MURAKKAB DASTURIY TIZIM HAYOTIY SIKLINI TA’MINLASH UCHUN INSTRUMENTAL VOSITALARNI TANLASH VA TAHLIL QILISH

Muallif: **Xo‘jamurodov Abdulaziz**
Samarqand davlat universiteti Urgut filiali, 3-bosqich talabasi
abdulazizxo‘jamurodov01@gmail.com
Ilmiy rahbar: **Suyarov Akram**
Samarqand iqtisodiyot va servis instituti dotsenti
akramsuyarov@mail.ru

Annotatsiya: Ushbu maqolada dasturiy ta’minotning murakkab tizimlarini ishlab chiqish, ularga xizmat ko‘rsatish va boshqarish jarayonlarida instrumental vositalarni tanlash zarurati haqida so‘z boradi. Shuningdek, zamonaviy dasturiy vositalarning funksiyalari, afzalliklari va ular yordamida hayotiy siklni qanday ta’minlash mumkinligi yoritilgan.

Kalit so‘zlar: murakkab tizim, hayotiy sikl, instrumental vositalar, dasturiy ta'minot, boshqaruv, tahlil.

Kirish

Murakkab dasturiy tizimlar ko‘plab foydalanuvchilar, modullar, texnologiyalar va xizmatlarni birlashtiradi. Ularni yaratish, boshqarish va yangilab borish uchun aniq reja va mos vositalar talab qilinadi. Dasturiy tizimning hayotiy sikli bu — uni loyihalashdan boshlab, foydalanishga topshirish va keyingi texnik xizmatgacha bo‘lgan jarayonlarning to‘liq majmuasidir. Bu jarayonni to‘g‘ri tashkil qilish uchun maxsus instrumental vositalardan foydalaniladi.

1. Dasturiy tizim hayotiy sikli bosqichlari

Murakkab tizimlar hayotiy sikli quyidagi asosiy bosqichlardan iborat:

1. **Talablarni yig‘ish va tahlil qilish**
2. **Loyihalash (design)**
3. **Ishlab chiqish (development)**
4. **Testlash (testing)**
5. **Joriy etish (deployment)**
6. **Texnik xizmat ko‘rsatish (maintenance)**
7. **Monitoring va yangilash (update & support)**

Har bir bosqichda vazifalarni osonlashtirish uchun maxsus dasturiy vositalar ishlatiladi.

2. Instrumental vositalar va ularning funksiyasi

2.1. Talablarni boshqarish vositalari

- **JIRA**, **IBM DOORS**, **Confluence**

Talablarni yig‘ish, saqlash, tahlil qilish va ularni jamoa bilan boshqarish imkonini beradi.

2.2. Modellash va dizayn vositalari

- **Enterprise Architect**, **Visual Paradigm**, **Lucidchart**, **Draw.io**

UML diagrammalari, klass diagrammalari, sekvensiya va arxitektura diagrammalarini yaratish uchun ishlatiladi.

2.3. Ishlab chiqish muhiti (IDE)

- **Visual Studio**, **IntelliJ IDEA**, **Eclipse**, **PyCharm**

Dasturiy kod yozish, sinash, versiyalarni boshqarish (Git bilan integratsiya) imkonini beradi.

2.4. Testlash vositalari

- **Selenium**, **JUnit**, **Postman**, **TestRail**

Avtomatlashtirilgan testlar yaratish, ularni bajarish va natijalarni ko‘rish imkonini beradi.

2.5. Versiyalarni boshqarish tizimlari

- **Git**, **GitHub**, **GitLab**, **Bitbucket**

Loyihadagi o‘zgarishlarni kuzatish, jamoaviy ishlash va xavfsizlikni ta’minlash.

2.6. Monitoring va xizmat vositalari

- **Nagios**, **Zabbix**, **Grafana**, **Prometheus**

Tizim holatini kuzatish, xatoliklarni aniqlash va vaqtida xabar berish.

3. Instrumental vositalarni tanlash mezonlari

Dasturiy vositalarni tanlashda quyidagi omillar hisobga olinadi:

Mezoni	Tavsifi
Moslik	Loyihaning texnologik talablariga mos kelishi
Foydalanish qulayligi	Interfeysning tushunarli va sodda bo‘lishi
Jamoaviy ishlash imkoni	Bir nechta foydalanuvchi tomonidan birgalikda ishlash imkoniyati
Xavfsizlik	Ma’lumotlarni saqlash va o‘zgartirishda himoyalash

Litsenziya turi	Ochiq kodli yoki tijorat asosida foydalanish imkoniyati

4. Amaliy misol: Dasturiy tizim ishlab chiqish jarayonida vositalardan foydalanish

Aytaylik, bir tibbiy muassasa uchun elektron tibbiy karta tizimi ishlab chiqilmoqda. Quyidagi vositalar tanlanadi:

- **Talablar uchun:** JIRA + Confluence
- **Modellashtirish uchun:** Visual Paradigm
- **Ishlab chiqish uchun:** Visual Studio (C#)
- **Testlash uchun:** Postman (API) + Selenium (veb-testlar)
- **Monitoring:** Grafana + Prometheus

Bu vositalar o‘zaro integratsiya qilingan holatda butun hayotiy siklni qamrab oladi va samaradorlikni oshiradi.

Xulosa: Murakkab dasturiy tizimlarni ishlab chiqishda har bir bosqich uchun mos instrumental vositani tanlash muhimdir. To‘g‘ri tanlangan vositalar jamoaning ishini yengillashtiradi, xatolarni kamaytiradi va sifatli dastur yaratishga yordam beradi. Standartlarga asoslangan, samarali tanlovlar orqali tizimning hayotiy siklini to‘liq va mustahkam boshqarish mumkin.

Foydalanilgan adabiyotlar

1. Sommerville I. – *Software Engineering*, 10th Edition
2. Pressman R. – *Software Engineering: A Practitioner’s Approach*, 8th Edition
3. ISO/IEC 12207: Software Lifecycle Processes
4. JIRA, Confluence, Visual Paradigm rasmiy hujjatlari
5. www.guru99.com – Test va boshqaruv vositalar sharhlari
6. Software Testing Help (online resurslar)