XP (Extreme Programming) методология

XP (Extreme Programming) методология ли е?

Не и Да. Не, това не е методология, в смисъла на многото документи и допълнителни рамки на разработчиците, през които трябва да преминат. Но да, всъщност е методология, по смисъла на това, че е повторяем процес за разработване на софтуер, въпреки че е една от леките методологии.

Extreme Programming (XP) е методология за разработка на софтуер, която е предназначена за подобряване на качеството на софтуера в съответствие с променящите се изисквания на клиентите. Като вид гъвкава методология за разработка на софтуер тя препоръчва често "освобождаване" в кратки цикли на развитие (timeboxing), което цели подобряване на производителността и въвеждане на нови контролно-пропускателни пунктове, където изискванията на клиента могат да бъдат приети.

Други елементи на „екстремното програмиране" (extreme programming) включват: програмиране по двойки или извършване на обширен преглед на кода, тестова единица на целия код, като се избягва програмиране на функции, докато те не са действително необходими, хоризонтална управленска структура, простота и яснота в кода, очакване на промени в изискванията на клиента в течение на времето и по-добро разбиране на проблема (промените), честа комуникация с клиента и между програмистите. Методологията носи името си от идеята, че се вземат полезните елементи от традиционните практики на софтуерното инженерство, на теория наречени "екстремни" нива.

Критиците отбелязват няколко потенциални недостатъка, включително проблеми с нестабилни изисквания, няма документация на компромисите на потребителски спорове, както и липсата на цялостна дизайн спецификация или документ.

Планиране и обратна връзка в Extreme Programming

Цели

Екстремното програмиране се описва, като дисциплина за разработка на софтуер, която организира хората да произвеждат по-високо качествен софтуер по-продуктивно.

XP се опитва да намали разходите за промените в изискванията, чрез няколко кратки цикли на развитие, вместо да има един дълъг цикъл. В тази доктрина промените са естествени, неизбежни и дори желателни аспект на проектите за софтуерна разработка и трябва да бъдат планирани вместо да се правят опити да се определя един стабилен набор от изисквания.
Екстремно програмиране също въвежда редица основни ценности, принципи и практики на върха на гъвкавата рамка за програмиране.

- **Дейности**

XP описва четири основни дейности, които са свързани с процеса на софтуерна разработка: **кодиране, тестване, слушане и проектиране**. Всяка от тези дейности е описана по-долу.

- ✓ **Кодиране**

Защитниците на XP твърдят, че единственият наистина важен продукт на процеса на софтуерна разработка е софтуерният код, който един компютър може да интерпретира. Без код, няма работещ продукт. Кодирането може да се използва, за да се намери най-подходящото решение на даден проблем.

- ✓ **Тестване**

При XP подхода малкото тестване може да елиминира няколко недостатъка, а при много тестове може да се елиминират много повече недостатъци.

- При тестването на дадена единица (Unit tests) се определя дали дадена функция работи по предназначение. Един програмист пише толкова автоматизирани тестове, колкото счита, че е необходимо, за да "разбие" кода, и ако всички тестове се изпълняват успешно, тогава кодирането е завършено. Всяка част от код, която е писана трябва да бъде тествана, преди да се премине към следващата характеристика.
- Приемните тестове верифицират, че изискванията са разбрани от програмистите и отговарят на действителните изисквания на клиента. Това се случва във фазата на проучване след приключване на планираната работа.

*„Testathon” е събитие, при което програмистите се срещат, за да си сътрудничат за писането на тестовете, то е нещо от типа „мозъчна атака” по отношение на софтуерното тестване.

- ✓ **Слушане**

Програмистите трябва да слушат това, което клиентите имат нужда системата да направи, каква "бизнес логиката" е необходима. Те трябва да разберат тези нужди достатъчно добре, за да предоставят обратна връзка на клиентите относно техническите аспекти на това, как проблемът може да бъде решен, или не може да бъде решен.

- ✓ **Проектиране**

С оглед улеснение, разбира се може да се каже, че развитието на системата не се нуждае повече от кодиране, тестване и слушане. Ако тези дейности се извършват добре, резултатът винаги трябва да бъде една система, която работи. На практика, това няма да работи. Човек може да извърви дълъг път, без проектирането, но в даден момент работата ще „заседне”. Системата става твърде сложна и зависимостите в рамките на системата спират да бъдат ясни. Човек може да избегне това, чрез създаване на структура на дизайн, която организира логиката в системата. Чрез добрият дизайн ще се избегнат много зависимости в рамките на системата, това означава, че промяната на една част няма да засегне и други части на системата.

Стойности

XP признава първоначално четири стойности през 1999 година. Във второто издание на XP (Extreme Programming Explained) е добавена и нова стойност. Така стойностите стават пет и са следните:

1. Комуникация

Изграждането на софтуерни системи изисква комуникация относно системните изисквания между разработчиците на системата. Във формалните методологии за разработка на софтуер, тази задача се постига, чрез документация. XP техниките за програмиране, може да се разглеждат, като методи за бързо изграждане и разпространяване на институционални знания между членовете на екипа за разработка. Целта е да се даде на всички разработчици общ изглед на системата, който съвпада с изгледа при потребители на системата. За тази цел, екстремното програмиране насърчава простият дизайн, съвместна работа на потребители и програмисти, честа вербална комуникация и обратна връзка.

2. Простота

Екстремното програмиране (Extreme Programming) насърчава работата да започне с най-лесното (просто) решение. Допълнителните функционалности могат да бъдат добавени по-късно. Разликата между този подход и по-конвенционалните методи за разработка на системи е фокусът върху проектирането и кодирането за нуждите на днешния ден, а не на тези от утрешният ден, следващата седмица или следващия месец. Привържениците на XP признават недостатъка, че това понякога може да доведе до повече усилия утре, за да се промени системата; твърдението им е, че това е повече от компенсирано, чрез предимството да не се инвестира в евентуални бъдещи изисквания, които може да се променят, преди да станат уместни. Кодиране и проектиране за несигурни бъдещи изисквания предполага риск от разходи за ресурси, които може да не бъдат необходими в бъдеще. Във връзка със стойността на "комуникацията", простотата в дизайна и кодирането трябва да се подобри качеството на комуникация. Един прост дизайн с много прост код може лесно да бъде разбран от повечето програмисти в екипа.

Обратна връзка

В XP обратната връзка се отнася до различни измерения на развитието на системата:

- **Обратна връзка от системата:** от писане на тестова единица (unit tests), или при провеждане на периодични тестове за интеграция, програмистите имат пряка обратна връзка от състоянието на системата след въвеждане на промени.
- **Обратна връзка от клиента:** функционални тестове (известен още като тестове за приемане) са написани от страна на клиента и тестери. Така те получават конкретна обратна информация за текущото състояние на тяхната система. Този преглед е планиран за извършване веднъж на всеки две или три седмици, така че клиентът може лесно да направлява развитието.
- **Обратна връзка от екип:** когато клиентите идват с новите изисквания в планирането екипа директно дава оценка на време, което ще му отнеме за изпълнение.

Обратната връзка е тясно свързана с комуникацията и простота. Пропуски в системата лесно се комуникират, чрез писане на тест, който доказва, че известна част от кода ще се счупи. Пряка обратна връзка от системата казва на програмистите да прекодира тази част. А клиентът е в състояние да тества системата периодично в съответствие с функционалните изисквания, което е известно като потребителски истории (user stories).

4. Кураж / Смелост (Courage)

Няколко практики включват в себе си смелостта. Един от тях е декрета, винаги дизайна и кода да са за днес, а не за утре. Това е опит да се избегне появата на затънал в дизайн проект изискващ много усилия за изпълнение на каквото и да било друго. Куража дава възможност на разработчиците да се чувстват комфортно с редакциите на техния код, когато е необходимо. Това означава прегледа на съществуващата система и промените така да се извършват, че бъдещите промени да могат да се осъществяват по-лесно. Друг пример за смелост е знанието кога кода може да бъде захвърлен: смелост да се премахне изходния код (source code), поради това, че е остарял, без значение, колко усилия са използвани за създаването му. Също така, кураж означава постоянство: Програмистът може да работи над сложен проблем целия ден, а след това да реши проблема бързо на следващия ден, само ако е постоянен.

5. Уважение

Стойността на уважението включва уважение към другите, както и самоуважение. Програмистите никога не трябва да извършат промени, които ще провалят съществуваща тестова единица, в противен случай това ще забави работата на техните колеги. Потребителите уважават тяхната собствена работа, като винаги се стремят за високо качество и търсят най-добрия дизайн за решение.

Приемането на четирите преходни стойности води до спечеленото от останалите в екипа уважение. Никой в екипа не трябва да се чувства неоценен или игнориран. Това гарантира високо ниво на мотивация и насърчава лоялност към екипа и към целта на проекта. Тази стойност е много зависима от другите стойности, и е много по-ориентирани към хората в екипа.

Правилник

Първата версия на правила за XP е публикувана през 1999 г. от Дон Уелс (Don Wells) в сайта на XP. 29 правила са дадени в категориите на планиране, управление, проектиране, кодиране и тестване. **Планиране, управление и проектиране** са изрично формулирани по този начин, за да се противопоставят на твърденията, че XP не поддържа тези дейности.

Друга версия на правилата на XP е била предложена от Кен Ауер (Ken Auer) в XP / Света на гъвкавите методологии 2003 (XP/Agile Universe 2003). Той усеща, че XP методологията е определена от нейните правила, а не от практиките й (които са обект на по-голяма вариация и двусмисленост). Той определя две категории: *"Правила за намеса"*, които диктуват на околната среда, в които разработки на софтуер може да се осъществи ефективно методологията, и *"Правила на Играта"*, които определят минута по минута дейности и правила в рамките на *правилата за намеса*.

- **Принципи**

Принципите, които формират основата на XP, се основават на току-що описаните стойности / ценности и са предназначени за насърчаване на вземането на решения в процеса на работа по проект за разработване на дадена система. Принципите са предназначени да бъдат по-конкретни от стойностите и по-лесно да се транслират в насоки в практически ситуации.

- **Обратна връзка**

Екстремното програмиране вижда обратната връзка, като най-полезна, ако се прави бързо и изразява, че времето между определена дейност и нейната обратна връзка е от решаващо значение за обучение и за извършване на промени. За разлика от традиционните методи за разработване на системи, при XP контакта с клиента се случва в по-чести интервали от време. Клиентът има ясен поглед върху системата, която се разработва. Той или тя може да даде обратна връзка и да направлява

развитието, както е необходимо. Тестовете (unit tests) също допринасят за принципа за бързо придобиване на обратна връзка. Когато се пише код, тестовата единица осигурява директна обратна връзка за това, как системата реагира на промените, които са направени. Ако, например, промените засягат част от системата, която не е в обхвата на програмист, който ги е направил, този програмист няма да забележи недостатъка. Има голям шанс този бъг да се появи, когато системата е в производство.

- **Възприемане на промяната**

Принципът на всеобхватна промяна, се отнася за това да не се работи срещу промените, а те да бъдат приемани. Например, ако в една от регулярните срещи изглежда, че изискванията на клиента са се променили драстично, програмистите трябва да приемат това и да планират новите изисквания за следващата итерация.

- **Спорни въпроси**

Практиките в XP сериозно са се обсъждали. Привържениците на екстремното програмиране (extreme programming) твърдят, че когато исканите промени от клиента са неформални и се извършват на място при него, процесът става гъвкав и спестява разходи за официални промени. Критиците на XP твърдят, че това може да доведе до скъпа преработка и пълзене покрай обхвата на проекта и извън това, което е било предварително договорено или финансирано.

Други потенциално спорни аспекти на Екстремно програмиране, включват:

- ✓ Изискванията да са изразени, като автоматизирани тестове за приемане, а не под формата на специфични документи.
- ✓ Изискванията да са определени постепенно, а не да се правят опити да се получават предварително.
- ✓ От софтуерните разработчици обикновено се изисква да работят по двойки.
- ✓ Няма голяма предварителна проектантска подготовка. Повечето от проектантска дейност се осъществява в движение и постепенно, започвайки с "най-простото нещо, което би могло да работи" и след това се добавят по-сложните неща.
- ✓ Представител на клиента е прикрепен към проекта. Тази роля може да стане единствената точка за провал на проекта и поради тази причина някои хора я намират за източник на стрес. Също така, има опасност ръководството на микро ниво, което няма технически познания, да се опитва да диктува използването на технически характеристики на софтуера и архитектура.

- **Скалируемост**

Исторически, XP работи само с екипи от дванадесет или по-малко хора. Един от начините да се заобиколи това ограничение е проекта да се раздроби на по-малки парчета, както и екипа на по-малки групи. Твърди се, че XP е била използвана успешно от екипи от над сто разработчици. Разумен успех на раздробени проекти ръководени по XP методологията са били също с до шестдесет души.

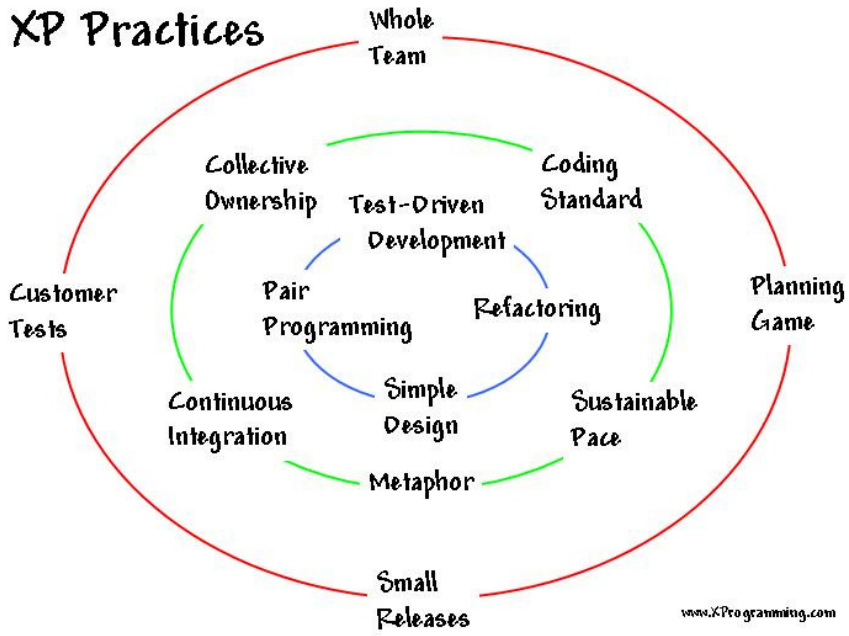
През 2004 г. Industrial Extreme Programming (IXP) е въведена като еволюция на XP. Тя е предназначена да доведе способността за работа в големи и разпределени екипи. Тя вече има 23 практики и гъвкави стойности. Тъй като това е нов член на Agile семейството, няма достатъчно данни, за да докаже своята използваемост, но той претендира да е отговор на това, което вижда, като несъвършенства в XP.

3.5. Заключение за XP

XP е набор от практики, които отговарят на ценностите и принципите на Agile методите. XP е дискретен метод, докато Agile е класификация. Има много Agile методи, XP е просто един от тях.

Както казах вече нито един от останалите Agile методи не са толкова добре дефинирани, или имат по-широк обхват, колкото XP. Scrum, например, е приблизително еквивалентна на XP Планирането (Planning game practice) с елементи на целия екип. Въпреки, че има разлики в детайлите, е справедливо да се каже, че Scrum е подмножество на XP. Наистина, много Scrum екипи увеличат своите процеси, чрез добавяне на много от практиките на XP, като например Проверка при приемане (Acceptance Testing), Програмиране по двойки (Pair Programming), Постоянна интеграция (Continuous Integration) и особено процес на разработване задвижван от тестове (Test Driven Development).

От всички методи на Agile, XP е единственият метод, който осигурява дълбоки и пълни дисциплини за начина, по който разработчиците си вършат ежедневната работа. От тези дисциплини, процеса на разработване задвижван от тестове (Test Driven Development) е най-революционният и въздействащ.



Фигура, показваща практики и основните "цикли" на XP