ЕКСТРЕМНО ПРОГРАМИРАНЕ

Доц. д-р Ася Стоянова-Дойчева

Distributed @learning center

ВЪВЕДЕНИЕ

- Създатели – Kent Beck, Ward Cunningham и Ron Jeffries – 1996 г.
- XP е подход за разработка на обектно-ориентиран софтуер, който е адаптивен и ориентиран към потребителите.

Всичко в софтуера се променя. Изискванията се променят. Проекта се променя. Бизнесът се променя. Технологията се промения. Екипът се променя. Членовете на екипа се променят.

Проблемът не е самата промяна, защото промени ще се случват непрекъснато. Проблемът по-скоро е нашата неспособност да се справим с промяната.

Kent Beck, eXtreme Programming Explained

КАКВО Е ЕКСТРЕМНОТО ПРОГРАМИРАНЕ?

- Екстремното програмиране е дисциплина от софтуерното разработване, базирана на **простота, комуникация, обратна връзка, кураж и уважение**.
- Успешно програмиране, защото набляга на клиентското участие
- Представлява елементарен и ефективен метод, даващ възможност за колективен начин на разработка

КАКВО Е ЕКСТРЕМНОТО ПРОГРАМИРАНЕ?

- **Простота** - при XP се започва с възможно най-опростен дизайн и решение на дадения проблем, който се подобрява, чрез рефакторинг като при този начин на програмиране се пише за днес, а не за утре.

- **Комуникация** - при XP се стимулира вербалната комуникация, за разлика от другите концепции, при който комуникацията става чрез документацията.

КАКВО Е ЕКСТРЕМНОТО ПРОГРАМИРАНЕ?

- **Обратна връзка**
  - *от системата* - с помощта на unit тестове или периодични интеграционни тестове (integration tests) програмистите имат директна обратна връзка от състоянието на системата след като промените са били имплементирани; с помощта на тази обратна връзка по-лесно може да се открие грешка в кода и дадения фрагмент да се пренапише;
  - *от клиента* - функционалните тестове са писани от клиента и от тестерите. Те ще получат ясна представа от моментното състояние на системата. Тези тестове са предвидени да се правят веднъж на 2-3 седмици, за да може клиентът от близо да следи развитието;
  - *от екипа* - след като клиентът даде новите си изисквания екипът веднага да може да даде конкретен отговор колко точно време ще отнеме да се имплементират новите изисквания.

КАКВО Е ЕКСТРЕМНОТО ПРОГРАМИРАНЕ?

- **Кураж** - кураж да правиш дизайн и пишеш код за днес, а не за утре; кураж да пренапишеш даден код, който не отговаря на новите изисквания, независимо от факта, че си му отделил много време и усилия. Куража помага на разработчиците да се чувстват добре, когато техния код има нужда от refactoring.

- **Уважение** - в XP членовете на екипа трябва се уважават един друг, защото се смята, че по-този начин се подобрява работата в екипа, при което се получават по-добри резултати

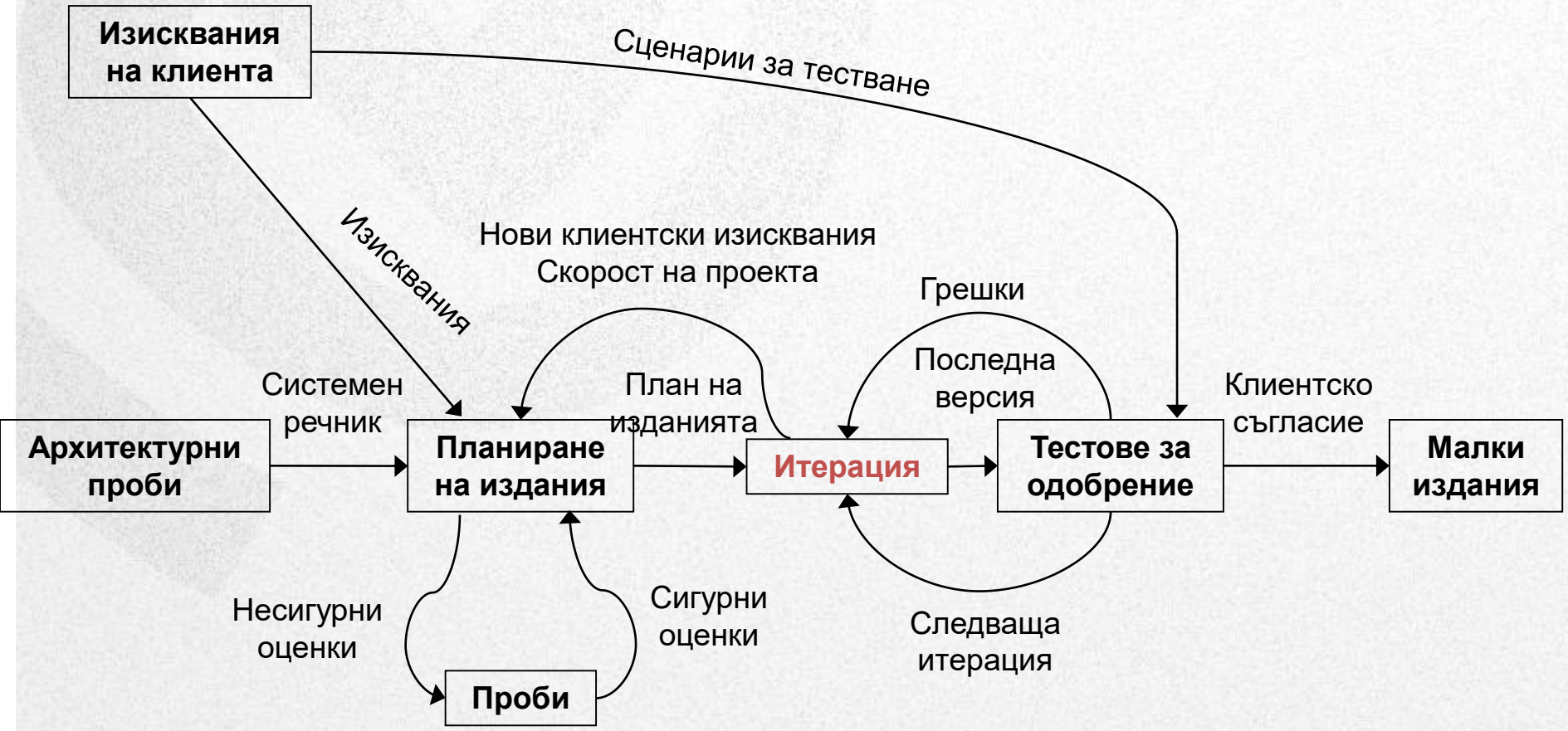
КАКВО Е ЕКСТРЕМНОТО ПРОГРАМИРАНЕ?

- Екстремното програмиране е създадено в отговор на проблема за променящите се изисквания. Клиентите не винаги имат категорична идея за това какво ще прави системата ;
- XP е пригодено за малки групи от програмисти между 2 и 12 човека. XP екипите не включват само разработчиците, а също така и мениджъри и клиенти;
- Друго изискване е тестването. Трябва да се създават автоматизирани единични и функционални тестове;
- XP проектите доказват по-голямата производителност на програмистите, в сравнение с други проекти, в рамките на същите среди на разработка.

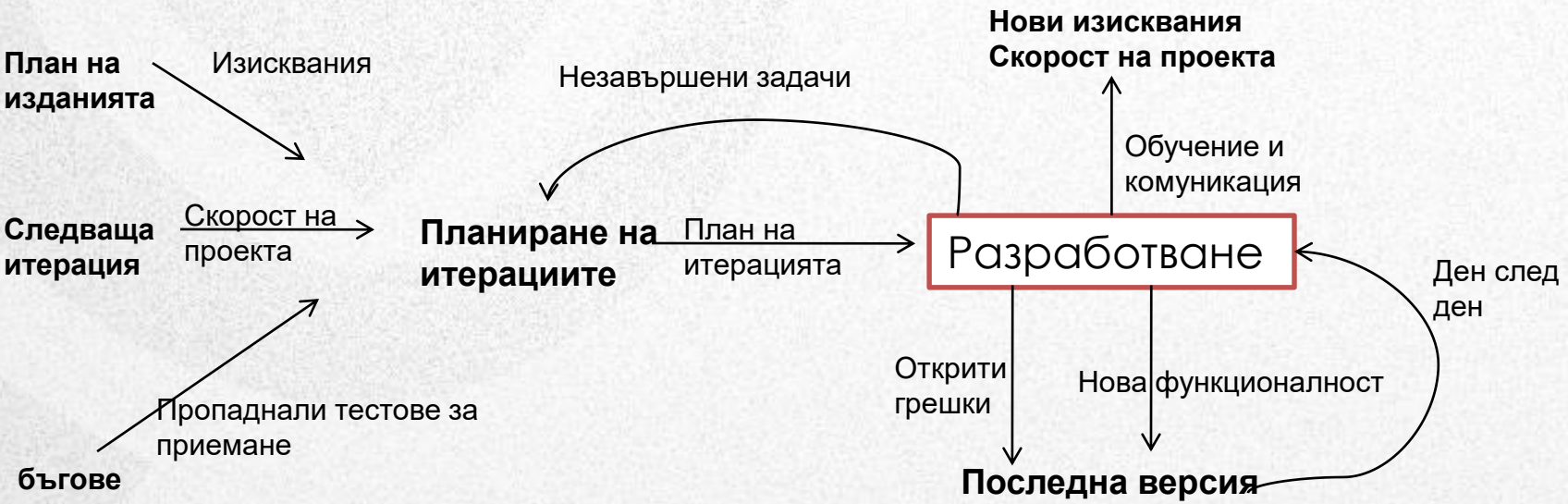
ЦЕЛИ НА XP

- да се справи с проблема на постоянно изменящите се изисквания – процеса трябва да е адаптивен;
- да е ориентиран към потребителите – те стават основни участници в процеса;
- да се доставя софтуера точно когато се изисква от клиентите;

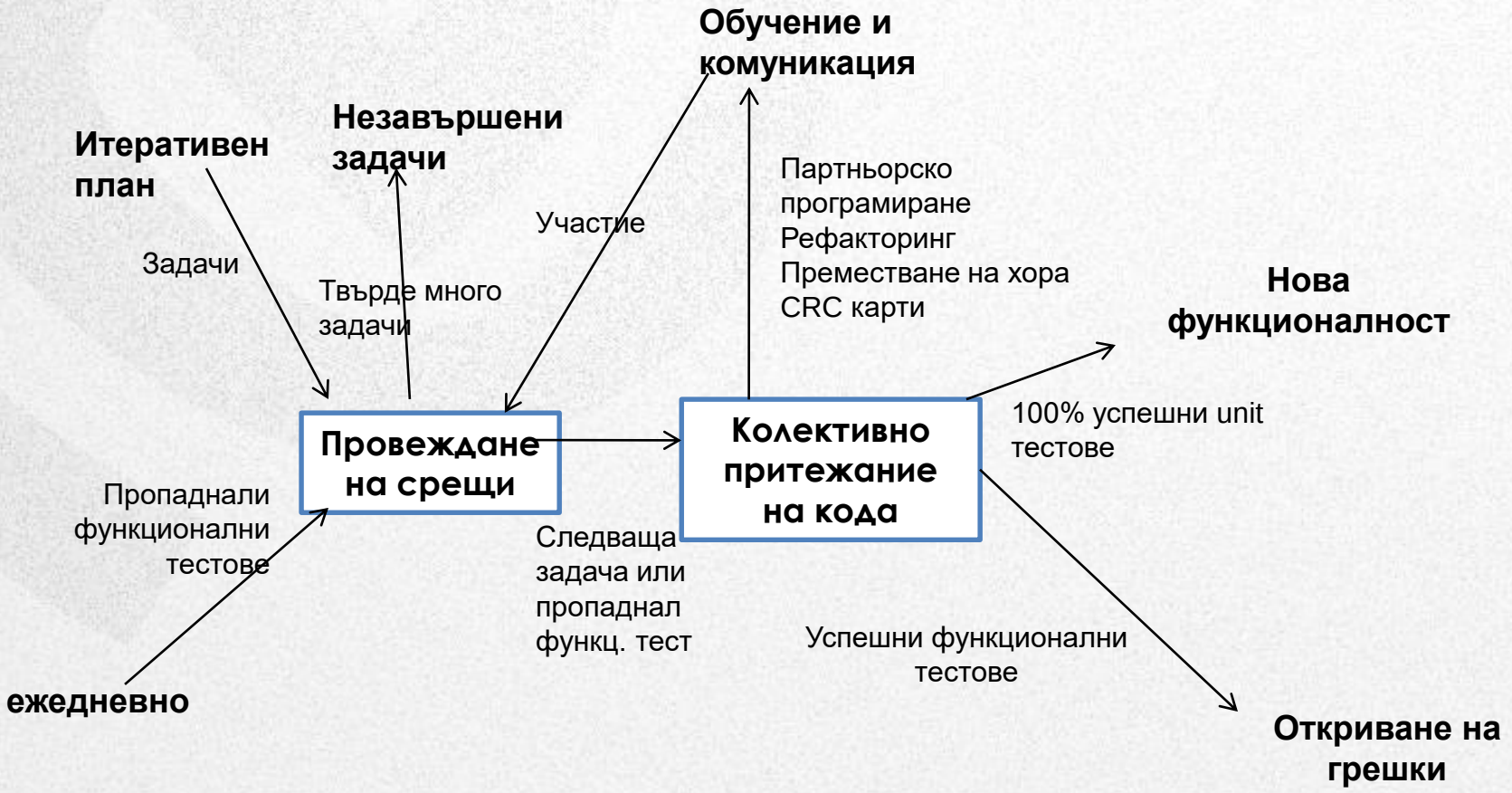
ПРОЦЕС НА РАЗРАБОТВАНЕ



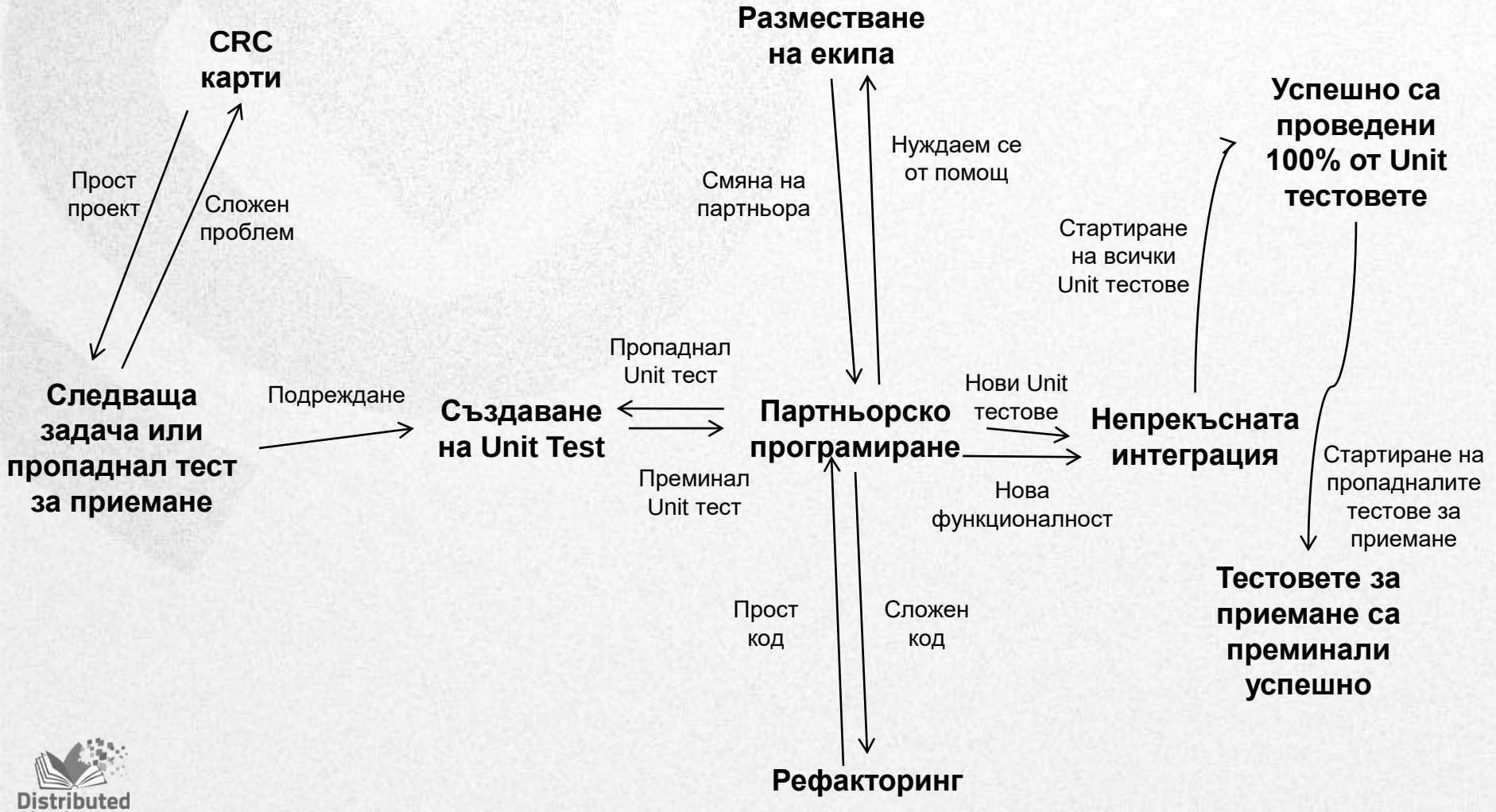
ИТЕРАЦИЯ



РАЗРАБОТВАНЕ



ПАРТНЬОРСКО ПРОГРАМИРАНЕ



Distributed @learning center

СПОРЕН МОМЕНТ.

- XP е базирано на еволюционно, а не на предварително проектиране!!!!!!
- Основните практики, които се използват, създават нова среда, в която еволюционното проектиране изглежда напълно убедително – непрекъсната интеграция, тестване и refactoring.
- Трябва да се отбележи обаче, че предварителното проектиране също намира място в процеса на XP - времето, което предшества реализацията на конкретни задачи по време на итерация

ОСНОВНИ ПРАКТИКИ НА XP

- *Цялостен екип*
- *Игрово планиране*
- *Малки издания*
- *Клиентски тестове*
- *Прост проект*
- *Партньорско програмиране*
- *Разработка чрез тестване*
- *Рефакторинг*
- *Постоянна интеграция*
- *Колективно притежаване на кода*
- *Стандарт в кодирането*
- *Системен речник*
- *Постоянно темпо*

ЦЯЛОСТЕН ЕКИП

- По принцип с XP не се създават продукти за много хора - създават се за даден конкретен клиент;
- екипа трябва да включва бизнес представител, наречен "Клиентът", който осигурява изискванията, поставя приоритетите и направлява проекта – всички етапи от проекта изискват участието на клиента;
- програмисти;
- тестващи;
- анализатори;

ИГРОВО ПЛАНИРАНЕ

- Управлението на проекта е важен момент в XP. Има две ключови стъпки при планиране в XP:
  - *Планиране на изданията* – извършва се от клиента след оценка на сложността на изискванията от разработчиците.
  - Итеративно планиране - практика, при която на екипа се дава насочване всеки две седмици. XP екипите изграждат софтуера в двуседмични итерации, предавайки работещ и успешен софтуер, в края на всяка итерация.

ИГРОВО ПЛАНИРАНЕ

- XP е една от малкото методологии, които признават, че като се пристъпва към проекта, не може да се знае всичко. По време на развитие на проекта се учат както разработчици, така и клиенти - ефективна ще бъде само методология, която признава и поощрява такива изменения.

- Това става възможно благодарение на **игрови планирания** - концепции, въведени от Kent Beck - да се посочи примерен план и да се коригира по време на изясняване на ситуациите.
  - Атрибутите на тези планирания са карти с указания (на всяка карта се намира задание на клиента), които ръководят итерационния процес на разработка на проекта, а също така и примерния план за следващия пуск (или два), както е описано в публикацията *Planning Extreme Programming* [Beck&Fowler]. Приложимостта на такъв стил на планиране се определя от това, какви са правилно взетите бизнес решения от клиента и правилно взетите технически решения от групата на разработчиците. Без всичко това процеса не е хармоничен.

ИГРОВО ПЛАНИРАНЕ

- Разработчиците определят:
  - Приблизителен срок за разработка на заданията;
  - Разходите при използване на различните технически възможности;
  - Структурата на екипа от разработчици;
  - Риска при изпълнение на всички задания;
  - Реда на разработване на заданията по итерации (ако първо се изпълнят задачи с голяма степен на риск, риска може да се намали);

ИГРОВО ПЛАНИРАНЕ

- Клиентите определят:
  - Размер (задания за цялостния проект и задания за всяка итерация);
  - Дата на изданията;
  - Приоритети (кои функции трябва да бъдат разработени в първата опашка от чакащи, в зависимост от тяхното значение за бизнеса);
- Планирането се прави постоянно – като всеки в екипа може да внесе в плана корекции.

КЛИЕНТСКИ ТЕСТОВЕ

- Клиентите дефинират *функционални тестове*, които с помощта на разработчиците се автоматизират.
- *Използвани ресурси*. Изразходването на памет може съществено да повлияе на фактическата работоспособност на функционално безупречната система. Поради това е необходимо да се тестват използваните ресурси.
- *Стабилността* е предпоставка за изискването: "системата не трябва да се разпадне при работа".

КЛИЕНТСКИ ТЕСТОВЕ (…)

- Функционалните тестове се създават на базата на потребителските изисквания, избрани по време на събиранията за повторно планиране. Клиентът определя сценарии за тестване. Дадено изискване може да има един или много функционални теста. Всеки функционален тест представя някакъв очакван резултат от системата. Клиентите са отговорни за потвърждаване на коректността на функционалните тестове и преглеждайки резултатите, решават кои от неуспешните тестове са с най-висок приоритет.
- Множеството функционални тестове не гарантират напълно работоспособността на системата. Зад рамката остават поведението на системата при натоварване, използваните ресурси, стабилността и не-издържливостта.
- Функционални тестове се изпълняват многократно в XP проект, обикновено ежедневно или по-често, и разбира се всяка седмица. Тестовете дават представа за това, колко от желаната функционалност ще работи. Тъй като те се изпълняват много често, най-добре е да бъдат автоматизирани.

КЛИЕНТСКИ ТЕСТОВЕ

- Съществуват няколко механизма, които могат да бъдат използвани за изпълнение на функционални тестове:
  - Ръчен тест
  - GUI тестващи инструменти
  - Код
  - Скрипт
  - Електронна таблица
  - Шаблон

МАЛКИ ИЗДАНИЯ

- Екипът създава ново издание на всяка итерация
- Клиентите могат да ползват тези издания за всякакви цели – за оценка или предоставяне на крайните потребители.

ПРОСТ ПРОЕКТ

- Разработчиците в XP изграждат софтуера чрез прост проект – проекта се поддържа адекватен на текущата функционалност на системата;

- Проектирането в XP не е нито предишно нито предстоящо нещо – проектирането продължава по време на цялата разработка (пусково и итерационно планиране)

ПРОСТ ПРОЕКТ

- Критиците на XP заявяват, че този вид програмиране, отхвърля проектирането. Това не е вярно. Работата е в това, че обширните подходи се опитват да обхванат всичко, с изключение на най-фундаменталните задачи. При екстремното програмиране се отделя достатъчно много внимание на проекта, така че неговата разработка се осъществява постоянно. Винаги, в някакъв момент от време, се използва най-простият проект, който гарантира изпълнимост, и внася изменения в процеса на работа, така че да отразява реалната ситуация.

ПРОСТ ПРОЕКТ

- Най-простият проект е този, който:
  - осигурява преминаване на всички тестове;
  - не съдържа дублиране на код;
  - ясно показва намеренията на програмистите за целия код;
  - предполага използване на колкото е възможно по-малък брой класове и методи;

ПРОСТ ПРОЕКТ

- В XP са много популярни два лозунга: "Do the Simplest Thing that Could Possibly Work" ("Търсете най-простото решение, което може да разработите") и YAGNI ("You Aren't Going to Need It" - "Това не ви е необходимо"). И двата се олицетворяват като една от практиките на XP, под названието "прост проект".
- Използва се refactoring, за да се поддържа проекта прост.

ПРОСТ ПРОЕКТ (…)

- По принципа YAGNI, не е необходимо да се пише код днес, ако той е нужен за такива свойства на програмата, които ще се реализират само утре. На пръв поглед в това няма нищо сложно. Сложността започва, когато се заговори за неща като програмни рамки за създаване на приложения, компоненти за повторно използване и гъвкав проект. Трябва да се каже, че проектирането им е доста сложно. XP не препоръчва създаване на гъвкави компоненти и рамки преди да е необходима точно някаква функционалност. По-добре е тези структури да бъдат нарастващи в зависимост от изискванията. Например, ако днес е нужен клас Money, който обработва събиране, а не умножение, то днес ще се реализира в този клас само събиране, дори ако се знае със сигурност, че умножението ще потрябва в следващи итерации. Много просто и бързо се прави това сега, като се остави това за следващата итерация – когато в нея се появи реална необходимост.
- Такова поведение е оправдано от икономическа гледна точка. Занимавайки се с работа, която е необходима само утре, се изразходват най-много усилия и време, предназначени за задачи, които е трябвало да бъдат свършени днес. Плана на изданието ясно показва, над какво е нужно да се работи в настоящия момент. Когато се работи над това, което ще трябва в бъдеще, се нарушава споразумението с клиента. Появява се риск да не се изпълни всичко, записано в изискванията за текущата итерация. Когато рано се използват в работата над проекта грешни решения, тогава е по-лошо. Привържениците на методологията XP считат, че в такава ситуация много лесно се взема неправилно решение.

ПАРТНЬОРСКО ПРОГРАМИРАНЕ

- Целият реализиран софтуер в XP е изграден от двама програмисти – гарантира се по-добър проект, по-добро тестване и по-добър код.
- Преимущества:
  - Най-малко двама се запознават с всеки компонент на системата.
  - Вероятността, при която двама могат да решат да не проведат тестване или пропуснат някой или друг етап, е много по-малка.
  - При смяна на партньор настъпва разпространение на информация сред членовете от екипа разработчици.
  - Кодът винаги се проверява, дори и от един човек.

ПАРТНЬОРСКО ПРОГРАМИРАНЕ (…)

- Pair Programming - двама програмисти, който работят заедно на един комютър, driver и navigator. Докато driver-а пише на компютъра, navigator-а следи работата му. И е добре на половин час да си разменят ролите, а всеки ден да се сменят партньорите. Предимствата на pair programming-а са че по този начин се пише по-верен код, правят се по-малко логически грешки; разменят се знания, защото колкото и да знае даден човек никога не може да знае всичко и винаги може да научи повече; така се сближават хората от екипа, нещо много важно за XP. Ако хората се сменят по-често повече от тях ще бъдат въведени в различните специфики и по този начин всеки ще е много по-добре запознат с цялостния продукт и комуникацията ще е по-лесна. Смята се, че по този начин има по-малко прекъсвания на работата, което води до по-голяма продуктивност.

ПАРТНЬОРСКО ПРОГРАМИРАНЕ

- Проблеми:
  - Неспособен служител;
  - Безрезултатен спор;
  - Пасивност;
  - План на работното помещение;
  - Клавиатурата;
  - Стандарти в кодирането;
  - Закъснения;
  - Отговорност;
  - Екип.

РАЗРАБОТКА ЧРЕЗ ТЕСТВАНЕ

- Unit тестове – създават се по време на писане на кода от разработчиците;
- Функционални тестове – създават се по време на събиране на изискванията от клиента.
- Тестовете се създават преди писане на кода.
- Тестовете се третират като среда, в която преминава само правилен код.

REFACTORING

- В XP се използва процес на постоянно подобряване на проекта наречен refactoring.

- Процеса на рефакторинг е съсредоточен върху премахване на дублирането и увеличаване на съгласуваността на кода. Високата съгласуваност и ниската свързаност, е критерий за добре проектиран код.

ПОСТОЯННА ИНТЕГРАЦИЯ

- Системата се запазва напълно интегрирана през цялото време.
- Масова интеграция – не се препоръчва
- Честа интеграция – лесно откриване на проблемите

КОЛЕКТИВНО ПРИТЕЖАВАНЕ НА КОДА

- Всеки член на екипа, трябва да има право, да внася подобрения в програмния код. Притежатели на кода се явяват всички, така възниква отговорността да се поддържа от всеки.

СТАНДАРТ В КОДИРАНЕТО

- XP екипите следват общ стандарт на кодиране, така че целият код в системата изглежда като написан от един, много способен човек;

- Това, което би било невъзможно да се определи е, кой точно от разработчиците е написал този или онзи участък от кода.

СИСТЕМЕН РЕЧНИК

- Системен речник в XP е аналог на това, което в повечето методологии се нарича архитектура. Това дава на екипа от разработчици последователна картина, която описва, как работи съществуващата система, къде трябва да се намират новите компоненти и в какъв вид те трябва да бъдат представени.

ПОСТОЯННО ТЕМПО

- Извънредната работа поглъща ума и мотивацията на екипа. Вместо това е по-добре да се използват срещи за планиране на изданията, за да се промени обхвата на проекта или да се определи подходящото темпо. Не е добра идеята за увеличаване на ресурсите с добавяне на повече хора, когато се закъснява с реализацията на проекта.

ДОКУМЕНТАЦИЯ В XP

- Документация на изискванията
- Документация на проекта
- Друга документация

ДОКУМЕНТАЦИЯ НА ИЗИСКВАНИЯТА

- CRC карти;
- Основното е, че изискванията се документират в много по-пълна и информативна форма, която не е писмена. Те имат форма на автоматизирани тестове, с които се контролират резултатите в използвания програмен продукт.

ДОКУМЕНТИРАНЕ ПРОЕКТА НА СИСТЕМАТА

- CRC карти;
- UML диаграми – разработени формално;
- При приключване на проекта програмистите трябва да напишат кратък документ, в който да бъде описана архитектурата на системата или нейният системен речник (възможно е с няколко UML-диаграми), основни понятия, най-важните класове, и т.н.

ДОКУМЕНТИРАНЕ НА ПРОГРАМНИЯ КОД

- Самия код трябва да говори за себе си;
- Използване на коментари там където е необходимо;
- Unit тестовете също представляват документация – те отразяват това, което кода прави за съответния случай.
- При необходимост се разработва и друга документация.

ЛИТЕРАТУРА

- Kent Beck, "Embracing Change with Extreme Programming", *eXtreme Programming Pros and Cons: What Questions Remain?, IEEE Computer Society Dynabook* (September 2001), http://www.computer.org/SEweb/Dynabook/WhatIs.htm
- *Kent Beck, Martin Fowler, "Planning Extreme Programming", (Addison – Wesley, 2000)*
- http://www.extremeprogramming.org
- http://www.xprogramming.com
- http://www.agile.com