VERSIYALARNI BOSHQARISH STRATEGIYALARI (GIT FLOW, TRUNK-BASED DEVELOPMENT)

Fazliddinov Xolmurod Dilmurod o‘g‘li
FarDU Axborot tizimlari va texnologiyalari yo‘nalishi 3-kurs talabasi
xolmurodfazliddinov@gmail.com
Abdukadirov Baxtiyor Abduvaxitovich
FarDU Axborot texnologiyalari kafedrasi dotsenti
bakxtiyor.uz@gmail.com

Anotatsiya: Ushbu maqolada zamonaviy dasturiy ta’minot ishlab chiqish jarayonida versiya boshqaruv tizimi (VCS) va Gitning ahamiyati ko‘rib chiqilgan. Git paydo bo‘lishidan oldin mavjud bo‘lgan muammolar tahlil qilinib, bugungi kunda eng ko‘p qo‘llaniladigan ikki strategiya – “Git Flow” va “Trunk-Based Development (TBD)” batafsil solishtirilgan. Har bir strategiyaning ijobiy va salbiy tomonlari, qaysi holatlarda eng samarali ishlashi va qachon muammolarga olib kelishi aniq misollar bilan yoritilgan. Natijada, loyihaning bosqichi, jamoa tajribasi va rivojlanish tezligiga qarab to‘g‘ri strategiya tanlashning muhimligi ta’kidlangan.

Kalit so‘zlar: Git, versiya boshqaruv tizimi, Git Flow, Trunk-Based Development, TBD, branch, pull request, dasturiy ta’minot ishlab chiqish, jamoaviy hamkorlik, CI/CD

VERSION CONTROL STRATEGIES (GIT FLOW, TRUNK-BASED DEVELOPMENT)

Fazliddinov Xolmurod Dilmurod o‘g‘li
3rd year student, Information Systems and Technologies Fergana State University (FarDU)
xolmurodfazliddinov@gmail.com
Abdukadirov Bakhtiyor Abduvakhitovich
Associate Professor of the Department of Information Technologies of FarSU
bakxtiyor.uz@gmail.com

This article examines the importance of Version Control Systems (VCS) and Git in modern software development processes. The problems that existed before the emergence of Git are analyzed, and the two most widely used strategies today – “Git Flow” and “Trunk-Based Development (TBD)” – are compared in detail. The advantages and disadvantages of each strategy, the situations in which they work most effectively, and when they can cause problems are explained with clear examples. As a result, the importance of choosing the right strategy according to the project stage, team experience, and development speed is emphasized.

Keywords: Git, Version Control System, Git Flow, Trunk-Based Development, TBD, branch, pull request, software development, team collaboration, CI/CD

СТРАТЕГИИ УПРАВЛЕНИЯ ВЕРСИЯМИ (GIT FLOW, TRUNK-BASED DEVELOPMENT)

Фазлиддинов Холмурод Дилмурод угли
Студент 3 курса направления «Информационные системы и технологии»
Ферганский государственный университет (FarDU)
xolmurodfazliddinov@gmail.com
Абдукадирова Бахтияра Абдувахитовича
доцента кафедры информационных технологий ФерГУ
bakxtiyor.uz@gmail.com

Аннотация: В данной статье рассматривается важность систем контроля версий (VCS) и Git в современных процессах разработки программного обеспечения. Проанализированы проблемы, существовавшие до появления Git, а также подробно сравниваются две наиболее часто используемые сегодня стратегии — «Git Flow» и «Trunk-Based Development (TBD)». Преимущества и недостатки каждой стратегии, ситуации, в которых они работают наиболее эффективно, и когда они могут привести к проблемам, освещаются на конкретных примерах. В результате подчеркивается важность правильного выбора стратегии в зависимости от этапа проекта, опыта команды и скорости разработки.

Ключевые слова: Git, система контроля версий, Git Flow, Trunk-Based Development, TBD, ветка, pull request, разработка программного обеспечения, командная работа, CI/CD

Kirish: Zamonaviy dasturiy ta’minot ishlab chiqish jarayonlari murakkablik darajasining oshishi va jamoaviy ishlash zaruratining ortib borishi bilan ajralib turadi. Bunday sharoitda kod ustida bir nechta dasturchining bir vaqtning o‘zida ishlashi, o‘zgarishlarni kuzatib borish va muvofiqlashtirish muhim masalaga aylanadi. Shu nuqtayi nazardan, *versiya boshqaruv tizimlari (Version control systems, VCS),* ayniqsa Git, dasturiy loyihalarni samarali yuritishning ajralmas qismidir.

Git – bu ochiq manbali, taqsimlangan versiya boshqaruv tizimi bo‘lib, u dasturchilarga koddagi har bir o‘zgarishni saqlash, tarixini kuzatish, turli filiallar (branch) ustida ishlash va jamoa bilan hamkorlikda muammosiz ishlash imkonini beradi. Git Flow kabi strategiyalar yordamida kodni strukturaviy boshqarish va ishlab chiqarish jarayonini tizimli tashkil etish mumkin.

Git, dasturiy ta’minot ishlab chiquvchilari loyihalarini ishlab chiqishda o‘zlarining fayllarini papkalarga zaxiralashardi. Bu loyihaga ega bo‘lgan dasturchi uchun muammo emas. Biroq, bu bir nechta dasturchilarni o‘z ichiga olgan jamoaga kelganda jiddiy muammolarni keltirib chiqaradi. Ishlab chiquvchilar markazlashtirilgan tizimga ega

bo‘lmaganligi sababli, har kim o‘zining zaxira nusxasida ishlagan va butun loyiha davomida ishlab chiqilgan fayllar holatini kuzatib bora olmagan. Natijada, dasturchilar uchun vaqt yo‘qotishlari mavjud edi.

Versiyalarni boshqarish tizimi – loyiha ishlab chiquvchilari tomonidan ma’lum muddatlarda zaxira nusxasini yaratish va kerak bo‘lganda zaxiralangan eski versiyaga osongina qaytish imkonini beruvchi tizimdir.

Ko‘pincha ma’lumotlar fayllar yoki hujjatlar sifatida ko‘riladi va ushbu fayllardagi o‘zgarishlarni kuzatib boriladi. Bu alohida fayllar haqidagi sezgilarga mos keladi, lekin identifikator o‘zgarganda, masalan, fayllar nomini o‘zgartirish, bo‘lish yoki birlashtirish paytida muammolarni keltirib chiqaradi. Shunga ko‘ra, Git kabi ba’zi tizimlar o‘rniga ma’lumotlarga o‘zgarishlarni bir butun sifatida ko‘rib chiqadilar, bu oddiy o‘zgarishlar uchun kamroq intuitiv, ammo murakkabroq o‘zgarishlarni soddalashtiradi.

Agar bir nechta odam bitta ma’lumotlar bazasi(to‘plami) ustida ishlayotgan bo‘lsa, ular bilvosita ma’lumotlar bazasining boshqa oqimini (ishchi nusxalarini yoki ingl. Branches) yaratib olishadi va bu keyinchalik ma’lumotlarni birlashtirishda muammolar keltirib chiqaradi. Oddiy kollabrativ hujjat tahrirlashda fayllarni bloklash yoki kimdir ishlayotgan hujjatda boshqa insonning ishlashini oldini olish orqali bu muammo yechiladi.

Dasturiy injiniringida Versiya Boshqarish tizimi (shuningdek, manba nazorat tizimi, manba kodi boshqaruv tizimi sifatida ham ma’lum) kompyuter dasturlari, dokumentatsiyalar, yirik veb-saytlar yoki boshqa ma’lumotlar bazasidagi o‘zgarishlarni nazorat qilish va boshqarish uchun ishlab chiqilgan tizim. Versiya nazorati dasturiy ta’minot konfiguratsiyasini boshqaruvchi komponent hisoblanadi.

Dasturiy ta’minot manbasi kodlaridagi o‘zgarishlar sonlar yoki harflar yordamida identifikatsiyalanadi. Misol uchun, dastlabki kod fayllari to‘plami „Versiya 1“ deb nomlanadi, birinchi o‘zgartirish sodir bo‘lgandan so‘ng esa „Versiya 2“ va shu tarzda tizimdagi o‘zgarishlar qayd etib boriladi. Ushbu versiyalar bir-biriga solishtirilishi, qayta tiklanishi va birlashtirilishi mumkin.

Versiyalar boshqaruv tizimlari ko‘pincha mustaqil ilova sifatida ishlaydi, biroq matn protsessorlari va elektron jadvallar, veb hujjatlar, va kontentni boshqarish tizimlari(ingl. CMS) kabi turli xil dasturiy ta’minotga o‘rnatiladi. Qayta tiklash tizimi (Revision Control) hujjatni oldingi tahririga qaytarish imkonini beradi, bu muharrirlarga bir-birlarining tahrirlarini kuzatish, xatolarni tuzatish uchun muhim ahamiyatga ega.

Git Flowning ijobiy va salbiy tomonlari: Ko‘rib turganingizdek, pull requests (tortish so‘rov) har doim ham eng yaxshi tanlov bo‘lmasligi mumkin. Ulardan faqat tegishli joylarda foydalanish kerak.

Quyidagi holatlarda Git Flow yaxshi samara beradi:

- ❖ Ochiq kodli loyihani ishga tushirganingizda: Bu uslub ochiq kodli dunyodan kelib chiqadi va u yerda eng yaxshi ishlaydi. Hamma hissa qo‘sha olishi mumkinligi sababli, siz barcha o‘zgarishlarga juda qattiq kirish huquqiga ega bo‘lishni xohlaysiz. Siz kodning har bir satrini tekshirish imkoniyatiga ega bo‘lishni xohlaysiz, chunki

ochig‘i, siz odamlarning hissa qo‘shishiga ishonishingiz mumkin emas. Odatda, bu tijorat loyihalari emas, shuning uchun ishlab chiqish tezligi tashvish tug‘dirmaydi.

❖ Agar sizda ko‘plab yosh dasturchilar bo‘lsa: Agar siz asosan yosh dasturchilar bilan ishlasangiz, ularning ishini diqqat bilan tekshirish usuliga ega bo‘lishingiz kerak. Siz ularga ishlarni qanday qilib samaraliroq bajarish bo‘yicha bir nechta maslahatlar berishingiz va ularning ko‘nikmalarini tezroq oshirishga yordam berishingiz mumkin. Pull requestlarini qabul qiladigan odamlar kod sifatining yomonlashuvining oldini olish uchun takroriy o‘zgarishlar ustidan qat’iy nazoratga ega.

❖ Agar sizda o‘rnatilgan mahsulot bo‘lsa: Bu uslub, shuningdek, sizda allaqachon muvaffaqiyatli mahsulot bo‘lsa ham yaxshi ishlaydi. Bunday hollarda, odatda dastur ishlashi va yuklash imkoniyatlariga e’tibor qaratiladi. Bunday optimallashtirish juda aniq o‘zgarishlarni talab qiladi. Odatda, vaqt cheklov emas, shuning uchun bu uslub bu yerda yaxshi ishlaydi. Bundan tashqari, yirik korxonalar bu uslub uchun juda mos keladi. Ular har bir o‘zgarishni diqqat bilan nazorat qilishlari kerak, chunki ular ko‘p million dollarlik investitsiyalarini buzishni xohlamaydilar.

Git Flow qachon muammolarga olib keluvchi holatlar:

➢ Agar siz endigina boshlayotgan bo‘lsangiz: Agar siz endigina boshlayotgan bo‘lsangiz, unda Git oqimi siz uchun emas. Ehtimol, siz tezda minimal hayotiy mahsulot yaratmoqchisiz. Pull request(tortish so‘rov)larni bajarish butun jamoani sezilarli darajada sekinlashtiradigan katta to‘siqni yaratadi. Siz bunga shunchaki qodir emassiz. Git oqimining muammosi shundaki, pull requestlar ko‘p vaqt talab qilishi mumkin. Shunchaki tez rivojlanishni shu tarzda ta’minlashning iloji yo‘q.

➢ Tez takrorlash kerak bo‘lganda: Mahsulotingizning birinchi versiyasiga yetganingizdan so‘ng, mijozlaringiz ehtiyojlarini qondirish uchun uni bir necha marta aylantirishingiz kerak bo‘ladi. Yana bir bor, bir nechta filiallar va pull (tortish) so‘rovlari ishlab chiqish tezligini sezilarli darajada pasaytiradi va bunday hollarda tavsiya etilmaydi.

➢ Asosan katta dasturchilar bilan ishlaganingizda: Agar sizning jamoangiz asosan uzoq vaqt davomida bir-biri bilan ishlagan katta dasturchilardan iborat bo‘lsa, unda sizga yuqorida aytib o‘tilgan pull request mikro-boshqaruvi kerak emas. Siz dasturchilaringizga ishonasiz va ularning professional ekanligini bilasiz. Ularga o‘z ishlarini bajarishga ruxsat bering va Git oqimi byurokratiyasi bilan ularni sekinlashtirmang.

Stvolga asoslangan ishlab chiqish - bu versiyalarni boshqarish amaliyoti bo‘lib, unda ishlab chiquvchilar kichik, tez-tez yangilanishlarni asosiy “stvol” yoki asosiy filialga birlashtiradilar. U birlashish va integratsiya bosqichlarini soddalashtirganligi sababli, CI/CD ga erishishga yordam beradi va dasturiy ta’minotni yetkazib berish va tashkilot faoliyatini yaxshilaydi.

Dasturiy ta’minotni ishlab chiqishning dastlabki kunlarida dasturchilar zamonaviy versiyalarni boshqarish tizimlaridan foydalanish imkoniyatiga ega emas edilar. Aksincha,

ular o‘zgarishlarni kuzatish va kerak bo‘lganda ularni o‘zgartirish vositasi sifatida dasturiy ta’minotning ikkita versiyasini bir vaqtning o‘zida ishlab chiqdilar. Vaqt o'tishi bilan bu jarayon ko‘p mehnat talab qiladigan, qimmatga tushadigan va samarasiz bo‘lib chiqdi.

Versiyalarni boshqarish tizimlari rivojlanib borishi bilan turli xil ishlab chiqish uslublari paydo bo'ldi, bu dasturchilarga xatolarni osonroq topish, hamkasblari bilan parallel ravishda kod yozish va versiya tezligini oshirish imkonini berdi. Bugungi kunda ko‘pchilik dasturchilar sifatli dasturiy ta’minotni yetkazib berish uchun ikkita ishlab chiqish modelidan birini - Gitflow va trunk-asosidagi ishlab chiqishni qo‘llaydilar.

Trunk-Based Development (TBD) muammolarga olib kelishi mumkin bo‘lgan holatlar quyidagilar:

- *Ochiq kodli loyihani ishga tushirganingizda*: Agar siz ochiq kodli loyihani ishga tushirayotgan bo‘lsangiz, Git flow yaxshiroq variant hisoblanadi. O‘zgarishlar ustidan juda qattiq nazoratga muhtojsiz va hissa qo‘shuvchilarga ishonishingiz shart emas. Axir, har kim hissa qo‘shishi mumkin. Shu jumladan onlayn trollar ham.
- *Agar sizda ko‘plab yosh dasturchilar bo‘lsa:* Agar siz asosan yosh dasturchilarni yollasangiz, ularning nima qilayotganini qat’iy nazorat qilish yaxshiroqdir. Qattiq tortish so‘rovlari ularga o‘z ko‘nikmalarini oshirishga yordam beradi va potentsial xatolarni tezroq topadi.
- *Agar siz mahsulot yaratgan bo‘lsangiz yoki katta jamoalarni boshqarsangiz:* Agar sizda allaqachon muvaffaqiyatli mahsulot bo‘lsa yoki ulkan korxonada katta jamoalarni boshqarsangiz, Git oqimi yaxshiroq fikr bo‘lishi mumkin. Siz millionlab dollarga teng bo‘lgan yaxshi tashkil etilgan mahsulot bilan nima sodir bo‘layotganini qat’iy nazorat qilishni xohlaysiz. Ehtimol, ilova ishlashi va yuklash imkoniyatlari eng muhim narsalardir. Bunday optimallashtirish juda aniq o‘zgarishlarni talab qiladi.

Xulosa. Git – ochiq manbali, taqsimlangan versiya boshqaruv tizimi bo‘lib, dasturchilarga kod o‘zgarishlarini ishonchli saqlash, tarixini kuzatish, parallel ishlash va jamoaviy hamkorlikni ta’minlaydi. Bu orqali loyihani boshqarish soddalashadi va samaradorlik oshadi.

Biroq, loyiha turiga qarab turli strategiyalar qo‘llaniladi: ochiq kodli loyihalar, yosh dasturchilar va barqaror mahsulotlarda Git Flow nazorat va sifat uchun afzal bo‘lsa, yangi loyihalar va tajribali jamoalarda Trunk-Based Development tezlik va sodda jarayon uchun yaxshiroqdir. Tanlov loyihaning bosqichi va jamoa tajribasiga bog‘liq.

Adabiyotlar ro‘yxati:

1. Driessen, V. (2010). *A successful Git branching model*. nvie.com. https://nvie.com/posts/a-successful-git-branching-model/ (Git Flow strategiyasining asosiy manbai)
2. Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development* (2-nashr). O‘Reilly Media.
3. Paul Hammant va boshqalar. *Trunk Based Development*. Rasmiy sayt:

https://trunkbaseddevelopment.com/ (Trunk-Based Development haqida asosiy resurslar)

4. Atlassian. *Gitflow Workflow* va *Trunk-based development*. Atlassian Git Tutorials. https://www.atlassian.com/git/tutorials/comparing-workflows

5. Google Cloud. *DevOps Tech: Trunk-based development*. https://cloud.google.com/architecture/devops/devops-tech-trunk-based-development

6. Neely, S., & Stolt, S. (2013). *Continuous Delivery vs. Feature Branching*. (TBD va Git Flow solishtirish bo‘yicha tadqiqotlar)