Доклад

По Софтуерно инженерство

1. Етапи в разработката на софтуер

**Планиране**

Събират на бизнес изисквания, след това се провеждат срещи между заинтересованите страни, разработчиците трябва добре да се запознаят с изискванията на клиента. Предвиждат на евентуалните проблеми, които могат да възникнат.

**Анализ**

Екипът дефинира детайлно целия проект, като проверява неговата приложимост.

Работният процес е разделен на малки задачи, за да може разработчици, тестъри, дизайнери и мениджъри на проекта да дадат оценка за своите задачи.

Те оценят дали дадена задача е приложима относно цена, време, функционалност, надеждност и др.

**Дизайн**

Дефинира се дизайнът на софтуера на база събраните изисквания от фазата “Планиране”, определят се системните и хардуерните изисквания за проекта, след което се създава стратегия за тестване.

**Разработка**

Разработката е най-дългият етап в софтуерната разработка. След определяне на дизайна на продукта, работата се разпределя на малки задачи между всички участници.

**Тестване**

След като софтуерът е разработен, той се подлага на тестване. Софтуерът не преминава в следващ етап на разработка, ако тестърите не го одобрят. Намерените бъгове и проблеми се документират и софтуерът се връща в етап на разработка

**Инсталация и поддръжка**

Софтуера се предава на клиентите, за да се инсталира на техните устройства, а след това софтуера преминава в етап на поддръжка.

Поправят се възникнали грешки и софтуера се подновява от време на време

2.Методологии за разработка на софтуер

**Waterfall**

Всеки етап трябва да приключи преди да започне следващия.

Няма припокриване между етапите.

Състои се от вече споменатите етапи на софтуерната разработка.

**Agile (гъвкава методология)**

Комбинация от итеративни и инкрементални модели за разработка.

Разбива продукта на малки инкрементални части, а тези части се снабдяват в итерации.

Всяка итерация трае средно между 1 и 3 седмици.

Всяка итерация включва вече споменатите етапи на разработка на софтуер.

След всяка итерация работещият продукт се показва на клиента и заинтересованите страни.

**Scrum**

Най-популярният agile фреймуърк.

Фокусиран върху това как да се управляват задачи в среда за разработка базирана върху екипна работа.

Използва итеративен и инкрементален подход за разработка.

Кратки периоди на итерациите.

Сравнително лесна имплементация.

Фокус върху бързите и чести доставки на работещ софтуер

Състои се от Scrum екипи и техните роли, събития, артефакти и правила

Всеки компонент във фреймуърка има специфично значение.

**Kanban**

Визуален начин за управление на задачи и работен процес.

Използва kanban табло с колони и карти.

Всяка карта представлява определена задача.

Колоните организират задачите според техния прогрес или етапа на разработка, в който се намират.

Мениджърите на екипи могат бързо да назначат задача на даден екип, когато той е без такава.

**Непрекъсната интеграция**

Практика при разработката на софтуер.

Всеки екип трябва подсигурява, че успешен build и последващо тестване са проведени за всяка промяна по кода в софтуерна програма.

Концепцията гарантира премахването на проблема с откриване на грешки в по-късен етап от разработката.

Гарантира, че промените по кода никога не се правят в изолация.

3.Софтуерни изисквания

Изискванетое условие, необходимо на потребителя за решаване на проблем или постигане на цел. Това е първата стъпка в развитието на една система. Спецификацията на софтуерните изисквания изброява всички изисквания, заявени от потребителя по непоследователен начин.

Софтуерната документация се нарича още спецификация на софтуерните изисквания.

**Компоненти на спецификацията на софтуерните изисквания**

Спецификацията на софтуерните изисквания включва следните подробности

**Функционалност:** Насочва се към това, което софтуерът трябва да прави

**Производителност:** Отнася се до скоростта, времето за реакция, наличността, времето за възстановяване, софтуерната функция и т.н.

**Външен интерфейс:** Той разглежда как софтуерът взаимодейства с хората, хардуера на системата, друг хардуер и друг софтуер.

**Атрибути:** Насочва се към преносимостта, коректността, сигурността, надеждността, поддръжката и т.н.

**Ограничения на дизайна, наложени на внедряване:** Той се отнася до изискваните стандарти в сила, език за изпълнение, политики за интегритет на базата данни, ограничения на ресурсите, операционна среда и т.н.

4.Случаи на употреба (use cases) и Потребителски истории (user stories)

**Случаи на употреба (use cases)** –

Случаят на употреба е писмено описание на това как потребителите ще изпълняват задачи на вашия уебсайт. Той очертава, от гледна точка на потребителя, поведението на системата, докато тя отговаря на заявка. Всеки случай на употреба е представен като поредица от прости стъпки, започващи с целта на потребителя и завършващи, когато тази цел бъде изпълнена.

**Ползи от случаите на употреба**

Случаите на използване добавят стойност, защото помагат да се обясни как трябва да се държи системата и в процеса те също така помагат да се направи мозъчна атака какво може да се обърка. Те предоставят списък с цели и този списък може да се използва за установяване на цената и сложността на системата. След това екипите на проекта могат да преговарят кои функции стават изисквания и се изграждат.

**Потребителски истории (user stories)** –

Потребителските истории са част от гъвкав подход, който помага да се измести фокусът от писане на изисквания към говорене за тях. Всяка гъвкава потребителска история включва писмено изречение или две и, което е по-важно, предизвиква поредица от разговори относно характеристиките и функционалността, които представлява потребителската история.

5.**Спецификация на изискванията** –

Потребителските истории са част от гъвкав подход, който помага да се измести фокусът от писане на изисквания към говорене за тях.

Всяка гъвкава потребителска история включва писмено изречение или две и, което е по-важно, предизвиква поредица от разговори относно характеристиките и функционалността, които представлява потребителската история.

Потребителските истории са начин да се опише желаната функционалност на артикулите за изоставане на продукта. Потребителските истории с висок приоритет обикновено са по-подробни; потребителските истории с нисък приоритет обикновено са по-малко подробни.

Екипите добавят подробности, когато историите стават по-приоритетни, или чрез създаване на критерии за приемане, или чрез разделяне на големи истории на по-малки части (или и двете). Прочетете, за да откриете повече за потребителските истории и шаблона за потребителски истории, както и да видите някои примери за потребителски истории.

1.Етапи в разработката на софтуер –

<https://github.com/BG-IT-Edu/School-Programming/blob/main/Courses/Applied-Programmer/Software-Engineering-Concepts/01-course-introduction/course-introduction.pdf>

2.Методологии за разработка на софтуер –

<https://github.com/BG-IT-Edu/School-Programming/blob/main/Courses/Applied-Programmer/Software-Engineering-Concepts/01-course-introduction/course-introduction.pdf>

3.Софтуерни изисквания –

<https://www.educba.com/software-requirement-specification/>

4.Случаи на употреба (use cases) и Потребителски истории (user stories) - <https://www.usability.gov/how-to-and-tools/methods/use-cases.html> , <https://www.mountaingoatsoftware.com/agile/user-stories>

5.Спецификация на изискванията

<https://github.com/BG-IT-Edu/School-Programming/blob/main/Courses/Applied-Programmer/Software-Engineering-Concepts/04-software-requirements-and-prototyping/software-requirements-and-prototyping.pdf>

Изготвил: Валентин Кьосев 12,,а‘‘ клас