7.Управление на конфигурации- Резюме

-Поради факта, че софтуерът се променя често, системите(С) могат да бъдат разглеждани като набор от версии, всяка от които трябва да бъде поддържана и управлявана.

- Версиите реализират предложения за промени, корекции на грешки и адаптации за различни хардуерни и операционни системи.

- Управлението на конфигурацията (УК) се занимава с политики, процеси и инструменти за управление на променящи се софт. С. Необходимо ви е УК, защото е лесно да се загуби следа от това какви промени и версии на компоненти са били вкл. във всяка версия на С.

Дейности в управлението на конфигурацията

Управление на промените- Проследяване на заявки за промени в софтуера от клиенти и разработчици, анализ на разходите и въздействието на промените, и вземане на решения дали промените да бъдат внедрени.

- **Управление на версиите**- Проследяване на многобройни версии на компоненти на системата и гарантиране, че промените, направени от различни разработчици, не се намесват помежду си.

-**Изграждане С**- Процесът на сглобяване на програмни компоненти, данни и библиотеки и тяхното компилиране, за да се създаде изпълнима С.

-Управление на Release- Подготвяне на софтуер за външно пускане и проследяване на версиите на системата, които са били пуснати за използване от клиентите.

Configuration item or software configuration item (SCI) - Всеки елемент, свързан със софтуерен проект (дизайн, код, тестови данни, документ и др.), който е поставен под конфигурационен контрол.

Често има различни версии на даден **configuration item**. Всеки **configuration item** има уникално име.

Configuration control - Процесът на гарантиране, че версиите на C и компонентите са записани и поддържани, така че промените да бъдат управлявани и всички версии на компонентите да бъдат идентифицирани и съхранявани през целия живот на C.

Version - Екземпляр на **configuration item**, който се различава по някакъв начин от други екземпляри на този елемент. Всяка **version** има уникален идентификатор, който често се състои от името на **configuration item** плюс номер на версията.

Baseline е **колекция от версии** на компоненти, които изграждат дадена система. **Baselines** са контролирани, което означава, че версиите на компонентите, съставляващи системата, не могат да бъдат променяни. Това гарантира, че винаги трябва да е възможно възстановяването на **baseline** от неговите съставни компоненти.

Codeline е набор от версии на софтуерен компонент и други **configuration items**, от които този компонент зависи.

Mainline - Последователност от baselines, представляващи различни версии на дадена система.

Release- Версия на C, пусната за клиенти и др. за използване.

Workspace - Лично работно пространство, в което софтуерът може да бъде модифициран, без да се засягат други разработчици, които също могат да го използват или променят.

Branching - Създаването на нова **codeline** от версия в съществуваща **codeline**. Новата **codeline** и съществуващата **codeline** след това могат да се развиват независимо една от друга.

Merging -Създаването на нова версия на софтуерен компонент чрез сливане на отделни версии от различни **codelines**. Тези **codelines** може да са били създадени от предишен **branch** на някоя от участващите **codelines**.

System building -Създаването на изпълнима версия на система чрез компилиране и свързване на подходящите версии на компонентите и библиотеките, които съставляват системата.

Управление на промените(УП)

-Организационните нужди и изисквания се променят през целия жизнен цикъл на С, грешките трябва да бъдат коригирани, а С – адаптирани към промени в тяхната среда.

-**УП** има за цел да гарантира, че еволюцията на системата е контролиран процес и че се дава приоритет на най-спешните и икономически ефективни промени.

-Процесът на **УП** е свързан с анализ на разходите и ползите от предложените промени, одобряване на онези, които са оправдани, и проследяване на компонентите в С, които са били променени.

Фактори при анализа на промените

-Последствията от неизвършването на промяната

-Ползите от промяната

-Броят на потребителите, засегнати от промяната

-Разходите за извършване на промяната

-Цикълът на пускане на продукта

Управлението на версиите (**Version Management - VM**) е процесът на проследяване на различните версии на софт. компоненти/ **configuration items**, както и на С, в които компонент. се използват.

-Включва също така гарантиране, че промените, направени от различни разработчици в тези версии, не си пречат взаимно.

-УП - процес на управление на **codelines** и **baselines**.

-**Codeline** е последователност от версии на изходен код, като по-късните версии са производни на по-ранните версии.

-**Codelines** обикновено се отнасят до компоненти на системите, така че за всеки компонент съществуват различни версии.

-Baseline представлява дефиниция на конкретна система.

-**Baseline** - версиите на компонентите, вкл. в С, както и спецификацията на използ. библиотеки, конфигурац. файлове и др.

Baselines могат да бъдат определяни чрез **configuration language**, който позволява дефиниране на компонентите, включени в дадена версия на конкретна система. **Baselines** са важни, тъй като често се налага възстановяване на определена версия на цялата система.

Системи за управление на версии

-**Идентификация на версиите и release -**Управляваните версии получават уникални идентификатори, когато бъдат качени в С.

Управление на съхранението - За да се намали необходимото дисково пространство за множество версии на компоненти, които се различават минимално, системите за управление на версии обикновено предоставят механизми за оптимизирано съхранение.

Запис на историята на промените - Всички промени, направени в кода на дадена С или компонент, се записват и съхраняват в регистър.

Независима разработка -С за управление на версии следи кои компоненти са били **checkout-нати** за редактиране и гарантира, че промените, направени от различни разработчици, не си пречат.

Поддръжка на проекти- С за управление на версии може да поддържа разработката на няколко проекта, които споделят едни и същи компоненти.

Codeline branches-Вместо линейна последователност от версии, която отразява промените в компонента с течение на времето, може да съществуват няколко независими последователности. Това е нормално при разработката на С, където различни разработчици работят независимо върху различни версии на изходния код и го променят по различни начини.

-В даден момент може да се наложи **сливане на codeline разклонения**, за да се създаде нова версия на компонент, която включва всички направени промени.

-Ако промените засягат различни части на кода, версиите на компонента могат да бъдат слети **автоматично**, като се комбинират **делтите** (разликите), които се отнасят до кода.

Управлението на конфигурацията (**Configuration Management - УК**) е процесът на управление на развиваща се софтуерна система. При поддръжка на система се сформира **УК екип**, който гарантира, че промените се интегрират **контролирано** и че се води документация с детайли за всички извършени промени.

Основните **процеси на управление на конфигурацията** са:

- **Управление на промените** (**Change Management**)
- **Управление на версиите** (**Version Management**)
- **Сглобяване на системата** (**System Building**)
- **Управление на изданията** (**Release Management**)

Управлението на промените включва оценяване на предложенията за промени от клиенти и други заинтересовани страни и вземане на решение дали е икономически ефективно тези промени да бъдат внедрени в нова версия на системата.

Управлението на версиите се занимава с проследяването на разл. версии на софт. компоненти, докато върху тях се правят промени.

Управление на конфигурацията и изграждане на системата

- **Изграждането на С** е процесът на създаване на цялостна, изпълнима С чрез компилиране и свързване на компонентите на С, външните библиотеки, конфигурационните файлове и др.
- Инструментите за изграждане С и инструментите за управление на версиите трябва да комуникират, тъй като процесът на изграждане включва извличане на версии на компоненти от хранилището, управлявано от С за управление на версиите.
- Конфигурационното описание, използвано за идентифициране на базовата линия, също се използва от инструмента за изграждане на С.

Build System Functionality

- **Генериране на build скриптове**
- **Интеграция със система за управление на версии**
- **Минимална повторна компилация**
- **Създаване на изпълнима система**
- **Автоматизация на тестовете**
- **Отчитане (Reporting)**
- **Генериране на документация**

Минимизиране на повторната компилация

- Инструментите за изграждане на система обикновено са създадени, за да **минимизират броя на необходимите компилации**.
- Те правят това, като проверяват дали вече съществува компилирана версия на даден компонент. Ако такава версия е налична, **няма нужда от повторна компилация**.
- **Уникален подпис (signature)** идентифицира всяка версия на сорс кода и обектния код и се променя, когато сорс кодът бъде редактиран.
- Чрез **сравняване на подписите** на сорс и обектните кодове може да се определи дали даден сорс код е бил използван за генериране на обектен код.

Идентифициране на файлове

Модификационни времеви марки (Timestamps)

- Подписът на сорс кодовия файл представлява **часът и датата на последната му модификация**.
- Ако сорс кодовият файл на даден компонент е бил модифициран след последната компилация на обектния файл, **системата предполага, че е необходима повторна компилация**.

Контролни суми на сорс кода (Checksums)

- Подписът на сорс кодовия файл може да бъде **контролна сума (checksum)**, изчислена от съдържанието на файла.
- Дори **една малка промяна в сорс кода** води до различна контролна сума, което гарантира, че файловете с различни контролни суми са наистина различни.

Timestamps срещу Checksums

Timestamps

- Понеже **сорс и обектните файлове са свързани по име**, обикновено не е възможно да се **изградят различни версии на един и същ сорс код в една и съща директория**, тъй като всички те ще генерират **обектни файлове със същите имена**.

Checksums

- При **повторна компилация на даден компонент**, вместо да презапише стария обектен файл (както се случва при timestamps), **се генерира нов обектен файл и се маркира с подписа на сорс кода**.
- Това позволява **паралелна компилация**, което означава, че различни версии на даден компонент могат да бъдат компилирани **по едно и също време**.

Agile Building

1. Извличане на основната система (**mainline system**) от **системата за управление на версии** в **личното работно пространство** на разработчика.
2. **Изграждане на системата** и изпълнение на автоматизирани тестове. Ако тестовете се провалят, **билдът се счита за неуспешен** и трябва да се уведоми разработчикът, който е качил последната версия.
3. Извършване на промени в компонентите на системата.
4. **Изграждане на системата в личното работно пространство** и повторно изпълнение на тестовете. Ако тестовете не преминат, **редактирането продължава**.

5. След като системата премине тестовете, **тя се изпраща към билд системата**, но **не се комитва като нова baseline версия**.
6. **Изграждане на системата на билд сървъра** и изпълнение на тестовете, за да се гарантира, че **други разработчици не са направили промени, които могат да доведат до грешки**.
7. Ако всички тестове преминат успешно, **промените се комитват в основната версия (mainline) като нова baseline версия**.

Daily Building (Ежедневно изграждане)

- Организацията задава **фиксирано време за доставка на компонентите** (например 14:00).
- Разработчиците трябва да предоставят **нови версии на компонентите до този час**.
- От тези компоненти се **изгражда нова версия на системата**.
- След това системата се **предоставя на екипа за тестване**, който изпълнява **предварително дефинирани тестове**.
- **Откритите грешки** се документират и се връщат на разработчиците за корекция.

Release Management

- **Системен release** представлява **версия на софтуерна система, която се разпространява до клиентите**.
- В масовия пазар на софтуер има два основни типа release:
    - **Major releases** – предлагат значителни нови функционалности.

    - **Minor releases** – съдържат поправки на грешки и подобрения, базирани на обратна връзка от клиенти.

- При **персонализиран софтуер**, всяка версия може да бъде различна за различни клиенти, като **едновременно могат да се поддържат няколко версии на системата**.

Release Tracking (Проследяване на версии)

- В случай на проблем, може да бъде необходимо **възпроизвеждане на точно определена версия** на софтуера, предоставена на даден клиент.
- Когато се създава release, той **трябва да бъде документиран**, за да може по-късно да бъде възстановен в същата форма.
- Това е особено важно за **вградени системи с дълъг жизнен цикъл**, които могат да бъдат използвани в продължение на **много години**.

Release Reproduction (Възпроизвеждане на версия)

- За да се документира release, трябва да се запазят:
    - **Специфичните версии на сорс кодовите компоненти**
    - **Изпълнимите файлове (executables), конфигурационни и данни файлове**
    - **Версиите на операционната система, библиотеките и инструментите, използвани при изграждането**

Release Planning (Планиране на release)

- Освен техническата работа, свързана със създаването на release, трябва да се подготвят **рекламни и маркетингови материали**, за да се убедят клиентите да преминат към новата версия.

- **Честота на release-ите**:
    - **Твърде чести release-и** могат да отблъснат клиентите, особено ако изискват **хардуерни ъпгрейди** или **допълнителни разходи**.
    - **Твърде редки release-и** могат да доведат до **загуба на пазарен дял**, тъй като клиентите могат да преминат към конкурентни продукти.

Release Components (Компоненти на release)

- Освен **изпълнимия код**, release-ът може да включва:
    - **Конфигурационни файлове**
    - **Файлове с данни (например съобщения за грешки)**
    - **Инсталационна програма**
    - **Документация (електронна и хартиена)**
    - **Опаковка и рекламни материали**

- **Factor Description**
- **Technical quality of the system**
- Ако бъдат докладвани **сериозни системни грешки**, които влияят върху начина, по който **много клиенти** използват системата, може да се наложи **пускане на fault repair release**.
  По-малки грешки могат да бъдат поправени чрез **patches** (обикновено разпространявани през Интернет), които могат да бъдат приложени към **текущата версия на системата**.
- **Platform changes**
- Може да се наложи **създаване на нов release** на софтуерно приложение, когато бъде **пусната нова версия на операционната система**.
- **Lehman's fifth law (виж Chapter 9)**
- Този „закон" гласи, че ако добавите **много нова функционалност** в системата, това вероятно ще **въведе нови бъгове**, които ще **ограничат обема на функционалността** в следващия release.
  Следователно, release с **значителни нови функционалности** може да се наложи да бъде последван от **release, фокусиран върху отстраняване на грешки и подобряване на производителността**.
- **Competition**

- При **масовия пазарен софтуер** може да се наложи **нов system release**, ако конкурентен продукт **въведе нови функционалности**. Ако тези функции не бъдат предоставени на съществуващите клиенти, може да настъпи **загуба на пазарен дял**.
- **Marketing requirements**
- Маркетинговият отдел на организацията може да е **поел ангажимент**, че нови release-и ще бъдат **налични на определена дата**.
- **Customer change proposals**
- При **персонализирани системи** клиентите може да са **поръчали и заплатили** за конкретен набор от **промени в системата** и очакват **release** веднага щом тези промени бъдат **изпълнени**.

Ключови точки

- **Изграждането на система** включва компилиране и свързване на компонентите в изпълним файл.
- Софтуерът трябва **често да се компилира и тества**, за да се откриват грешки на ранен етап.
- Управлението на release-ите включва **решения за дати на release, подготовка на документи и архивиране на версиите**.