# Дистрибутирани системи за управление на сорс код

Зорница Атанасова Костадинова, 4 курс, КН, фн: 80227,
Искрен Ивов Чернев, 4 курс, КН, фн: 80246

2 февруари 2011 г.

**Цел на проекта**

Да запознае читателя с историята и развитието на системите за управление на код (SCM). Ще бъдат сравнени централизираните и дистрибутираните системи както на високо ниво - предимства, недостатъци, сфери на приложение - така и на ниско ниво - обща архитектура, представяне на данните, използвани алгоритми и структури от данни. Ще бъде обърнато специално внимание на дистрибутираните системи за управление на код, тъй като те са по-нови като концепция и вече успешно заместват централизираните системи във все повече проекти, най-забележимо тези с отворен код, но също и в големи компании които по исторически причини използват централизирани системи (google, facebook).

# Съдържание

# 1 Увод

Системите за управление на код (SCM) играят важна роля в процеса на разработване на софтуер. Те съхраняват историята на развитие на файловете като по този начин позволяват на потребителя да прегледа направените промени по различни критерии (времеви период, потребител направил промяната и др.). Правенето на промени по кодовата база също е благоприятствано от факта, че винаги може да се игнорират и състоянието на проекта да бъде върнато към по-старо и стабилно състояние. Историята на развитие на проекта може да бъде използвана и от хора интересуващи се от прогреса по проекта (мениджъри, клиенти), с цел създаване на план за по-нататъшно развитие, оценка за свършена работа и други.

Софтуерните проекти обикновено се развиват в няколко направления:

- едно или няколко стабилни направления - използват се от обикновения потребител;
- направление за тестване (beta версия) - използват се от по-напреднали потребители, които искат да получат нововъведенията колкото се може по-скоро, на цената на по-нестабилно изпълнение;
- направление за развитие - използва се от програмистите докато разработват най-новите промени по кода.

SCM предоставят възможности за управление на отделните направление, като по този начин може да се разграничи кои версии от историческото развитие се използват от програмисти, тестери и обикновени потребители.

# 2 История и развитие на SCM

Нуждата от SCM се появява през $60^{\text{те}}$ и $70^{\text{те}}$ години на XX век. През това време се появяват първите по-големи софтуерни проекти и става ясно, че поддържането на историята на развитие на проекта е крайно наложително. Развитието преминава през няколко периода:

## 2.1 Ръчно управление

В началото всеки сам е решавал проблема с пазене на история на проект или отделен файл. Нужен е бил механизъм за запаметяване на състоянието на един или няколко файла в един момент, за да може при нужда да се върне старо състояние. Това може да става със запазване на стара версия на файла с различно разширение (например `filename.old.1`, `filename.old.2` и т.н.) или създаване на архив на цяла директория. Тези подходи вършат работа при малки начинания, но имат големия недостатък че изразходват много памет. Ако един файл има дължина 1 Kb, и е бил запазен 20 пъти по време на своето създаване, тогава заеманата памет е приблизително $1000 * 20/2 = 10\text{Kb}$. Това означава, че заеманата памет нараства линейно с нарастването на запаметените ревизии. На практика обаче, всяка ревизия се различава с малко от предходната, т.е. заеманата памет теоретично може да бъде пропорционална само на размера на проекта, независимо от броя на ревизиите.

## 2.2 Локално управление

Първите софтуерни продукти за управление на код са работели на ниво отделен файл. Т.е. те са запазвали историята на един файл независимо от останалите. Това може да бъде постигнато като за всяка нова ревизия бъде запазена разликата с предишната. Първата система, която реализира това е SCCS[2]. Тя използва формата припокриващи се разлики (interleaved deltas) за да запазва различията между версиите на един файл. Системата е била разработвана от Bell Labs и се разпространявала чрез заплащане. RCS[3] е наследник на SCCS и набрала голяма популярност през $80^{\text{те}}$ години, защото била безплатен и развит еквивалент на SCCS. RCS също предоставяла възможност за следене на история на всеки файл по отделно. Освен работната версия на файла се пазел и файл с историята, съдържащ списък с различията еволюирали файла от началното до крайното му състояние.

## 2.3 Централизирано управление

Системите до момента предоставят възможност за пазене на историята на отделни файлове на компютъра на който са били създадени. Често тази история е трябвало да се споделя между група хора - например екип програмисти. Дистрибутирането на файла с историята между няколко компютъра се оказала не-тривиална задача, особено при наличието на много файлове. Нужна била система, която е в състояние да управлява всички файлове на проекта и да осигури лесен и бърз начин за достъпване на историята на проекта от всеки член на екипа. С тези идеи бил създаден CVS[4] - една от най-известните системи за управление на код, използвана и в момента от немалко проекти. В нея било имплементирано запазването на история за всеки файл, точно както RCS както и централен сървър на който стояла информация за историята на всички файлове. Всеки клиент пазел само една текуща версия на всички файлове и при нужда от други версии се свързвал със сървъра и изтеглял нужната му информация. За да поправи много от проблемите открити с времето в CVS бил създаден Subversion[5]. Subversion има мото "CVS направен както трябва". Тъй като наистина решавал практични проблеми, налични в CVS, а също бил изключително лесен за използване много проекти използващи CVS преминават на Subversion. При Subversion се запазва идеята за централен сървър, на който стои цялата история на всички файлове като при нужда клиентите изтеглят нужната им информация.

## 2.4 Дистрибутирано управление

Дистрибутираните системи за управление на код приличат на централизирана система, при която всеки клиент пази пълната история на развитие на проекта. Тъй като всеки клиент държи пълно копие на историята и добавя сам нова история трябва да съществува механизъм за обмяна на промените направени между два клиента. Това означава, че може да се симулира модела на работа на централизираните системи като се избере един централен клиент и всички останали синхронизират с него. На практика се използва както централен клиент, така и синхронизиране на два нецентрални клиента, в зависимост от ситуацията. Най-видните примери за дистрибутирани

системи за управление са Git[6] и Mercurial[7]. Те набират голяма популярност сред опън сорс обществото, защото позволяват по естествен начин да се споделят промени направени от хора, който не се познават предварително и следователно няма как да споделят общ централен сървър.

## Литература

[1] http://example.com/

[2] http://sccs.berlios.de/

[3] http://www.gnu.org/software/rcs/

[4] http://savannah.nongnu.org/projects/cvs

[5] http://subversion.apache.org/

[6] http://git-scm.com/

[7] http://mercurial.selenic.com/

[8] http://ftp.newartisans.com/pub/git.from.bottom.up.pdf