu davlat metrologiya xizmati organi tomonidan amalga oshiriladigan metrologiya qoidalariga rioya qilinishi tekshirish maqsadidagi faoliyatdir.

O‘lchanadigan kattalik - o‘lchashga tortiladigan kattalik.

O‘lchash vositasini kalibrlash - bu kalibrlash laboratoriyasi tomonidan o‘lchash vositasining metrologik xarakteristikasi haqiqiy qiymatlarini va qo‘llanilishga yaroqliligini aniqlash va tasdiqlash maqsadidagi muolajalar majmuidir.

O‘lchash vositalarini qiyoslash – O‘lchash vositalarining o‘rnatilgan texnik talablarga muvofiqligini aniqlash va tasdiqlash maqsadida davlat metrologik xizmat idoralari (boshqa vakolatlangan idoralar, tashkilotlar) bajaradigan amallar majmui.

O‘lchash vositalarini tayyorlash (ta’mirlash, sotish, ijaraga berish) ga lisenziya – Ko‘rsatilgan faoliyat turlari bilan shug’ullanishga huquqini tasdiqlovchi, yuridik va jismoniy shaxslarga davlat metrologik xizmat idoralari tomonidan beriladigai hujjat.

O‘lchash vositalarini metrologik attestatlash – donalab ishlab chiqarilgan (yoki O‘zbekiston hududiga donalab keltirilgan) o‘lchash vositalarining, ularning xossalarini sinchiklab tadqiq etish asosida, qo‘llanishga huquqli ekanligini metrologik xizmat tomonidan tan olish.

Metrologik xizmatlar, markazlar, laboratoriyalarni akkreditlash –metrologik xizmatlar, markazlar, laboratoriyalarning o‘rnatilgan akkreditlash doirasida o‘lchashlar birliligini ta’minlash bo‘yicha ishlarni o‘tkazishga huquqligini rasmiy tan olish.

Yuridik shaxslarning metrologik xizmatlarini o‘lchash vositalarini kalibrlash huquqiga akkreditlash – yuridik shaxslar metrologik xizmatlarining o‘rnatilgan doirada o‘lchash vositalarini kalibrlashni o‘tkazish huquqini rasmiy tan olish.

O‘lchashlarni bajarish metodikalarini metrologik attestatlash – O‘lchashlarni bajarish metodikasining unga qo‘yilgan metrologik talablarga muvofiqligini baholash va tasdiqlash maqsadida o‘tkaziladigan tadqiqot.

O‘lchashlarni bajarish metodikasi – O‘lchashlar natijalariga avvaldan ma’lum bo‘lgan xatolik bilan erishishni ta’minlaydigan ishlar va qoidalar majmui.

O‘lchash noaniqligi – Tekshirish- bu belgilangan talablarni to‘laqonli ta’minlash uchun obyektiv dalillar yordamida ko‘rikdan o‘tkazishni ta’minash.

Murakkablikni lokalizatsiya qilish - bu ob’ektga yo‘naltirilgan yondashuvdan foydalangan holda konstruktsiyalashning usuli.

Dasturiy ta’minotning murakkabligi - bu dasturlashning ajralmas xususiyati.

Refaktoring – bu dasturning tashqi xulq-atvoriga ta’sir qilmasdan ichki strukturasini o‘zgartirish jarayonidir.

Dasturiy ta’minot reinjiniringi - bu ishlayotgan dasturiy ta’minotdan foydalanib, yangi funksionallikni yaratish yoki katta o‘zgarishlar orqali xatolarni yo‘q qilish jarayoni.

Konstruktsiyalarni boshqarish – konstruktsiyaga qo‘yilayotgan talablarni qondirish uchun bilim, tajriba va usullardan foydalanish.

Metodologiya – dasturiy tizimlarni yaratish va umumiy falsafiy birlashtirishda foydalaniladigan mexanizmlar to‘plami.

Metod – konseptual tushunchalar, bazaviy notatsiya, ushbu tushunchlarning grafik ma’nosi va modellarni qurishni qoidalari, shuningdek loyihalashtirish va ishlab chiqish jarayoni.

Metodika – ma’lum metod asosida qurilidagian dasturiy tizimni qurishning qadamalarining yetarlicha tushunarli berilishi.

Dasturiy ta’minotni testlash (Software Testing) - dasturning real va kutilgan natijalari orasidagi moslikni ma’lum tartibda tanlangan chekli test to‘plami (sun’iy tarzda tashkil etilgan vaziyatlar) asosida tekshirish.

Tekshirish (Verification) - bu tizimni yoki uning komponentlarini joriy bosqich boshida shakllantirilgan talablarga qanoatlantirishini baholash jarayonidir.

Validasiya (Validation) - bu ishlab chiqilgan DTning kutilgan natijalarga va foydalanuvchi extiyojlariga, tizim talablariga mosligini aniqlash.

Testlashni rejalashtirish (Test Plan) - bu testlash bo‘yicha bajarilishi kerak bo‘lgan ishlarni qamrab olgan xujjat.

Testni Qoplanishi (**Test Coverage**) – bu testlash sifatini baholash mezonlaridan biri, bajariladigan kodning yoki talablarni testlar yordamida qoplanishidir.

Dasturiy ta’minotni testlash darajalari deganda uning alohida modul ustida, bir guruh modullar yoki butun tizim ustida tekshirilishi tushuniladi.

FOYDALANILGAN ADABIYOTLAR RO‘YXATI

1. M. H. Aripova, M. H. Qayumova, M. Z. Babamuxamedova. Tizimli dasturiy ta'minot: o‘quv qo‘llanma. - T.: TATU, 2016.
2. Ш. А. Назиров, Р. В. Кабулов, С. К. Уринбоев. Системное программное обеспечение: Методическое пособие для практических занятий. ТУИТ. - Т. : ТАТУ, 2008.
3. Романов А.А. Конструирование программного обеспечения: учебное пособие. – Ульяновск: УлГТУ, 2016.
4. Данилкин Ф.А., Сычугов А.А. Конструирование программного обеспечения: учебное пособие. Изд-во ТулГУ, 2010.
5. Милованов И.В., Лоскутов В.И. Основы разработки программного обеспечения вычислительных систем: учебное пособие. – Тамбов: Изд-во ГОУ ВПО ТГТУ, 2011.
6. Patton, Ron. Software Testing: монография / R. Patton. - 2th. ed. - Indiana : SAMS Publishing, 2006.
7. Зубкова Т.М. Технология разработки программного обеспечения. Учебное пособие. — Оренбург: Оренбургский государственный университет, 2017.
8. Гагарина Л.Г. Технология разработки программного обеспечения. / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул. – М.: «ФОРУМ»: ИНФРА-М, 2009.
9. Брауде Дж. Технология разработки программного обеспечения. СПб.: Питер, 2004.
10. Орлов С.А. Программная инженерия. Учебник для вузов. 5-е издание обновленное и дополненное. – СПб.: Питер, 2016.
11. Фаулер М. Рефакторинг: улучшение существующего кода. - Пер. с англ. - СПб: Символ Плюс, 2003.
12. Г. А. Лисьев, П. Ю. Романов, Ю. И. Аскеров. Программное обеспечение компьютерных сетей и Web-серверов: учебное пособие / - М. : ИНФРА-М, 2019.
13. Маккарти, Джим. Правила разработки программного обеспечения: монография / Пер. с англ. - М. ; СПб ; Нижний Новгород : Русс. редакция; Питер, 2007.
14. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие для вузов–Москва: ФОРУМ: ИНФРА-М, 2011. - (Высшее образование).
15. Соммервилл, И. Инженерия программного обеспечения. пер. с англ. - 6-е изд. - Москва: Вильямс, 2002.

16. С.В. Назаров. Архитектура и проектирование программных систем: Монография - М.: НИЦ Инфра-М, 2013.
17. Б.В. Черников. Управление качеством программного обеспечения: Учебник - М.: ИД ФОРУМ: ИНФРА-М, 2012.
18. В.А. Гвоздева, И.Ю. Лаврентьева. Основы построения автоматизированных информационных систем: Учебник. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013.
19. Стив Макконнелл. Совершенный код = Code complete. — М.: Русская Редакция, 2010.
20. Э.Гамма, Р.Хелм, Р.Джонсон, Д.Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования /. – СПб.: Питер, 2009. – 366 с.
21. Джейсон Мак-Колм Смит Элементарные шаблоны проектирования : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2013. — 304 с.
22. Влиссидес Джон. Применение шаблонов проектирования. Дополнительные штрихи. : Пер. англ. М.: Издательский дом «Вильямс.
23. Тепляков С. В. Паттерны проектирования на платформе .NET – СПб.: Питер, 2015. – 320 с.
24. Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов.: Пер. с англ. М.: ДМК Пресс, 2002.
25. Буч Г., Рамбо Д., Джекобсон А. Язык UML Руководство пользователя — С-П.: Издательство «Питер», 2010 — 432 с.
26. Мартин Р. Чистый код. Создание, анализ и рефакторинг. Библиотека программиста. — СПб.: Питер, 2014. — 464 с.
27. Фаулер М. Рефакторинг. Улучшение существующего кода/ М. Фаулер. Символ-Плюс, 2008. 432 с.
28. Аллен Э. Типичные ошибки проектирования / Э.Аллен: Пер. с англ. – СПб.: Питер, 2003. – 224 с.
29. Чакон Скотт, Страуб Бен Git для профессионального программиста. — СПб.: Питер, 2016. — 496 с.: ил.
30. Брукс, Ф. Мифический человеко-месяц или как создаются программные системы / Ф. Брукс. — М.: СПб: Символ-Плюс, 2016. — 304 с.
31. С. Макконнелл Сколько стоит программный проект – СПб.: Питер – 2007.

32. Орлов С.А. Цилькер Б.Я. Технологии разработки программного обеспечения: современный курс по программной инженерии: учебник для вузов. — 4-е изд.— СПб: Питер, 2012.
33. Дубейковский, В. И. Эффективное моделирование с CA ERwin® Process Modeler: BPwin; AllFusion Process Modeler : практическое – 2-е изд., испр. и доп. – Москва : Диалог-МИФИ, 2009.
34. Кролл П., Крачтен Ф. Rational Unified Process - это легко. Руководство по RUP для практиков. Пер. с англ. - М.: КУДИЦ-ОБРАЗ, 2004 г.
35. Ивутин А. Н. , Волошко А. Г. Методология Agile для проектирования и сопровождения программных систем: учебное пособие. 2021. Издательство: ТулГУ.
36. Benjamin J. J. Voigt. Dynamic System Development Method. Department of Information Technology University of Zurich. 2004.

MUNDARIJA