



DASTURIY TA'MINOTNI ISHLAB CHIQISH JARAYONI

Mualliflar:

Abdisoatov Jasurbek Rustam o'g'li

TATU Dasturiy injiniring fakulteti 3-kurs talabasi: jasurabdisoatov03@gmail.com,

Ibroximov Ibratjon Iqboljon o'g'li

TATU Dasturiy injiniring fakulteti 3-kurs talabasi: ibratjonibroximov@gmail.com

Dasturiy ta'minotni ishlab chiqishning dastlabki kunlarida (70-yillargacha) dasturiy ta'minot "muhandislik qilish" kerak bo'lgan narsa ekanligi haqidagi g'oya keng tushunilmagan edi. Dasturiy ta'minotni ishlab chiqish oddiy, kvazi-ma'muriy jarayon sifatida qaraldi. Keng tarqalgan fikr shundan iborat ediki, har qanday oqilona qobiliyatli muhandis (so'zning an'anaviy ma'nosida) dasturlashga qo'lini burishi mumkin edi. Oxir oqibat, bu shunchaki "oqim sxemasini yaratish va uni kompyuter kodiga aylantirish" dan iborat edi (ibora 1962 yilda Bell Labs[1] tomonidan "Dasturlash" haqidagi ushbu ajoyib filmdan olingan).

Shubhasiz, bunday munosabat dasturiy ta'minotni ishlab chiqish qanchalik murakkab bo'lishi mumkinligini tushunolmadi. Shunga qaramay, bir qator muhim yutuqlarga erishildi. Dasturiy ta'minot o'z davri uchun juda murakkab edi (u taxminan 450 000 qator kodni o'z ichiga olgan va boshqa narsalar qatorida kosmik kemanding turli qismlariga quvvat taqsimotini boshqarish uchun mas'ul edi). Manba kodi endi onlaynda mavjud[2] va o'qishga arziydi, ayniqsa transkripsiya qilingan kod sharhlari uchun. Bular ishlab chiquvchilar ba'zida kodning ba'zi tafsilotlarini to'liq tushunish uchun qanday kurashganligi haqida yaxshi taassurot qoldiradi. Bunga yaxshi misol - LUNAR LANDING GUIDANCE EQUATIONS.agc moduli,

Bu erda o'zgaruvchilar uchun sharhlar "VAQTINCHI, I HOPE HOPE HOPE" va "Numero Misterioso" ni o'z ichiga oladi. Shunga qaramay, missiya muvaffaqiyatli bo'ldi va dasturiy ta'minot o'z ishini qildi. Bu dasturiy ta'minot injiniringi fan sifatida paydo bo'lishidan oldin va dasturiy ta'minotning murakkabligi keng miqyosda baholanishidan oldin sodir bo'lganligini hisobga olsak, bu hayratlanarli muvaffaqiyat edi.

Dasturiy ta'minotni ishlab chiqishga bo'lgan bu biroz e'tiborsiz munosabat asta-sekin o'zgarib bordi, chunki muhandislik muammosi dasturiy ta'minot bilan bog'liq sezilarli muammolarning ko'payishi bilan yuzaga keldi. Dasturiy ta'minot tizimlarining murakkabligi va ular ko'pincha uzluksiz ishlab chiqish va texnik xizmat ko'rsatishni talab qilishlari bilan muqarrar ravishda muammolarga olib kela boshladi. Bu amalga oshirish, ehtimol, 1968 yilda NATOning dasturiy ta'minot injiniringi bo'yicha konferentsiyasida eng yaxshi ifodalangan bo'lib, u erda "dasturiy ta'minot inqirozi" atamasi paydo bo'lgan. Ushbu kontseptsiya Edsger Deykstraning 1972 yilda Tyuring mukofotiga bag'ishlangan ma'ruzasida jamlangan [43]:



"Dasturiy ta'minot inqirozining asosiy sababi shundaki, mashinalar bir necha marta kuchliroq bo'ldi! To'g'risini aytganda: mashinalar yo'q ekan, dasturlash umuman muammo emas edi; Bizda bir nechta zaif kompyuterlar bo'lganida, dasturlash engil muammoga aylandi va endi bizda ulkan kompyuterlar bor, dasturlash ham xuddi shunday ulkan muammoga aylandi. "

Keyinchalik muammo Fred Bruks tomonidan ishlab chiqilgan dasturiy ta'minotni sanoat nuqtai nazaridan ishlab chiqish muammolari (hozirda dasturiy ta'minot muhandisligi adabiyotida muhim ahamiyatga ega) bo'yicha insholar to'plamida ishlab chiqilgan [25]. U OS/360 operatsion tizimini ishlab chiqishda IBM bilan ishlash tajribasiga tayangan.

Inqirozning keng tarqalgan tan olinishi dasturiy ta'minotni ishlab chiqishni tizimlashtirish bo'yicha bir qancha sa'y-harakatlarni keltirib chiqardi. Dasturiy ta'minotni ishlab chiqishni yanada "qattiq" qilish uchun yaxshi muhandislik amaliyotlarini ishlab chiqish kerakligi umumiy e'tirof etildi. Bu ishlab chiqish jarayonida sifat kafolatlarini yaratishga intiladigan ko'plab metodologiyalarga olib keldi.

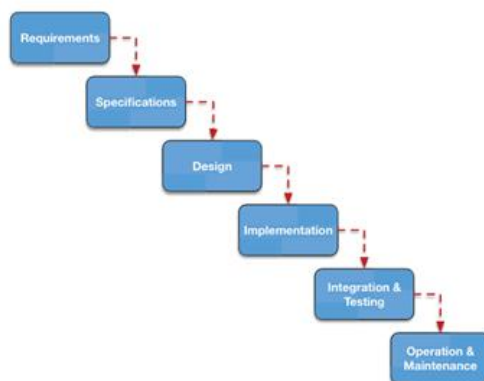
Sharshara modeli

Kanonik dasturiy ta'minotni ishlab chiqish jarayoni odatda sharshara modeli deb ataladi [114]. Bu dasturiy ta'minotni ishlab chiqish jarayonini butunlay ketma-ket, bosqichli jarayon sifatida ko'zda tutadi. Rivojlanish talablarni aniqlashdan boshlanadi, mahsulot ishlab chiqilgunga qadar va sinovdan o'tkazilgunga qadar dizaynga o'tadi va hokazo. Jarayon 1-rasmda tasvirlangan.

Sharshara modeli dasturiy ta'minotni ishlab chiqish jarayonining rasmiy tavsifini taqdim etish uchun birinchi mashhur harakatni ifodalaydi. Ehtimol, yaxshiroq alternativa yo'qligi sababli u de-fakto rivojlanish standartiga aylandi. U 80-yillar davomida AQSh Mudofaa vazirligi tomonidan ilgari surilgan va standart DOD-STD2167A [2] da mustahkamlangan, unda quyidagilar aytilgan:

... pudratchi quyidagi olti bosqichni o'z ichiga olgan dasturiy ta'minotni ishlab chiqish tsiklini amalga oshirishi kerak: Dastlabki loyihalash, batafsil loyihalash, kodlash va birliklarni sinovdan o'tkazish, integratsiya va sinovdan o'tkazish.

Shubhasiz, Sharshara modeli xato uchun kam joy qoldiradi. Talablar birinchi marta to'g'ri bo'lishi kerak, chunki rivojlanish boshlanganidan keyin ortga qaytish yo'q (hech bo'lmaganda modelga ko'ra). Natijada, palapartishlik modelini amalga oshirish ko'pincha hujjatlarga va hujjatlarning to'g'riligiga ishonch hosil qilishda ishtirok etadigan turli jarayonlarga katta e'tiborni o'z ichiga oladi.



1-rasm

Shuni ta'kidlash kerakki, 1-rasmda tasvirlangan sharshara yondashuvi standart talqinni ifodalasa-da, u dastlab Roys tomonidan nazarda tutilgan yondashuvni ifodalamaydi [89]. Roys tomonidan taklif qilingan jarayon aslida ikkita iteratsiya taklifini o'z ichiga oladi, bunda birinchi iteratsiya prototipni ishlab chiqishga qaratilgan. Biroq, bu oxir-oqibat ustunroq, qat'iyroq, ketma-ket talqin qilishda bekor qilindi.

Sharshara jarayonini qamrab olgan turli muammolar 80-90-yillar davomida yaqqol namoyon bo'ldi. Barcha talablarni oldindan belgilash zarurati va bu talablar to'liq aniq bo'lishi uchun ko'pincha haqiqiy emas edi. Bu sharshara modelining qabul qilinishini juda xavfli qildi; agar talablarni aniqlash bosqichi muvaffaqiyatsiz bo'lsa, uni tuzatish uchun katta mablag' talab qilinishi mumkin. Manfaatdor tomonlar[13] faqat boshida jalb qilingan, keyin esa loyiha ishlab chiqilayotganda uni "boshqarish" uchun rasmiy imkoniyatga ega bo'lmagan.

FOYDALANILGAN ADABIYOTLAR:

1. A brief history. Computer
2. The oracle problem in software testing: A survey. Software Engineering
3. Software Quality Assurance
4. Managing the development of large software systems
5. Software Testing Verification and Reliability
6. Software quality measurement. Advances in engineering software