# Ixtiyoriy dasturiy ta’minot tizimini modellash uchun UML (Unified Modeling Language) - birlashgan modellashtirish tilidan foydalanish

**Ishdan maqsad:** UML (Unified Modeling Language) - birlashgan modellashtirish tili ishlatish ko’nikmasiga ega bo’lish.

# UML (Unified Modeling Language) - birlashgan modellashtirish tili

UML (Unified Modeling Language) - birlashgan modellashtirish tili dasturiy ta’minot tizimlarini modellashda 14 ta turli xil diagramma turlaridan foydalanadi. UML dasturiy ta’minot tizimlarini modelini yaratishda standart yondashuv deb qabul qilingan. UML - bu ob'ektga yo'naltirilgan dasturiy ta'minot dizaynida ishlatiladigan modellashtirish tili. UML dasturiy ta'minot tizimini tashkil etuvchi tarkibiy qismlarni aniqlash va vizualizatsiya qilish imkoniyatini beradi. UML diagrammalari asosan tizimning ko'rinishi va xulq-atvor ko'rinishini aks ettiradi. Tizimning strukturaviy ko'rinishi klass diagrammasi, kompozitsion tuzilish diagrammasi va boshqalar kabi diagrammalar yordamida taqdim etilgan. Tizimning dinamik ko'rinishi ketma-ketlik diagrammasi, faoliyat diagrammasi va boshqalar kabi diagrammalardan foydalangan holda taqdim etilgan. UML-ning 2.2-versiyasi o'n to'rtta diagrammalarni o'z ichiga oladi, ular uchun yettita asosiy diagramma mavjud. tarkibiy ko'rinishni va xulq-atvor ko'rinishini ifodalovchi boshqa ettitani. Xatti-harakatlarning yettita diagrammasidan tizim bilan o'zaro munosabatlarni ifodalash uchun to'rtta diagramma ishlatilishi mumkin. UML modellashtirish uchun ishlatilishi mumkin bo'lgan vositalar mavjud, masalan, IBM Rational Rose. Ushbu til keng profilga ega va ochiq standart bo'lib, unda tizimning mavhum modelini yaratish uchun turli xil grafik belgilaridan foydalaniladi. UML dasturiy ta'minot tizimlarining barcha turlarini aniqlash, vizualizatsiya, hujjatlashtirish va dizaynni ta'minlash maqsadida yaratilgan. Ta'kidlash joizki, UML diagrammasining o'zi dasturlash tili emas, balki uning asosida alohida kod yaratish imkoniyatini ham beradi. UML-dan foydalanish barcha turdagi dasturlarni modellashtirish bilan tugamaydi. Shuningdek, bugungi kunda ushbu til turli xil biznes jarayonlarini modellashtirish, tizim dizaynini yuritish, shuningdek, tashkiliy tuzilmalarni namoyish qilishda faol foydalanilmoqda. UML-dan foydalangan holda dasturiy ta'minot ishlab chiquvchilari komponentlar, umumlashtirish, sinflar, xulq- atvor va umumlashtirish kabi umumiy tushunchalarni ifodalash uchun ishlatiladigan grafik konventsiyalar to'g'risida to'liq kelishuvni ta'minlashlari mumkin. Shu tufayli arxitektura va dizaynda ko'proq kontsentratsiyaga erishiladi.

# UML tarixi.

Ob'ektga yo'naltirilgan dasturlash tillari 70-yillarning o'rtalaridan 80- yillarning oxirigacha paydo bo'ldi, tadqiqotchilar ob'ektga yo'naltirilgan dasturlash tillarining yangi imkoniyatlarini va tobora murakkablashib borayotgan dasturlarning talablarini hisobga olish zarurati bilan duch kelganlarida, turli xil alternativ yondashuvlarni ishlab chiqishga majbur bo'ldilar. 1989 yildan 1994 yilgacha turli xil ob'ektga yo'naltirilgan usullar soni o'ndan ellikdan oshdi. Biroq, ko'plab foydalanuvchilar o'zlarining ehtiyojlarini to'liq qondiradigan modellashtirish tilini tanlashda qiynaldilar, bu esa "usul urushi" deb nomlandi.

Ushbu urushlar natijasida yangi avlod usullari paydo bo'ldi, ular orasida Grady Booch tomonidan yaratilgan Booch tillari, Ivar Jakobson tomonidan ishlab chiqilgan OOSE (Object-oriented software engineering) (Ob'ektga yo'naltirilgan dasturiy ta'minot muhandisligi) va OMT (Ob'ektlarni modellashtirish texnikasi) bu Jeyms Rumbaugh tomonidan ishlab chiqilgan. Bundan tashqari, Fusion, Shlaer- Mellor va Coad-Yourdon tomindan ham shunday tillar yaratildi. Ushbu usullarning har birini to'liq yaxlit va to'liq deb hisoblash mumkin, garchi ularning har biri nafaqat kuchli, balki kuchsiz tomonlariga ham ega edi. Booch usulining ekspresiv imkoniyatlari modelni loyihalashtirish va qurish bosqichlarida eng muhimlaridan biridir. OOSE talablarni tahlil qilish va shakllantirish hamda yuqori darajadagi dizayni uchun juda mos keladi. OMT-2 katta hajmdagi ma'lumotlarni qayta ishlashga yo'naltirilgan axborot tizimlarini tahlil qilish va rivojlantirish uchun juda foydali ekanligini isbotladi. 1990-yillarning o'rtalarida, Grady Booch (Rational Software Corporation), Ivar Jacobson (Objectory) va Jeyms Rambeau (General Electric) o'zlarining usullarini birlashtirishga urinishganida, yangi g'oyalarning tanqidiy massasi shakllana boshladi. Booch, OOSE va OMT tillarining asosiy mualliflari sifatida sheriklar yangi, birlashtirilgan modellashtirish tilini yaratishga harakat qildilar va uchta fikrga asoslanishdi.

Birinchisi, ishlab chiquvchilarning istaklaridan qat'i nazar, uchta usul ham teskari yo'nalishda rivojlangan. Ushbu evolyutsiyani alohida-alohida emas, balki birgalikda davom ettirish oqilona edi, bu kelajakda kiruvchi farqlarni va natijada foydalanuvchilar uchun noqulayliklarni bartaraf etishga yordam beradi.

Ikkinchidan, usullarni birlashtirib, barcha loyihalarni yagona yetuk tilda asoslashga imkon beradigan va asboblar yaratuvchilarga samaraliroq faoliyatga e'tibor qaratishlariga imkon beradigan ob'ektiv yo'naltirilgan modellashtirish vositalari uchun bozorga barqarorlikni olib kelish osonroq edi.

Uchinchisi, bunday hamkorlik har uchala usulni takomillashtirishga olib keladi va muammolarning yechimini taklif qiladi, ular uchun ajratilgan holda qabul qilingan har ikkisi ham juda mos emas edi.

Birlashtirishni boshlagan mualliflar uchta asosiy maqsadni qo'ydilar:

* ob'ektga yo'naltirilgan usullardan foydalangan holda, tizimni kontseptsiyadan tortib bajariladigan artefaktgacha bir butun sifatida modellashtirish;
* muhim vazifalarni bajarish uchun mo'ljallangan murakkab tizimlarga xos bo'lgan miqyosi masalasini hal qilish;
* nafaqat odamlar, balki kompyuterlar ham foydalanishi mumkin bo'lgan modellashtirish tilini yaratish.

Ob'ektga yo'naltirilgan tahlil qilish va loyihalash uchun til ixtiro qilinishi dasturlash tilini ishlab chiqilishidan unchalik farq qilmaydi. Birinchidan, vazifani cheklash talab qilingan. Til talablarni belgilash qobiliyatini o'z ichiga olishi kerakmi? Til vizual dasturlashga ruxsat berishi kerakmi? Ikkinchidan, ta'sirchan kuch va soddalik o'rtasidagi muvozanatni topish kerak edi. Juda sodda til uning yechishi mumkin bo'lgan vazifalar doirasini cheklaydi va juda murakkab narsa tajribasiz ishlab chiquvchini mag'lub qilishi mumkin. Bundan tashqari, mavjud usullarni birlashtirganda, ularning yordamida allaqachon ishlab chiqarilgan

mahsulotlarning mavjudligini hisobga olish kerak edi. Juda ko'p o'zgarishlar mavjud foydalanuvchilarni chetlashtirishi mumkin va tilning rivojlanishiga qarshilik ko'rsatish orqali mualliflar yangi foydalanuvchilarni jalb qilish va tilni osonroq va qulayroq qilish qobiliyatini yo'qotadi. UML ni yaratishda ishlab chiquvchilar ushbu muammolarning eng yaxshi yechimini topishga harakat qilishdi.

UML ning rasmiy yaratilishi 1994 yil oktyabr oyida, Rambeau Booch ishlagan Rational Softwarega o'tganidan so'ng boshlandi. Asl maqsad Booch va HTM usullarini birlashtirish edi. Umumiy metodning birinchi sinov versiyasi 0.8, o'sha paytda u 1995 yil oktyabrida paydo bo'lgan. Aynan shu vaqtlarda Jeykobson Rational ga qo'shildi va UML loyihasi OOSE ga qo'shildi. Birgalikdagi sa'y- harakatlar natijasida UML ning 0.9 versiyasi 1996 yil iyun oyida chiqdi. Yil davomida ijodkorlar dasturiy ta'minot muhandisligi sohasida ishlaydigan yirik kompaniyalarning mulohazalarini to'plashdi. Shu vaqt ichida ushbu kompaniyalarning aksariyati UMLni o'z bizneslari uchun strategik ahamiyatga ega deb hisoblashlari aniq bo'ldi. Natijada, UMLning konsortsiumi tashkil etildi, uning tarkibiga UMLni to'liq ta'riflash uchun ishlash uchun resurslarni qo'shishga tayyor tashkilotlar kirdi.

Tilning 1.0-versiyasi - bu Digital Equipment Corporation, Hewlett Packard, I-Logix, Intellicprp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational, Texas Instruments va Unisys kompaniyalari o'rtasidagi hamkorlikdagi ishi bo’ldi. UML 1.0 juda aniq belgilangan, ifodali, kuchli til bo'lib, har xil vazifalarga mos keladi. 1997 yil yanvar oyida u standart modellashtirish tilini yaratish bo'yicha tanlov uchun ob'ektlarni boshqarish guruhiga (OMG- Object Management Group) taqdim etildi.

1997 yil yanvar va iyun oylari orasida UML konsortsiumi OMGning chaqirig'iga javob beradigan deyarli barcha kompaniyalarni, ya'ni Andersen Consulting, Ericsson, ObjecTime Limited, Platinum Technology, Ptech, Reich Technologies, Softeam, Sterling Software va Taskon kompaniyalarini qamrab oldi. UML spetsifikatsiyalarini rasmiylashtirish va boshqa standartlashtirish guruhlari bilan muvofiqlashtirish uchun MCI Systemhouse dan Kris Kobrin va Rational-dan Ed Eyxolt boshchiligida Semantik guruh tashkil etildi. UML (1.1) ning qayta ko'rib chiqilgan versiyasi 1997 yil iyul oyida yana OMGga taqdim etildi. Sentyabr oyida versiya Tahlil va dizayn guruhi va OMG Arxitektura qo'mitasi yig'ilishlarida tasdiqlandi va 1997 yil 14 noyabrda barcha OMG a'zolarining umumiy yig'ilishida standart sifatida qabul qilindi.

UMLni rivojlantirish bo'yicha keyingi ishlar OMG Revision Task Force (RTF) tomonidan Kris Kobrin boshchiligida amalga oshirildi. 1998 yil iyun oyida UML 1.2, 1998 yil kuzida - UML 1.3 chiqdi.

# UML ning foydali va kamchilik tomonlari.

UML foydalanish diagrammasini va boshqalarni ajratib turadigan bir qator afzalliklari: Til ob'ektga yo'naltirilgan bo'lib, natijada tahlil va dizayn natijalarini tavsiflash texnologiyalari zamonaviy turdagi barcha turdagi ob'ektlarga yo'naltirilgan tillarda dasturlash usullariga semantik ravishda yaqinlashadi. Ushbu tildan foydalanib, tizimni har qanday mumkin bo'lgan nuqtai nazardan tavsiflash

mumkin, va uning hatti-harakatlarining turli jihatlari xuddi shu tarzda tasvirlangan. Barcha sxemalarni uning sintaksisiga nisbatan tezroq nazar tashlaganingizdan so'ng ham o'qish oson. UML sizga nafaqat dasturiy ta'minot muhandisligida foydalanishga hissa qo'shadigan o'z grafikangiz va matnli stereotiplaringizni kengaytirish, shuningdek tanitish imkonini beradi. Til juda keng tarqalgan va ayni paytda faol rivojlanmoqda.

**Kamchiliklari:** UML diagrammalarining qurilishi o'zining afzalliklari massasi bilan ajralib turishiga qaramay, ko'pincha quyidagi kamchiliklar tanqid qilinadi: Ishlarning aksariyat qismida tanqidchilar UML juda katta va murakkab ekanligini ta'kidlashadi va ko'pincha bu asossizdir. U juda ko'p keraksiz yoki deyarli foydasiz inshootlar va diagrammalarni o'z ichiga oladi va ko'pincha ushbu tanqid birinchi versiyaga emas, balki ikkinchi versiyaga qaratiladi, chunki yangi nashrlarda "qo'mita tomonidan ishlab chiqilgan" ko'proq murosalar mavjud. Semantikada turli xil noaniqliklar. UML o'zi, ingliz va OCL kombinatsiyasi bilan aniqlanganligi sababli, rasmiy ta'riflash texnikasi bilan aniq belgilangan tillarga xos bo'lgan cheklov mavjud emas. Ba'zi holatlarda OCL, UML va ingliz mavhum sintaksisi bir-biriga zid kela boshlaydi, boshqa hollarda ular to'liq emas. Tilning tavsifining noaniqligi foydalanuvchilarda ham, vositalarni yetkazib beruvchilarda ham teng ravishda namoyon bo'ladi, natijada turli xil spetsifikatsiyalarni izohlashning noyob usuli tufayli vositalarning nomuvofiqligiga olib keladi. Amalga oshirish va o'rganish jarayonida muammolar. Yuqoridagi barcha muammolar UMLni joriy qilish va o'rganish jarayonida ma'lum qiyinchiliklarni keltirib chiqaradi, ayniqsa bu menejment muhandislari oldindan ko'nikmalarga ega bo'lmagan holda undan zo'rlik bilan foydalanishga majbur qiladigan holatlarga taalluqlidir. Kod kodni aks ettiradi. Yana bir fikr shundaki, ahamiyat go'zal va jozibali modellar uchun emas, balki ishlaydigan tizimlarning o'zlari, ya'ni kod loyihadir. Ushbu fikrga ko'ra, dasturlarni yozishning yanada samarali usulini ishlab chiqish kerak. UML odatda bajariladigan yoki dastlabki kodni tiklash uchun modellarni tuzishda yondashuvlar uchun qadrlanadi. Aslida, bu yetarli bo'lmasligi mumkin, chunki bu tilda turing to'liqligi xususiyatlari mavjud emas va har bir yaratilgan kod oxir-oqibat UML vositasi tarjima qiladigan vosita taklif qilishi yoki aniqlay oladigan narsalar bilan cheklanadi. Mos kelmaslik. Ushbu atama tizimlarni tahlil qilish nazariyasidan ma'lum bir tizim kirishining boshqacha natijani idrok eta olmasligini aniqlash uchun keladi. Har qanday standart notation tizimida bo'lgani kabi, UML ba'zi tizimlarni boshqalariga qaraganda samaraliroq va ixcham tarzda namoyish qilishi mumkin. Shunday qilib, ishlab chiqaruvchi UML va boshqa dasturlash tillarining barcha kuchli tomonlarini o'zaro solishtirish uchun yanada qulay bo'lgan yechimlarga moyil. Agar rivojlanish tili ob'ektga yo'naltirilgan pravoslav doktrinaning asosiy printsiplariga mos kelmasa, ya'ni OOP tamoyillariga muvofiq ishlashga harakat qilmasa, bu muammo yanada ravshan bo'ladi. Umumjahon bo'lishga harakat qilish. UML har qanday mavjud ishlov berish tili bilan moslikni ta'minlashga intiladigan umumiy maqsadli modellashtirish tilidir. Muayyan loyiha kontekstida, dizayn jamoasi yakuniy maqsadga erishishi uchun ushbu tilning tegishli xususiyatlarini tanlashingiz kerak. Bundan tashqari, ma'lum

bir hududda UML dan foydalanish ko'lamini cheklashning mumkin bo'lgan usullari to'liq shakllanmagan, ammo tanqid ob'ekti bo'lgan rasmiyatchilikdan o'tadi.

**UML diagrammalari.**UML sinf diagrammasi bu tizimning tuzilishini tavsiflash uchun, shuningdek, bir necha xil sinflar orasidagi atributlar, usullar va bog'liqlikni namoyish qilish uchun mo'ljallangan statik strukturaviy diagramma. Shuni ta'kidlash kerakki, bunday diagrammalarning tuzilishida ularning qanday ishlatilishiga qarab bir nechta nuqtai nazar mavjud. UML diagrammalarining bir nechta turlari mavjud va ularning har biri amalga oshirishdan oldin yoki undan keyin (hujjat qismi sifatida) ishlab chiqilganligidan qat'i nazar, har xil maqsadga xizmat qiladi. Boshqa barcha turlarni o'z ichiga olgan ikkita eng keng toifalar - bu xatti-harakatlarning Xulq-atvor UML diagrammasi va Stukturaviy UML diagrammasi. Nomidan ko'rinib turibdiki, ba'zi UML diagrammalar tizim yoki jarayonning tuzilishini tahlil qilishga va tasvirlashga harakat qilsa, boshqalari tizimning xatti-harakatlarini, uning aktyorlari va qurilish tarkibiy qismlarini tavsiflaydi. Ular quyidagicha taqsimlanadi:

# Xulq-atvor UML diagrammaslari:

* Activity Diagram
* Use Case Diagram
* Interaction Overview Diagram
* Timing Diagram
* State Machine Diagram
* Communication Diagram
* Sequence Diagram

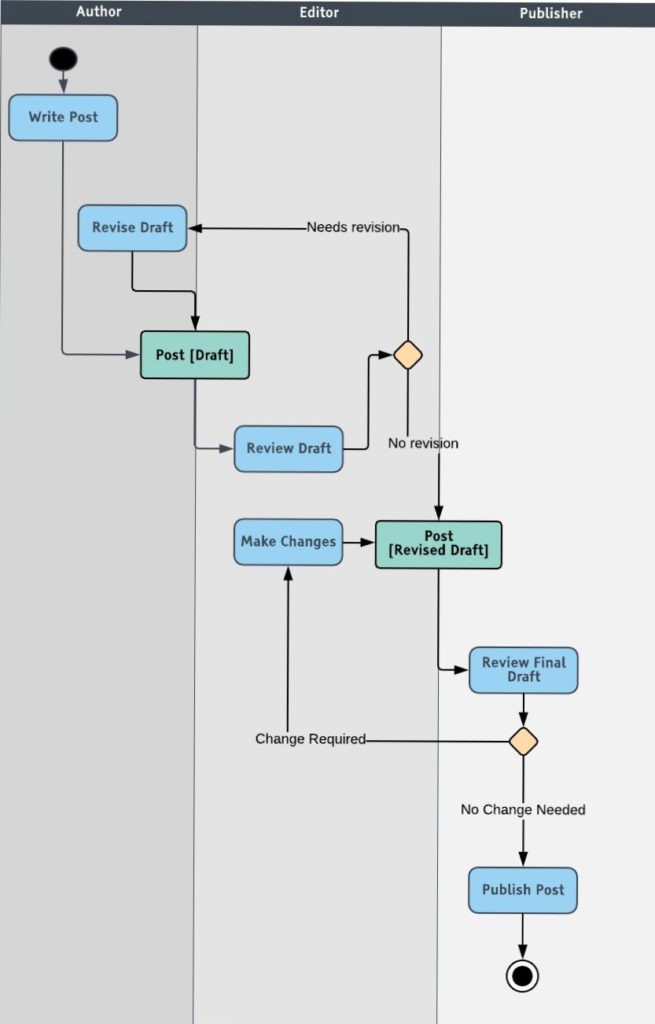
# Stukturaviy UML diagrammasi:

* Class Diagram
* Object Diagram
* Component Diagram
* Composite Structure Diagram
* Deployment Diagram
* Package Diagram
* Profile Diagram

# Xulq-atvor UML diagrammaslari.

**Activity Diagram (Faoliyat diagrammalari)**

Faoliyat diagrammalari, biznes jarayonlarini modellashtirish uchun eng muhim UML diagrammalaridir. Dasturiy ta'minotni ishlab chiqishda, odatda, turli xil harakatlar va harakatlar oqimini tavsiflash uchun foydalaniladi. Ular ketma-ket va parallel ravishda bo'lishi mumkin. Ular faoliyat tomonidan ishlatiladigan, iste'mol qilingan yoki ishlab chiqarilgan ob'ektlarni va turli xil faoliyat turlari o'rtasidagi munosabatni tavsiflaydi. Yuqorida aytilganlarning barchasi biznes jarayonlarini modellashtirishda juda muhimdir.



Jarayon ishlab chiqarilayotgan narsalarga emas, balki bir-birini yetaklaydigan va ular bir-biriga bog'langan, aniq boshlanishi va oxiri bo'lgan tadbirlar majmuiga yo'naltirilgan. Yuqoridagi misol tarkibni nashr qilish jarayonida sodir bo'ladigan tadbirlar majmuasini aks ettiradi. Ishbilarmonlik muhitida bu biznes jarayonlarini xaritalash yoki biznes jarayonlarini modellashtirish deb ham yuritiladi. Asosiy aktyorlar - muallif, muharrir va noshir. Ushbu misolda, ko'chadan biri sharhlovchi loyihani ko'rib chiqayotganda sodir bo'ladi va ba'zi bir o'zgartirishlar kiritish kerakligi to'g'risida qaror qabul qiladi. So'ngra muallif loyihani qayta ko'rib chiqadi va qayta ko'rib chiqish uchun tahlil qilish uchun uni qayta ko’rib chiqish liniyasiga tushiradi.

# Use Case Diagram (Case diagrammasidan foydalanish)

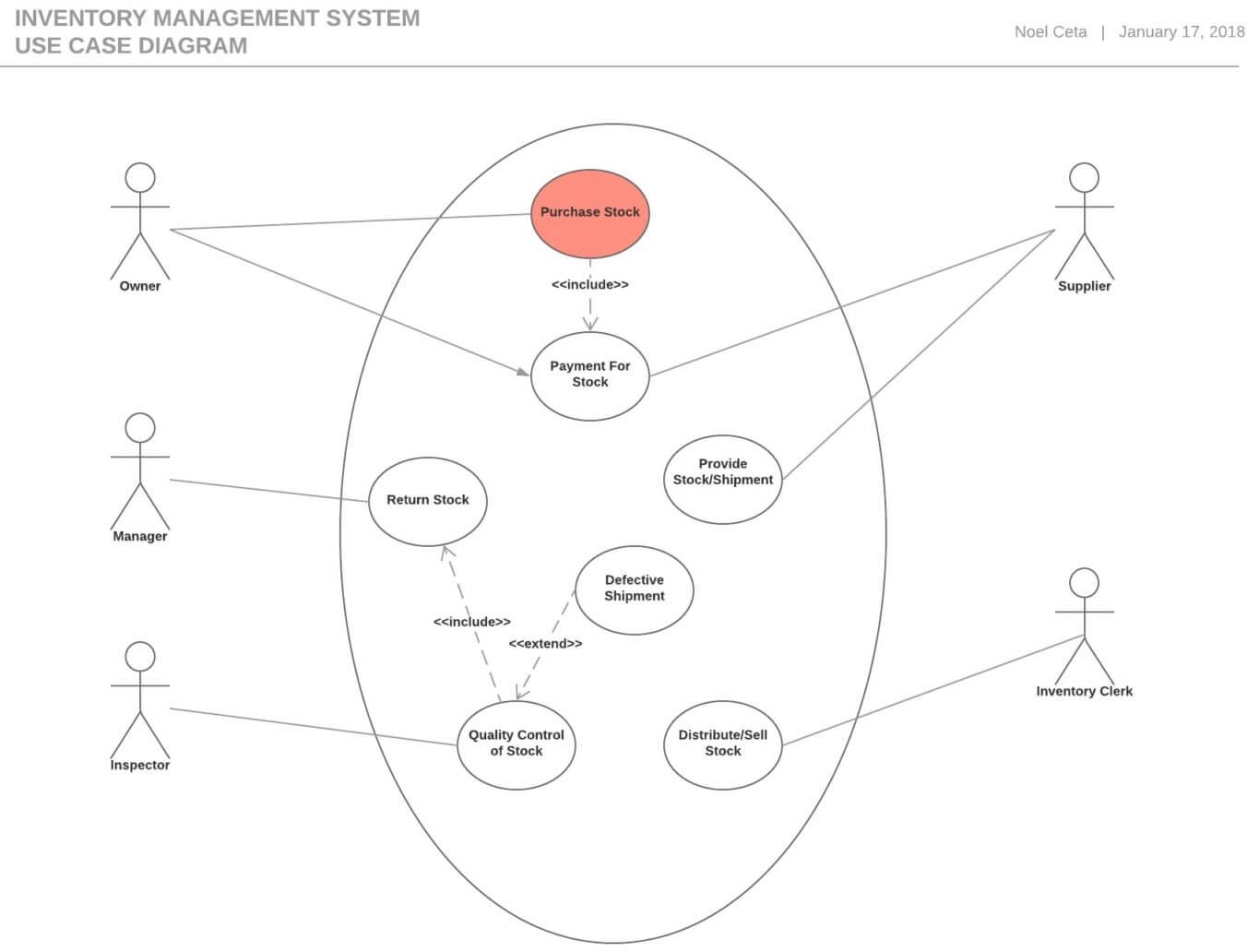
Tizimning asosiy qismi bu tizim bajaradigan funktsional talablardir. Use Case diagrammasi tizimning yuqori darajadagi talablarini tahlil qilish uchun ishlatiladi. Ushbu talablar turli xil foydalanish holatlari orqali ifoda etilgan. Ushbu UML diagrammasining uchta asosiy tarkibiy qismini ko'ramiz:

Funktsional talablar - foydalanish hollari sifatida ifodalanadi, bu harakatni tavsiflovchi fe'ldir.

Aktyorlar - ular tizim bilan o'zaro aloqada bo'lishadi; aktyor inson, tashkilot yoki ichki yoki tashqi dastur bo'lishi mumkin.

Aktyorlar va foydalanish holatlari o'rtasidagi munosabatlar - to'g'ri chiziqlar yordamida tasvirlangan.

Quyidagi misolda inventarizatsiyani boshqarish tizimi uchun UML diagrammasi tasvirlangan. Bunday holda egasi, yetkazib beruvchisi, menejeri, inventarizatsiya bo'yicha kotibi va inventarizatsiya inspektori mavjud.



Aylana konteynerlar ichida biz aktyorlar bajaradigan harakatlarni ifoda etamiz. Bunday harakatlar aktsiyalarni sotib olish va to'lash, aktsiyalar sifatini tekshirish, aktsiyalarni qaytarish yoki tarqatish kabi masalalarni o’z ichiga oladi. Tizimning ko'rinishini soddalashtirish va amalga oshirish tafsilotlarini aks ettirmaslik orqali tizim ichidagi aktyorlar o'rtasidagi dinamik harakatlarni ko'rsatish uchun foydalanilgan.

# Interaction Overview Diagram (O'zaro aloqalar diagrammasi)

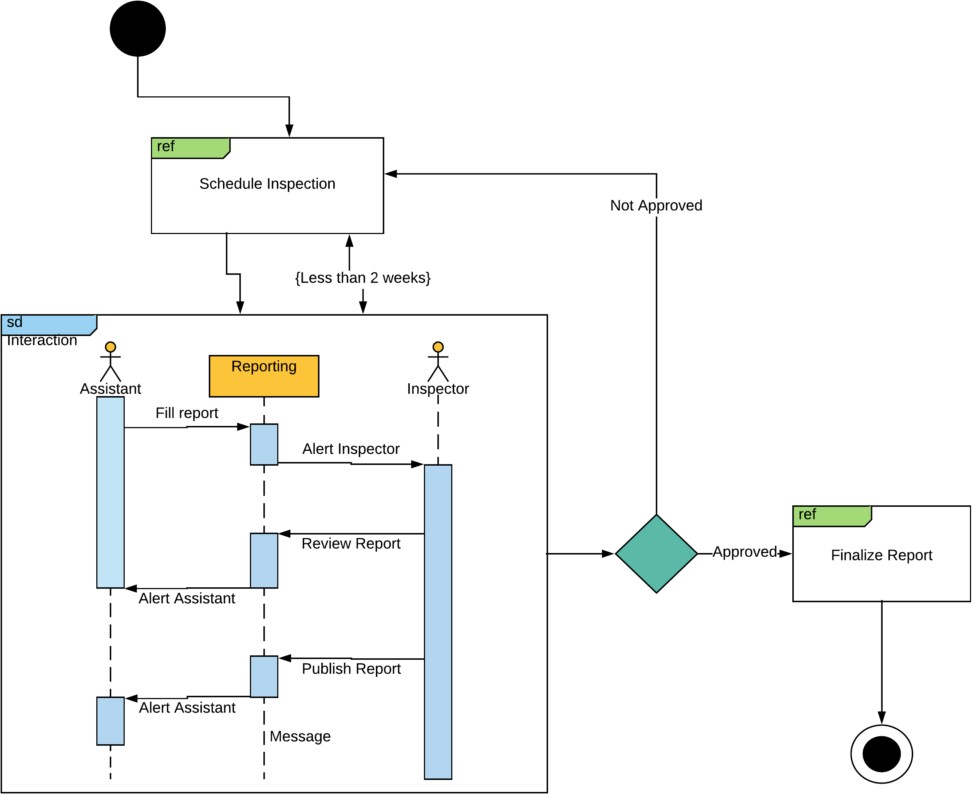
O'zaro aloqalar UML diagrammalari eng murakkab sxemalardan biridir. Hozircha biz faoliyat diagrammasi nima ekanligini ko’rib chiqdik. Bundan tashqari, xulq-atvor diagrammasi to'plamida o'zaro ta'sir diagrammasi mavjud unda to'rtta diagrammadan iborat, Bular:

O'zaro aloqalar diagrammasi Vaqt diagrammasi

Tartib diagrammasi Aloqa diagrammasi

Shunday qilib, o'zaro ta'sirning umumiy diagrammasi - bu turli xil ta'sir o'tkazish diagrammalaridan qilingan faoliyat diagrammasi. Bu o'zaro faoliyat diagrammasi bilan faoliyat diagrammalarining aralashmasi, ammo aksariyat veb- saytlar ularni ixtisoslashtirilgan faoliyat diagrammasi deb bilishilishadi. Buning ma'nosi shundan iboratki, siz o'zaro faoliyat, o'zaro ta'sirlashish, vaqtni cheklash, davomiylik va hokazo kabi elementlarni qo'shib, faoliyat diagrammasi ichida

ishlatiladigan izohlarning aksariyatidan foydalanishingiz mumkin.



Yuqoridagi misol UML diagrammalaridan tizimning dinamik harakatini, tarkibiy tuzilishini va ob'ektlar o'rtasidagi o'zaro ta'sirni tavsiflash uchun qanday ishlatilishini ko'rsatilagan. Bularning barchasi, voqealar sodir bo'lish vaqti va tartibini hisobga olgan holda, voqealar va xabarlar oqimining ketma-ketligini kuzatib boradi. Diagrammada, xuddi har qanday faoliyat diagrammasi singari, boshlang'ich va tugash nuqtasi mavjud. Keyinchalik, yuqori darajadagi ko'rinishda u to'rtburchaklar ramkalar yordamida o'zaro ta'sirlar va o'zaro ta'sirlardan foydalanishni tasvirlaydi. O'zaro ta'sirlar doirasida (to'rtburchaklar ramkalar) bunda uchta asosiy aktyorni o'z ichiga olgan to'liq mustaqil ketma-ketlik diagrammasi kiritilgan bular: yordamchi, vositachilarning hisobotlari tizimi va inspektor. Amallar ketma-ketligi tugagandan so'ng, oqim holati tarmoqlanadi va avvalgi o'zaro ta'sirni takrorlaydi yoki yangi o'zaro ta'sirga o'tadi va keyin oqimni tugatiladi

# Timing Diagram (Vaqt diagrammasi)

Vaqtni belgilash UML diagrammasi diqqat markazida o'z vaqtida turganda ob'ektlarning munosabatlarini ifodalash uchun ishlatiladi. Ob'ektlarning o'zaro ta'siri yoki bir-birini o'zgartirishi qiziqtirmaydi, aksincha ob'ektlar va aktyorlarning chiziqli vaqt o'qi bo'ylab qanday harakat qilishlarini tasvirlash bu yerda eng asosiysi hisoblanadi. Har bir alohida ishtirokchi hayot chizig'i orqali namoyish etiladi, bu asosan bosqich bir bosqichdan ikkinchisiga o'tish bosqichini tashkil

etuvchi bosqich bo'lib, asosiy e'tibor voqealarning davomiyligi va davomiylik cheklovlariga qarab sodir bo'ladigan o'zgarishlarga qaratilgan bo’ladi.

Vaqt UML diagrammasining asosiy tarkibiy qismlari:

**Lifeline** - individual ishtirokchi

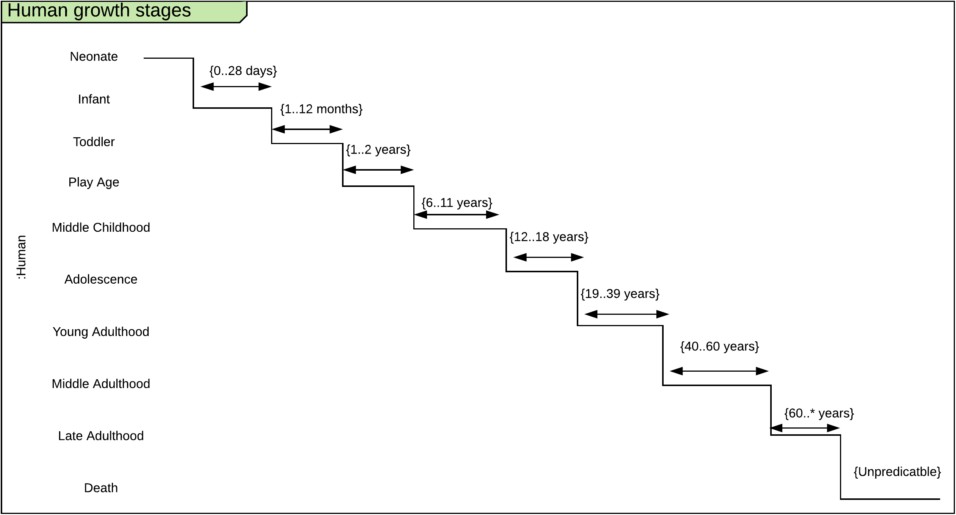
**State timeline**- bitta qutqaruv liniyasi turli xil holatlardan o'tishi mumkin

**Duration constraint**- cheklovni bajarish uchun zarur bo'lgan vaqtni ifodalovchi vaqt oralig'idagi cheklov

**Time constraint** - vaqt oralig'idagi cheklash, bu vaqt davomida ishtirokchi tomonidan biron bir narsani bajarishi kerak

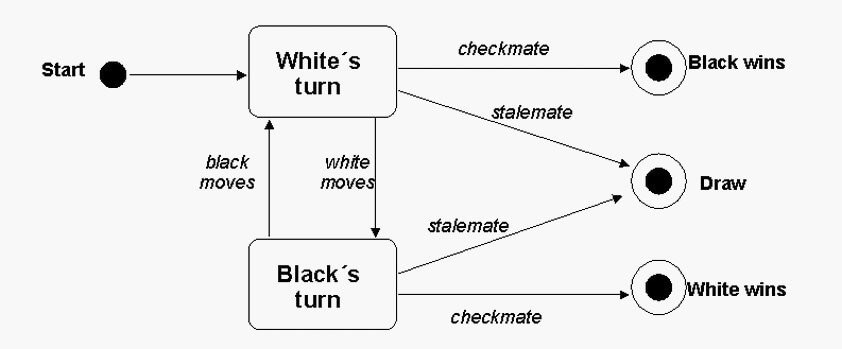
**Destruction occurrence** - individual ishtirokchini yo'q qiladigan va ushbu ishtirokchining hayot chizig'ining oxirini tasvirlaydigan xabar paydo bo'lishidir.

Vaqtning soddalashtirilgan UML diagrammasiga misol quyida keltirilgan. Bu inson o'sish bosqichlarini aks ettiradi. Natijada, u faqat bitta hayot chizig'iga ega bo’ladi.



# State Machine Diagram (Davlat mashina diagrammasi)

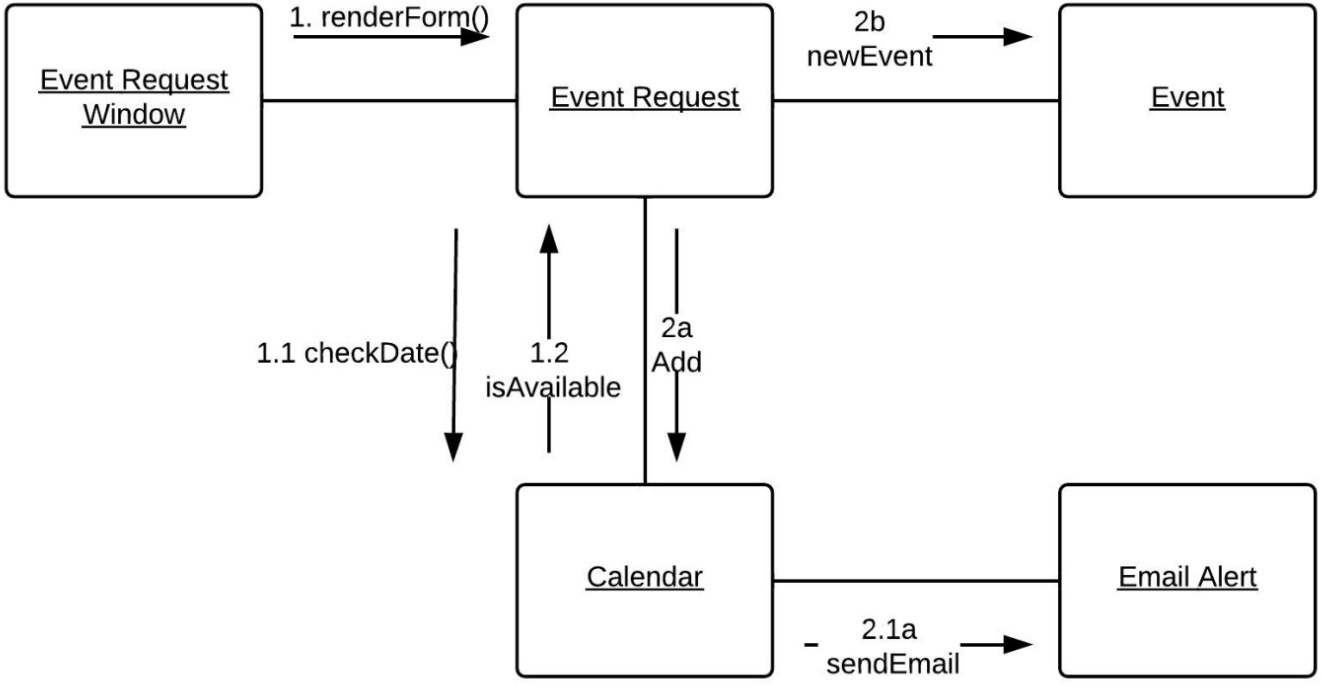
Statechart diagrammasi deb ham ataladigan davlat mashina UML diagrammalari tizim tarkibidagi komponentning har xil holatlarini tavsiflash uchun ishlatiladi. Bu holat nom mashinasini oladi, chunki diagramma asosan ob'ektning bir nechta holatini va uning ichki va tashqi hodisalar asosida qanday o'zgarishini tavsiflovchi mashinadir. Juda oddiy holatdagi mashinalar diagrammasi shaxmat o'yinining sxemasi bo'ladi. Oddiy shaxmat o'yini Oq va Qora tomonidan qilingan harakatlardan iborat. Oq birinchi harakatga ega bo'ladi va shu bilan o'yinni boshlaydi. O'yinning xulosasi Oqning navbati yoki Qora bo'lishidan qat'i nazar sodir bo'lishi mumkin. O'yin matematika, yutish, yutqazish yoki durang bilan yakunlanishi mumkin (mashinaning turli holatlari).



Statecharts asosan turli xil tizimlarning oldinga va teskari muhandisliklarida foydalanadi.

# Communication Diagram (Aloqa diagrammasi)

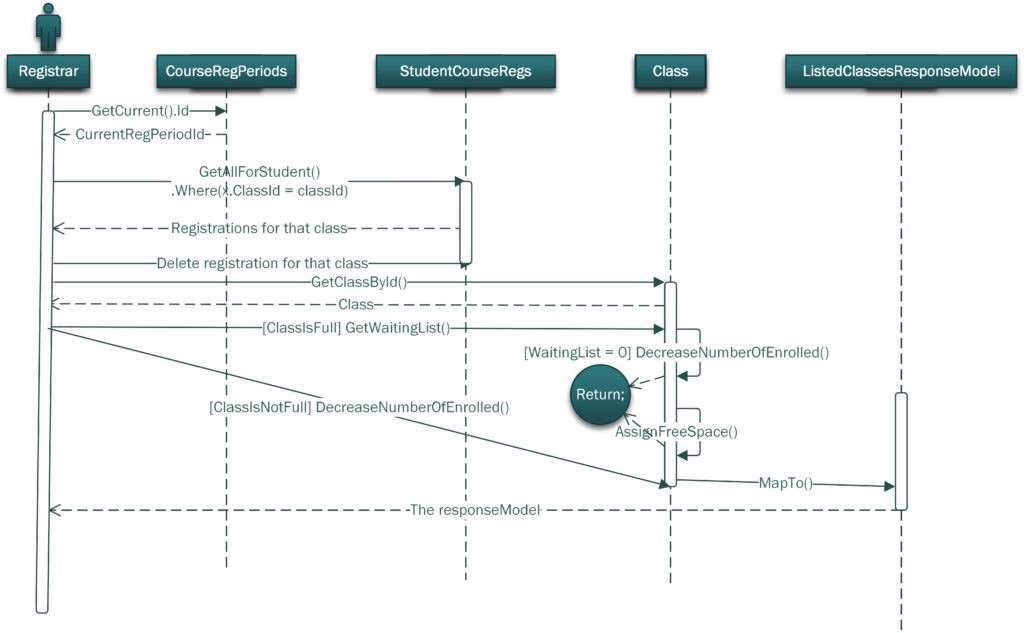
Ilgari hamkorlik diagrammasi deb nomlangan aloqa diagrammalari deyarli UML-da ketma-ketlik diagrammasi bilan bir xil, ammo ular ko'proq narsalarning o'zaro ta'siriga emas, balki qanday qilib o'zaro bog'liqlik va xabarlar orqali bog'lanishlariga e'tibor berishadi. Quyidagi misolda aloqa diagrammasi voqeani taqvimga qo'shish jarayonini tushuntiradi. Har bir satrdagi raqamlar ular yoqilgan tartib va variantlarni aks ettiradi. Biz bilamizki, ba'zi harakatlar harflardan foydalanilganligi sababli bir vaqtda sodir bo'ladi



# Sequence Diagram (Tartib diagrammalari)

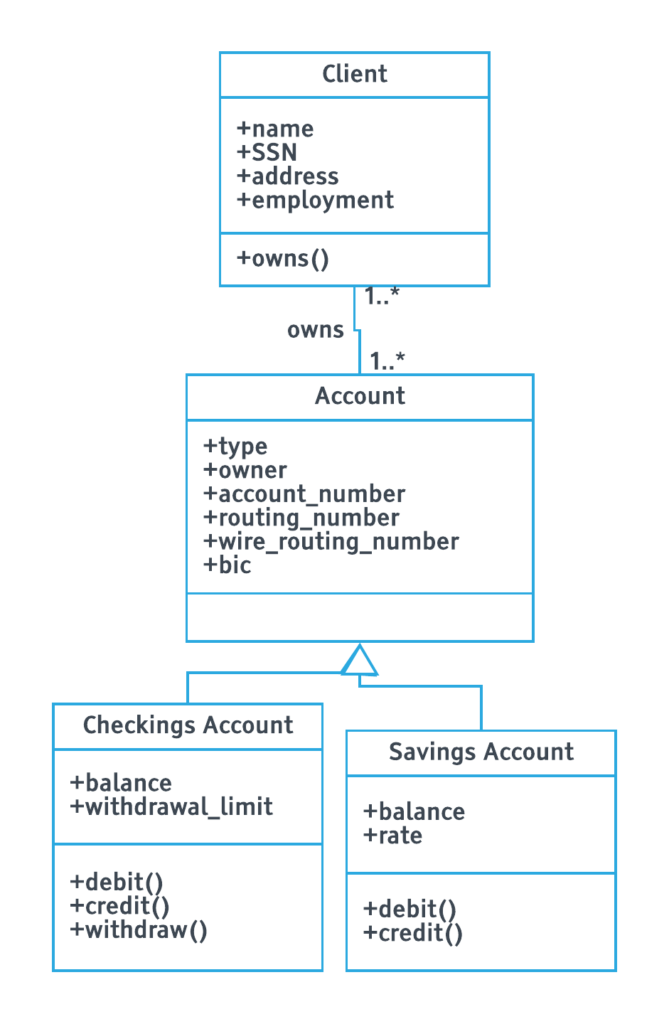
Tartib diagrammalari, nafaqat kompyuter fanlari jamoatchiligi, balki biznes dasturlarini ishlab chiqish uchun dizayn darajasidagi modellar orasida eng muhim UML diagrammalaridir. So'nggi paytlarda ular biznes jarayonlarini tasvirlashda mashhur bo'libmoqda. Bu diagrammalar aktyorlar va ob'ektlar o'rtasida sodir bo'ladigan xabarlar va o'zaro ta'sirlarning ketma-ketligini tavsiflaydi. Aktyorlar yoki ob'ektlar faqat kerak bo'lganda yoki boshqa ob'eykt ular bilan aloqa qilishni xohlaganda faol bo'lishi mumkin. Barcha aloqa xronologik tarzda namoyish etiladi.

Tizim tuzilishini tasvirlash uchun strukturaviy diagrammalardan foydalaniladi. Aniqrog'i, u dasturiy ta'minotni ishlab chiqishda tizimning arxitekturasini va turli xil tarkibiy qismlarning bir-biriga bog'liqligini (ularning o'zini tutishi yoki aloqasi emas, shunchaki turgan joyida) aks ettirish uchun ishlatiladi. Quyida kurslarni ro'yxatdan o'tkazish tizimi tasvirlangan ketma-ketlik diagrammasining namunasi ko’rsatilgan.



# Stukturaviy UML diagrammasi: Class Diagram (Sinf diagrammasi)

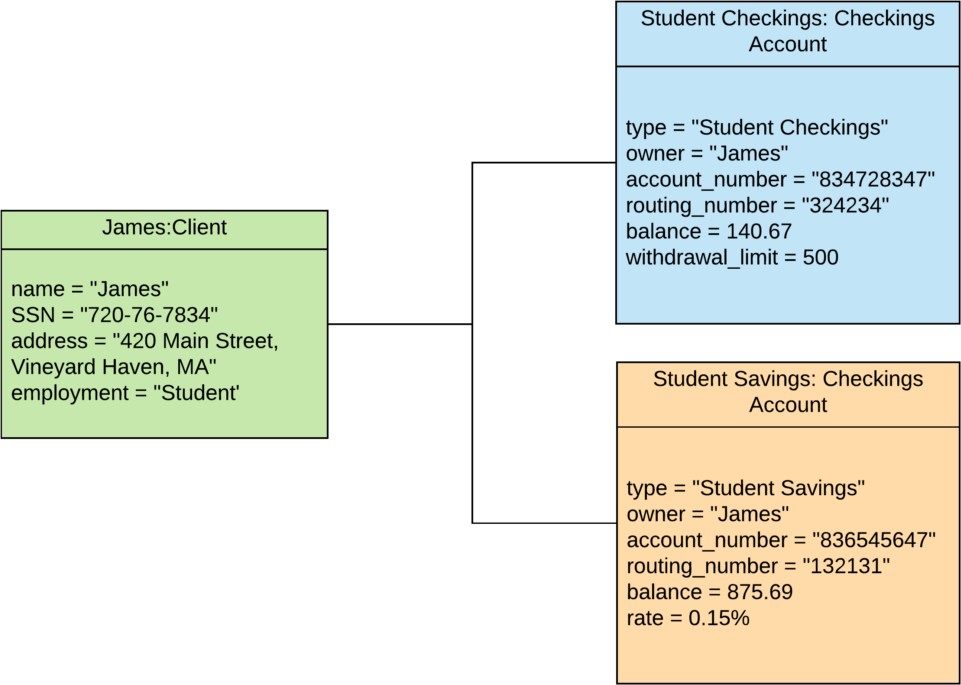
Sinf UML diagrammasi dasturiy hujjatlar uchun eng keng tarqalgan diagramma turi hisoblanadi. Hozirgi kunda yaratilgan dasturiy ta'minotlarning aksariyati hanuzgacha Ob'ektga yo'naltirilgan dasturlash paradigmasiga asoslanganligi sababli, dasturiy ta'minotni hujjatlashtirish uchun sinf diagrammalaridan foydalanish yaxshi fikrga aylandi. Bu sodir bo'ladi, chunki OOP sinflarga va ular orasidagi munosabatlarga asoslangan. Xulosa qilib aytganda, sinf diagrammalarida atributlari (ma'lumotlar maydonlari deb ham yuritiladi) va ularning xatti-harakatlari (a'zo funktsiyalari deb ham yuritiladi) bilan bir qatorda sinflar mavjud. Aniqrog'i, har bir sinfda uchta maydon mavjud: tepada sinf nomi, ismning ostidagi sinf atributlari, pastda sinf operatsiyalari. Turli sinflar o'rtasidagi munosabatlar (bog'lovchi chiziq bilan ifodalanadi), sinf diagrammasini tashkil qiladi.



Yuqoridagi misol asosiy sinf diagrammasini ko'rsatadi. "Chek hisobi" klassi va "Jamg'arma hisobi" klassi ikkalasi ham umumiy "Hisob" sinfidan olinadi. Meros bo'sh boshli strelka yordamida ko'rsatiladi. Diagrammadagi boshqa sinf - "Mijoz" klassi mavjud. Diagramma o'zi- o'zidan tushunarli va u turli sinflarni va ularning o'zaro bog'liqligini aniq ko'rsatib beradi.

# Object Diagram (Obyekt diagrammasi)

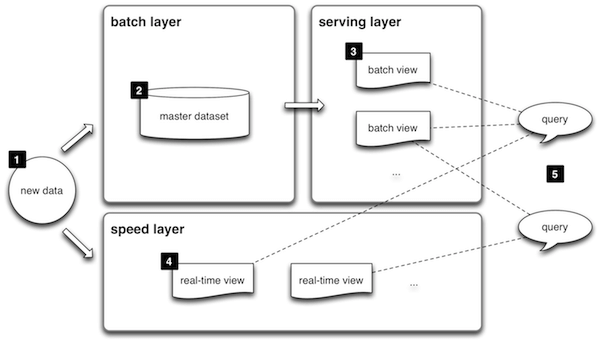
Dasturiy ta'minotni ishlab chiqishda sinflar ma'lumotlarning mavhum turlari hisoblanadi, ob'ektlar esa mavhum sinf namunalari (klassdan obyekt olish). Masalan, agar bizda umumiy mavhum tip bo'lgan "Avtomobil" klassi bo'lsa, unda "Avtomobil" sinfidan "Audi" meros olgan bo'ladi. Ob'ekt UML diagrammasi dasturiy ta'minot ishlab chiquvchilariga ular yaratgan umumiy mavhum tuzilmani (sinf diagrammasi) amalda qo'llashda hayotiy tuzilishni anglatishini tekshirishga yordam beradi, ya'ni sinf ob'ektlari asoslanganda bo’ladi.



Yuqoridagi ob'ekt UML diagrammasi biz ilgari ko'rsatgan sinf diagrammasiga asoslanadi. Unda biz ilgari yaratgan sinflar misollari (ob'ektlari) tasvirlangan. Aniqroq aytganda, "Mijoz" umumiy klassi, endi "Jeyms" deb nomlangan haqiqiy mijozga ega. Jeyms ko'proq umumiy sinfning namunasidir va u bir xil atributlarga ega, ammo berilgan qiymatlar bilan. Xuddi shu narsa Chek va Jamg'arma hisob varog'ida ham amalga oshirildi. Ularning ikkalasi ham o'z sinflarining ob'ektlari. "Account\_number" va "routing\_number" atributlari Chek va Jamg’arma hisobi uchun har xil ekanligini sezishingiz mumkin. Natijada, ushbu xususiyatlarni "Hisob" umumiy sinfiga emas, balki o'zlarining sinflariga qo'yish mantiqan to'g'ri keladi.

# Component Diagram (Komponent diagrammasi)

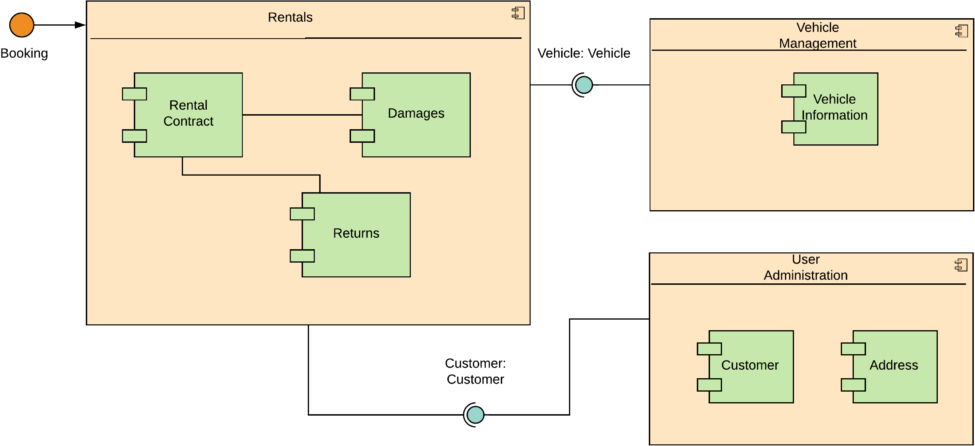
Murakkab tizimlarning hujjatlari bilan ishlashda UML komponent diagrammalari tizimni kichik tarkibiy qismlarga ajratishga yordam beradi. Ba'zan tizimning arxitekturasini tasvirlash qiyin, chunki u bir nechta bo'limlarni qamrab olishi yoki turli xil texnologiyalardan foydalanishi mumkin. Masalan, Lambda arxitekturasi UML komponent komponentasi yordamida namoyish etilishi mumkin bo'lgan murakkab me'morchilikning odatiy namunasidir. Lambda arxitekturasi - bu tarqatilgan tizimda ma'lumotlarni saqlash va qayta ishlash uchun bir nechta kompaniyalar tomonidan ishlaydigan ma'lumotlarni qayta ishlash arxitekturasi. U uch xil qatlamdan iborat: tezlik qatlami, ishchi qatlam va xizmat qiluvchi qatlam.



Yuqoridagi rasmda komponent diagrammasi qanday qilib murakkab tizimning soddalashtirilgan yuqori darajadagi ko'rinishini olishimizga yordam berishi mumkinligi ko'rsatilgan. Bu erda ishlatiladigan izohlar UML standartlariga mos kelmagan, ammo ular juda o'xshash.

# Composite Structure Diagram (Kompozit stuktura diagrammasi)

Ushbu turdagi UML diagrammasi odatda qo'llanilmaydi, chunki uning vazifasi juda aniq. U faqat sinfning ichki tuzilishini va turli sinf tarkibiy qismlari o'rtasidagi munosabatlarni aks ettiradi. Kompozit stukturalar diagrammasi biznes mutaxassislarini umuman qiziqtirmaydi, chunki ularning asosiy yo'nalishi tarkibiy qismlarning yuqori darajadagi ko'rinishiga va ular bir-biri bilan qanday aloqa qilishiga qaratilgan. Menejer uchun sinfning ma'lum bir ma'lumot a'zosi boshqa sinf ma'lumotlari bilan qanday bog'liqligini bilishi deyarli ahamiyatsiz. Quyida uning tashqi ko'rinishi haqida umumiy tasavvurga ega bo'lish uchun soddalashtirilgan misolni ko’rish mumkin.



# Deployment Diagram (Joylashtirish diagrammasi)

Dasturiy ta'minot va apparat o'rtasidagi munosabatni tasavvur qilish uchun tarqatish diagrammalaridan foydalaniladi. Aniqroq aytadigan bo'lsak, tarqatish sxemalari bilan biz dasturiy ta'minot tarkibiy qismlari (artefaktlar) tugunlar deb

nomlanuvchi apparat qismlariga qanday joylashtirilganligining fizik modelini tuzish mumkin. Web-dastur uchun odatiy soddalashtirilgan tarqatish diagrammasi quyidagilarni o'z ichiga oladi:

Tugunlar (dastur serveri va ma'lumotlar bazasi serveri) Artefaktlar (dastur mijozi va ma'lumotlar bazasi sxemasi

Tugunlarda artefaktlar joylashgan. Ma'lumotlar bazasi sxemasi ma'lumotlar bazasi serverida va dastur mijozi dastur serverida ishlaydi. Nomidan ko'rinib turibdiki, tarqatish diagrammasi har bir dasturiy ta'minot komponentining qaerda joylashganligini aniq ko'rsatib beradi.

# Package Diagram(Paket diagrammasi)

Paket diagrammasi biz yuqorida bayon qilgan UML diagrammalarini joylashtirish uchun so'l konteynerga o'xshaydi. Turli xil paketlarda tugunlar va artefaktlar mavjud. Ular model diagrammalarini va tarkibiy qismlarini guruhlarga ajratadilar, xuddi shu tarzda nomlar maydoni bir-biri bilan o'zaro bog'liq bo'lgan turli xil nomlarni o'z ichiga oladi. Oxir oqibat, yanada murakkab tizimlar va xatti- harakatlarni tasvirlash uchun paketni bir nechta boshqa paketlar tomonidan qurish mumkin. Paket diagrammasining asosiy maqsadi murakkab tizimni tashkil etuvchi har xil yirik komponentlar o'rtasidagi munosabatlarni ko'rsatishdir. Dasturchilar ushbu abstraktsiya imkoniyatini paketli diagrammalardan foydalanishning yaxshi afzalligi deb bilishadi.

# Profile Diagram (Profildiagrammasi)

Profil diagrammasi odatdagi UML diagrammasi turi emas. Aslida, bu boshqalarga o'xshash diagramma turi emas, balki ko'proq kengayish mexanizmi sifatida qaralishi mumkin. Stereotiplar, belgilangan qadriyatlar va cheklovlardan foydalangan holda siz allaqachon mavjud UML yozuvlarini kengaytirishingiz va sozlashingiz mumkin. Profil diagrammalari tilga o'xshaydi, agar siz ingliz tilida gaplashsangiz, yangi jumlalar yaratishingiz mumkin, agar siz profil diagrammalarida gaplashsangiz, u holda siz UML diagrammalari uchun yangi xususiyatlar va semantikani yaratishingiz mumkin.

Stereotiplar - mavjud UML elementlarini kengaytirish uchun ishlatiladi. Ular sizga yangi element yoki qurilish blokini yaratishga, tahrirlashga yoki yaratishga imkon beradi, bu esa keyinchalik to'g'ridan-to'g'ri diagrammada ishlatilishi mumkin.

Belgilangan qadriyatlar - bu allaqachon mavjud modellarga yangi atributlarni qo'shish deb o'ylang. Yangi teglangan qiymat navbati bilan yangi kalit so'zga olib keladi.

Cheklovlar - bu so'z o'z-o'zidan tushunarli, ammo cheklovlarni o'zingizning diagrammalaringizga qo'shishingiz mumkin bo'lgan yangi shartlar deb o'ylang. Masalan, cheklov bo'lishi mumkin: "qoldiq qoldiq $ 3 dan katta bo'lishi kerak". Ushbu cheklovdan bank tizimi tomonidan hisob-kitob hisobvarag'i qachon tugatilishini nazorat qilish uchun foydalanish mumkin. So'nggi paytlarda UML diagrammasi juda kuchli vositaga aylandi. Dastlabki bosqichlarda faqat dasturiy ta'minot ishlab chiqaruvchilari va IT sohasidagi mutaxassislar UML dan modellar, tizimlar va dasturiy ta'minot arxitekturasini hujjatlashtirishda foydalanganlar.

Hozirgi kunda UML diagrammalari turli sohalarda qo'llanilmoqda va ko'plab ishbilarmonlar ularni kundalik ishlarida qo'llashni boshladilar.

#### Ishni bajarish namuna

**Topshiriq:** Ko’p qavatli binolarda lift mavjud bo’lib, ushbu liftning ishlash tizimini loyihalashtirish va UML diagrammalarini shakllantirish lozim.

Dastlab lift tizimining qisqacha tasnifini keltiramiz. Umumiy ishlash prinsipiga ko’ra lift foydalanuvchi tomonidan chaqrilida, yo’nalish ko’rsatiladi va ko’rsatilgan yo’nalish bo’yicha belgilangan qavatga ko’tariladi yoki aksincha.

Liftda belgilangan qavatga yetib olish algoritmi quyidagicha:

1. liftni chaqirish (lift turgan joyiga nisbatan pastdan yoki yuqoridan);
2. lift eshigi ochiladi;
3. kerak bo’lgan qavat belgilanadi;
4. left eshigi yopiladi;
5. belgilangan qavatga harakatlanadi;
6. lift ehigi ochiladi;
7. lift eshigi yopiladi;
8. lift neytral holatga o’tadi.

\*Izoh: Agar zarur bo’lgan hollarda lift tizimi ishdan chiqib qolsa favqulotda yordam tugmasi (emergency call) orqali navbatchini (dispatcher) chaqirish mumkin.

Demak yuqoridagilardan kelib chiqqan holda mazkur tizimning talablarini ishlab chiqishimiz mumkin:

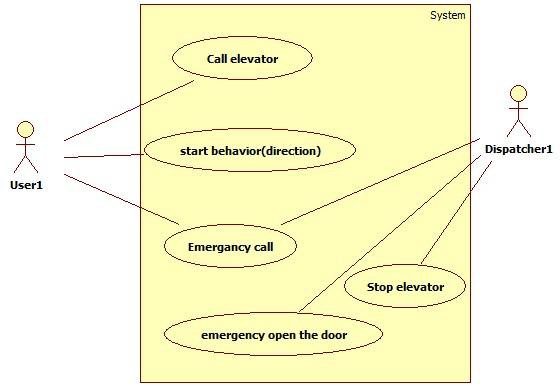
Lift tizimining talablari

* T-01: foydalanuvchi tomonidan chaqiruv amalga oshirilganda kerakli qavatga harakatlanishi kerak;
* T-02: lift belgilangan qavatga yetib kelganida to’xtashi va eshil ochilishi kerak;
* T-03: biror bir qavat ko’rsatilmaguncha ma’lum vaqt neytral holatda turishi kerak;
* T-04: harakatlanish uchun qavat ko’rsatilgandan keyin eshik yopilishi kerak;
* T-05: harakat ko’rsatilgan qavatgacha amlaga oshishi kerak;
* T-06: ko’rsatilgan qavatga yetib borganidan keyin lift to’xtashi bilan eshik avtomatik ochilishi kerak;
* T-07: ma’lumo vaqtdan keyin eshik avtomatik yopilishi kerak

Lift tizimining funksional talablari

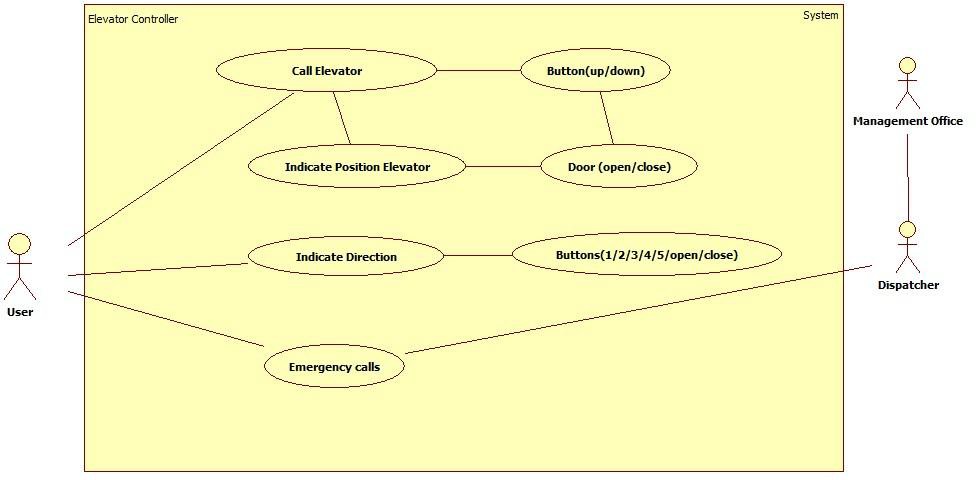
* FT-01: foydalanuvchi (user) – liftdan foydalanich jarayonida ketma-ketlikni to’g’ri bajarishi va harakatni amalga oshirishi lozim;
* FT-02: navbatchi (dispatcher) – liftda favqulotda vaziyat bo’lganida yoki ishdan chiqqanida birinchi yordam ko’rsatishi lozim;
* FT-03: lift qurilmasiga javobgar tashkilot boshqaruvchisi navbatchi faoliyatini muvoffiqlashtirib tuishi lozim;
* FT-04: lift tizimi xatosiz ishlashi va chidamli bo’lishi lozim.

Ushbu talablarni tizim xususiyatidan kelib chiqqan holda hohlagancha o’zgartirish va davom ettirish mumkin, Ushbu talablar ishlab chiqiladigan tizimning asosiy qismini tashkil etadi. Endi quyidagi rasmda lift tizimining umumiy sxemasini keltiramiz:



1-rasm. Lift tizimining umumiy sxemasi

Tizimning use case diagrammasi quyidagicha bo’ladi:

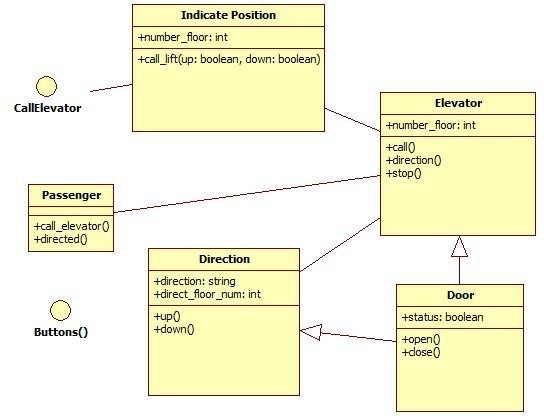


2-rasm. Lift tizimining use case diagrammasi

Tizimning use case diagrammasini ishlab chiqishda asosan quyidagi elementlardan foydalaniladi:

* + Use case (tizimning tashkil etuvchi modullari)
  + Actor (tizim foydalanuvchisi, administrator, dispatcher)
  + Dependency, generalization, association (tizim modullarini bog’lash qonuniyatlari)

Tizimning class diagrammasi quyidagicha bo’ladi:

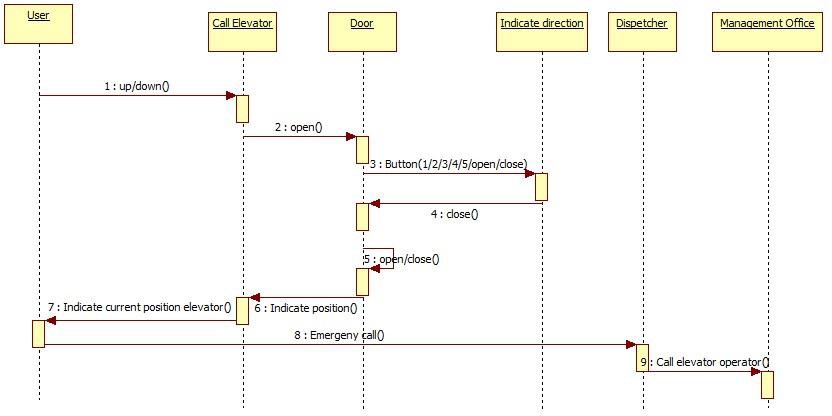


3-rasm. Lift tizimining class diagrammasi

Tizimning class diagrammasini shkllantishda esa quyidagi asosiy elementlardan foydalaniladi:

* + Classes (sinflar)
  + Interfaces (interfeyslar)
  + Collaborations (hamkorliklar)
  + Dependency, generalization, association (sinflar va interfeyslarni bir – biriga bog’lash qonuniyatlari).

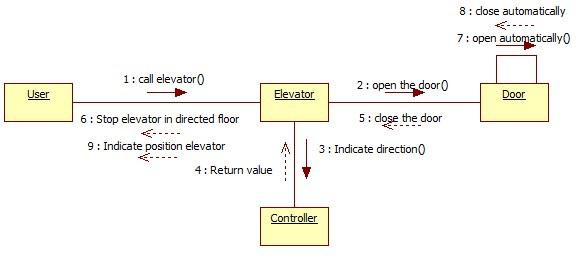
Tizimning sequence diagrammasi



4-rasm. Lift tizimining sequence diagrammasi

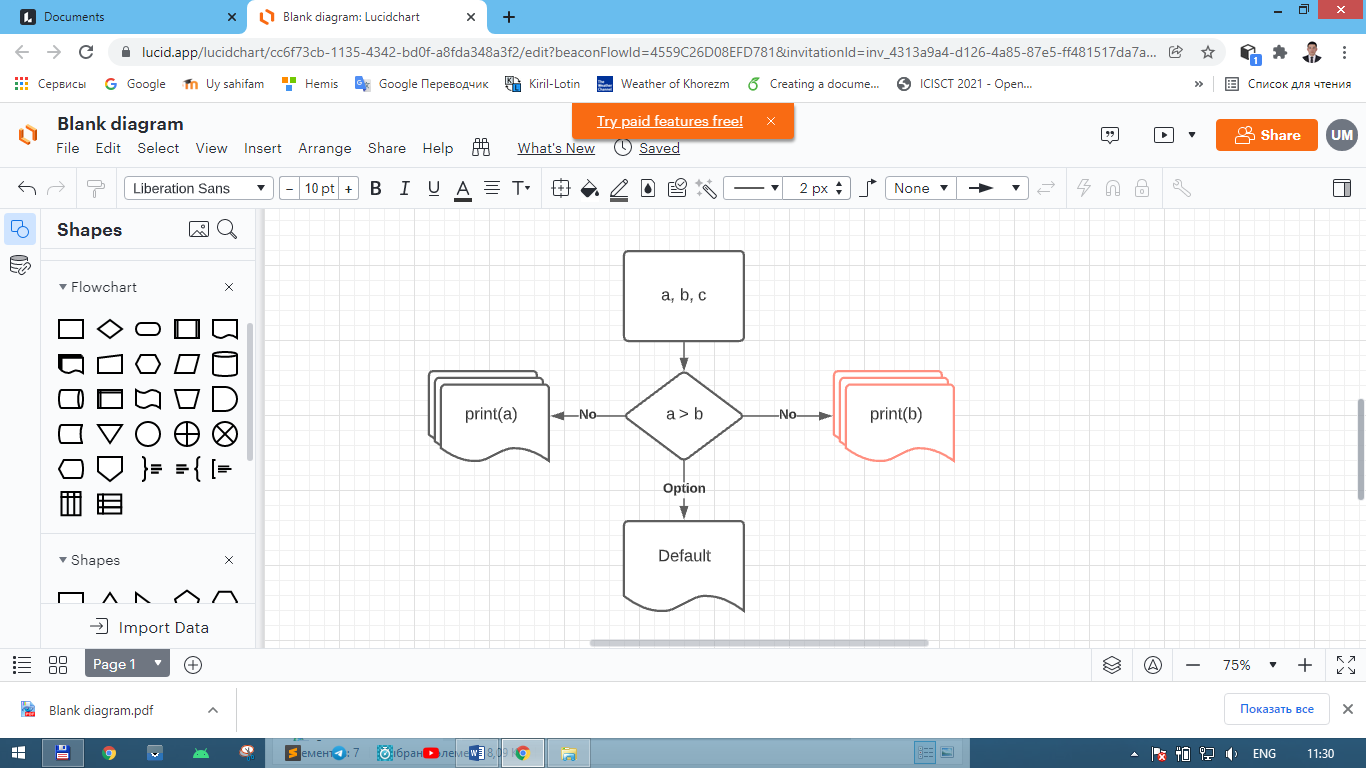
O’z navbatida quyidagi elementlar tizimning sequence diagrammasini shakllantirishda ishlatiladi:

* + Objects (tizim obyetlari)
  + Links (obyekt bog’lash va yo’naltirish belgisi)
  + Messages (bog’lanish holatini bildiruvchi xabarlar)
  + Respond Time (qayta aloqa vaqtini ko’rsatish) Tizimning collobaration (hamkorlik) diagrammasi



5-rasm. Lift tizimining collobaration diagrammasi

[https://lucid.app/lucidchart/cc6f73cb-1135-4342-bd0f-a8fda348a3f2/edit?beaconFlowId=4559C26D08EFD781&invitationId=inv\_4313a9a4-d126-4a85-87e5-ff481517da7a&page=0\_0#](https://lucid.app/lucidchart/cc6f73cb-1135-4342-bd0f-a8fda348a3f2/edit?beaconFlowId=4559C26D08EFD781&invitationId=inv_4313a9a4-d126-4a85-87e5-ff481517da7a&page=0_0)



**Topshiriq**

4 ta misolning diagrammasini UML tilida ifodalab berish.