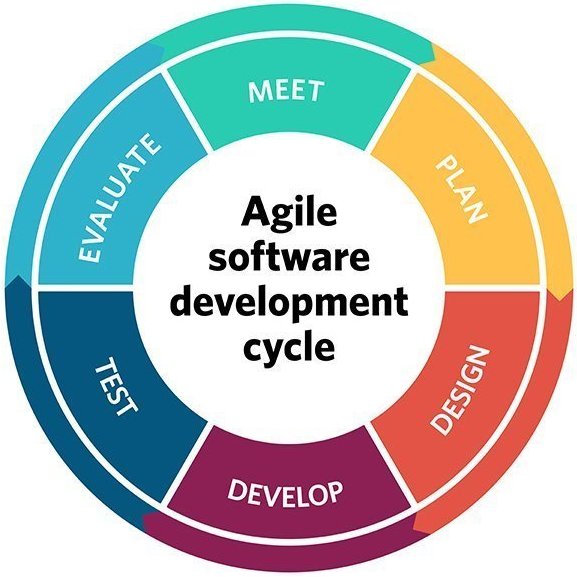
ФОН УНИВЕРЗИТЕТ СКОПЈЕ

ФАКУЛТЕТ ЗА ИНФОРМАТИКА



|  |
| --- |
| Професор: Студент:  Проф. д-р Оливер Илиев Марио Илиќ- 13647 |

|  |
| --- |
| СЕМИНАРСКА РАБОТА ПО ПРЕДМЕТОТ: |
| ОСНОВИ НА СОФТВЕРСКО ИНЖЕЊЕРСТВО |
| **ТЕМА: AGILE SOFTWARE DEVOLPMENT** |



СОДРЖИНА

[ ВОВЕД](#_Toc244266819) 3

[ 1. ИСТОРИЈА НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ](#_Toc244266821) 4

[ 2. МАНИФЕСТ ЗА АГИЛЕН РАЗВОЈ НА СОФТВЕР 5](#_Toc244266821)

[ 2.1. Агилни вредности за развој на софтвер 5](#_Toc244266823)

[ 2.2. Агилни принципи за развој на софтвер 6](#_Toc244266823)

[ 3. ПРЕГЛЕД НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ](#_Toc244266821) 7

[ 3.1. Итеративен, постепен и еволутивен преглед 7](#_Toc244266823)

[ 3.2. Ефикасна комуникација и лице-в-лице 8](#_Toc244266823)

[ 4. ФИЛОЗОФИЈА НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ](#_Toc244266821) 8

[ 4.1. Адаптивно наспроти предвидливо 9](#_Toc244266823)

[ 4.2. Агилен против водопад 10](#_Toc244266823)

[ 4.3. Код наспроти документација 11](#_Toc244266823)

[ 5. АГИЛНИ МЕТОТИ ЗА РАЗВОЈ НА СОФТВЕР](#_Toc244266821) 12

[ 5.1. Агилни практики за развој на софтвер 13](#_Toc244266823)

[ 5.2. Метод за прилагодување 14](#_Toc244266823)

[ 5.3. Големи размери, префрлани и дистрибуирани 15](#_Toc244266823)

[ 5.4. Регулирани домени 16](#_Toc244266823)

[ 6. ИСКУСТВО И УСВОЈУВАЊЕ НА АГИЛНИОТ СОФТВЕР](#_Toc244266821) 17

[ 6.1. Агилен против водопадот 17](#_Toc244266823)

[ 6.2. Заеднички агилни развојни проблеми во софтверот 18](#_Toc244266823)

[ 7. АГИЛНО УПРАВУВАЊЕ](#_Toc244266821) 20

[ ЗАКЛУЧОК](#_Toc244266823) 21

# ВОВЕД

Со оглед на стапката на технолошките промени и нејзиниот секојдневен пораст, економските пресврти, комплексноста на софтверите кои се бараат а која станува се поголема, барањата се значително поразлични, а главен стремеж е да се задоволат моменталните потреби на пазарот.

Притоа конвенционалниот проектен менаџмент значително заостануваше со одговорите зад ваквите варијации. Се разбира секоја компанија би сакала таа ризична експедиција да ја заврши со достава на одличен софтвер, софтвер кој до детали ќе ги исполни очекувањата на клиентите и сите заинтересирани страни во тие рамки, а се разбира да успее и соодветно да одговори кон таквите економски варијации задоволувајќи го пазарот во онаа мера во која и е очекувано, а сето тоа во определениот временски период и со определените средства.

Агилниот развој на софтвер е приод кон развојот на софтверот според кој барањата и решенијата се развиваат преку заеднички напори на самоорганизирачките и крос-функционалните тимови и нивните клиенти / крајните корисници . Се залага за адаптивно планирање, еволутивен развој, емпириско знаење и континуирано подобрување и го поттикнува брз и флексибилен одговор на промените.

Терминот агилен (понекогаш напишан Agile ) беше популаризиран, во овој контекст, од страна на Манифестот за агилен развој на софтвер. Вредностите и принципите застапени во овој манифест беа изведени и поткрепени со широк спектар на рамки за развој на софтвер , вклучувајќи Scrum и Kanban. Има значителни анегдотски докази дека усвојувањето агилни практики и вредности ја подобрува агилноста на софтверските професионалци, тимови и организации; сепак, некои емпириски студии не нашле научни докази.

# 1.ИСТОРИЈА НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ

Итеративен и делумното развој методи може да се датира уште од 1957 година, со еволутивното управување со проекти и адаптивни развој на софтвер се појавуваат во раните 1970-ти.

Во текот на 1990-тите, голем број на лесни методи за развој на софтвер се развиле како реакција на преовладувачките методи на тешка категорија , кои критичарите ги опишале како премногу регулирани, планирани и микро-управувани . Тие вклучуваат: брза апликација развој (РАД), од 1991 година; на унифициран процес и метод за развој на динамички системи (DSDM), и од 1994 година; Scrum , од 1995 година; Кристално јасно и екстремно програмирање (XP), и од 1996 година; и функционално развиен развој , од 1997 година. Иако сите овие потекнуваат пред објавувањето на Агилниот манифест, сега тие се колективно познати како агилни методи за развој на софтвер. Во исто време, слични промени беа во тек и во производството и во воздушниот простор.

Во 2001 година седумнаесет развивачи на софтвер се сретнаа во одморалиште во Snowbird , Јута, на разговори околу овие лесни развојни методи, меѓу кои и кај Кент Бек , Вард Канингам , Дејв Томас , Џеф Сатерленд , Кен Швабер , Џим Хајсмит , Алистер Кокберн и Боб Мартин . Заедно го објавија Манифестот за агилен развој на софтвер.

Во 2005 година, група предводена од Cockburn и Highsmith напиша дополнување на принципите за управување со проекти , PM Декларацијата за меѓузависност, за насочување на управување со софтверски проекти според агилни методи за развој на софтвер.

Во 2009 година, група која работеше со Мартин напиша проширување на принципите за развој на софтверот, Манифестот за изработка на софтверот , за насочување на агилниот развој на софтвер според професионално однесување и владеење.

Во 2011 година, Агилниот Алијанс го создаде Водичот за ефикасни практики (преименуван во Агилен Речник 2016), развивачки компендиум на отворени извори на работни дефиниции на агилни практики, термини и елементи, заедно со толкувања и упатства за искуство од светската заедница на агилни практичари.

# 2. МАНИФЕСТ ЗА АГИЛЕН РАЗВОЈ НА СОФТВЕР

Манифестот за агилен развој на софтвер идентификува една мрежа интерфејс. Интернет протоколот обезбедува пакети испратени до адреса на unicast до тој специфичен интерфејс.

## 2.1. Агилни вредности за развој на софтвер

Врз основа на нивното комбинирано искуство за развој на софтвер и помагање на другите да го сторат тоа, седумнаесетте потписници на манифестот изјавија дека тие ги ценат:

* Поединци и интеракции околу процесите и алатките
* Работен софтвер за сеопфатна документација
* Соработка на клиентите околу преговорите за договор
* Одговарајќи на промени во текот на следење на планот

Како што објаснува Скот Амблер :

* Алатките и процесите се важни, но поважно е да се имаат компетентни луѓе ефикасно да работат заедно.
* Добрата документација е корисна за да им помогне на луѓето да разберат како е изграден софтверот и како да го користат, но главната точка на развој е да креира софтвер, а не документација.
* Договорот е важен, но не е замена за тесна соработка со клиентите за да открие што им е потребно.
* План за проекти е важен, но не смее да биде премногу ригиден за да се прилагодат на промените во технологијата или животната средина, приоритетите на засегнатите страни и разбирањето на проблемот и неговото решавање на луѓето.

## 2.2. Агилни принципи за развој на софтвер

На Манифест за Агилниот развој на софтвер се базира на дванаесетте принципи:

1. Задоволството на клиентите со рана и континуирана испорака на вреден софтвер.
2. Добредојдовте во променливите барања, дури и во доцниот развој.
3. Најчесто давајте софтвер за работа (недели, а не месеци)
4. Секојдневна соработка меѓу деловните луѓе и програмери
5. Проектите се изградени околу мотивирани индивидуи, на кои им треба доверба
6. Лице-в-лице разговор е најдобра форма на комуникација (колокација)
7. Работниот софтвер е примарна мерка за напредок
8. Одржлив развој, способен да одржува постојано темпо
9. Континуирано внимание на техничка извонредност и добар дизајн
10. Едноставност - уметноста на максимизирање на износот на работа што не е направено - е од суштинско значење
11. Најдобрите архитектури, барања и дизајни произлегуваат од тимови за самоорганизирање
12. Редовно, тимот се одразува како да станат поефективни и соодветно се прилагодува

# 3. ПРЕГЛЕД НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ

## 3.1. Итеративен, постепен и еволутивен преглед

Повеќето агилни методи за развој ги разбиваат деловите за развој на производи во мали чекори што го минимизираат износот на планирање и дизајн однапред. Повторувања, или спринтови, се кратки временски рамки кои обично траат од една до четири недели. Секоја итерација вклучува крос-функционални тим кои работат во сите функции: планирање , анализа , дизајн , кодирање , единица за тестирање и прифаќање тестирање . На крајот од итерацијата се прикажува работен производ на засегнатите страни. Ова го минимизира севкупниот ризик и овозможува производот брзо да се прилагодува на промените. Итерацијата не може да додаде доволно функционалност за да гарантира ослободување на пазарот, но целта е да се има достапно ослободување на крајот од секоја повторување. Може да бидат потребни повеќе повторувања за да се ослободи производ или нови функции. Работниот софтвер е примарна мерка за напредок.

* **Многу краток циклус на повратни информации и циклус адаптација**:

Заедничка карактеристика во агилниот развој на софтверот е секојдневниот стенд-ап. На кратка седница, членовите на тимот пријавуваат едни со други што направиле претходниот ден во однос на итеративниот гол на нивниот тим, што имаат намера да направат денес кон целта, како и сите блокади или препреки што можат да ги видат до целта.

* **Квалитетен фокус**:

Специфични алатки и техники, како што се континуирана интеграција , автоматско тестирање на единици , програмирање на парови , тест-ориентиран развој , шеми на дизајн , однесување управувано развој , домен-управувано дизајн , код рефакторинг и други техники често се користат за подобрување на квалитетот и подобрување на развој на производи агилност. Ова е засновано на дизајнирање и градење на квалитет од почеток и можност да демонстрира софтвер за клиентите во секој момент, или барем на крајот од секоја повторување.

## 3.2. Ефикасна комуникација и лице-в-лице

Заедничка карактеристика во агилниот развој на софтверот е секојдневниот стенд-ап. Принципот на колокација е дека соработниците на истиот тим треба да се сместат заедно за подобро да го утврдат идентитетот како тим и да ја подобрат комуникацијата. Ова овозможува интеракција лице-в-лице , идеално пред таблата, што го намалува времетраењето на циклусот кое обично се прави кога прашањата и одговорите се посредувани преку телефон, постојан разговор, вики или е-пошта. Без оглед на тоа кој метод за развој се следи, секој тим треба да вклучи претставник на клиент ("Сопственик на производот" во Scrum ). Ова лице е договорено од засегнатите страни да дејствуваат во нивно име и се обврзува да бидат достапни за програмерите за да одговорат на прашања во текот на итерацијата. На крајот од секоја итерација, заинтересираните страни и претставникот на клиентот го разгледуваат напредокот и ги преиспитуваат приоритетите со цел да се оптимизира враќањето на инвестицијата (ROI) и да се обезбеди усогласување со потребите на клиентите и целите на компанијата. Во агилниот развој на софтвер, информациски радијатор е (нормално голем) физички екран кој се наоѓа на видно место во близина на развојниот тим, каде што пронаоѓачите можат да го видат. Тој претставува ажуриран преглед на развојот на производот. Еден индикатор изгради светлина исто така може да се искористи за информирање на тимот за состојбата на нивниот развој на производот.

# 4. ФИЛОЗОФИЈА НА АГИЛНИОТ РАЗВОЈ НА СОФТВЕРОТ

Во споредба со традиционалното софтверско инженерство, агилниот развој на софтвер главно насочува комплексни системи и развој на производи со динамички, недетерминистички и нелинеарни карактеристики. Точни проценки, стабилни планови и предвидувања често е тешко да се добијат во раните фази, а довербата во нив веројатно е ниска. Агилни лекари ќе се обидат да го намалат скокот на верата што е потребен пред да може да се добие некој доказ за вредност. Големите напред-спецификации, веројатно, би предизвикале многу отпад во такви случаи, т.е. не се економски економични. Овие основни аргументи и претходни искуства во индустријата, научени од годините на успеси и неуспеси, помогнаа во формирањето на агилниот развој на адаптивниот, итеративен и еволутивен развој.

## 4.1. Адаптивно наспроти предвидливо

Развојните методи постојат на континуум од адаптивни кон предвидливо. Агилни методи за развој на софтвер лежат на адаптивната страна на овој континуум. Еден од клучните методи за прилагодување на развојот е приодот на тркалање на бранот за планирање на распоредот, кој ги идентификува пресвртниците, но остава флексибилност на патот за нивно достигнување, а исто така им овозможува на самите пресвртници да се променат.

Адаптивните методи се фокусираат на брзо прилагодување на променливите реалности. Кога се менуваат потребите на проектот, се менува и адаптивниот тим. Адаптивниот тим има потешкотии да опише што точно ќе се случи во иднина. Колку е далеку подалеку датумот, толку попрецизен е адаптивниот метод за тоа што ќе се случи на тој датум. Прилагодливиот тим не може точно да објави кои задачи ќе ги извршат следната недела, туку само кои карактеристики планираат за следниот месец. Кога беше запрашан за ослободување шест месеци од сега, адаптивниот тим може да биде во можност да го пријави само извештајот за мисијата за ослободување или изјава за очекувана вредност наспроти трошоците.

Предвидливи методи, пак, се фокусираат на детално анализирање и планирање на иднината и обезбедување на познати ризици. Во крајности, предвидливото тим може точно да објави кои карактеристики и задачи се планираат за целата должина на развојниот процес. Предвидливите методи се потпираат на ефективна анализа на раната фаза и ако ова оди многу лошо, проектот може да има потешкотии во менувањето на правецот. Предвидните тимови често воведуваат контрола одбор за промена за да се осигурат дека ги земаат предвид само највредните промени.

Анализата на ризикот може да се искористи за да се избере помеѓу адаптивни ( агилни или вредносни ) и предвидувачки ( плански управувани ) методи. Бери Бохм и Ричард Тарнер сугерираат дека секоја страна од континуумот има своја домашна основа

## 4.2. Агилен против водопад

Една од разликите помеѓу агилните методи на развој на софтвер и водопадот е пристапот кон квалитетот и тестирањето. Во моделот водопад , секогаш постои посебна фаза на тестирање по фазата на изградба ; сепак, во агилниот развој на софтверското тестирање е завршено во истата итерација како програмирање.

Бидејќи тестирањето е направено во секоја итерација - која развива мало парче софтверски корисници, често може да ги користат тие нови делови на софтвер и да ја проверат вредноста. Откако корисниците ќе ја знаат вистинската вредност на обновениот дел од софтверот, тие можат да донесуваат подобри одлуки за иднината на софтверот. Имајќи вредност ретроспектива и софтвер за ре-планирање сесија во секоја итерација - Scrum обично има повторувања од само две недели - им помага на тимот постојано да се прилагодат своите планови, така што да се зголеми вредноста што ја обезбедува. Ова следи по шема слична на циклусот PDCA , бидејќи работата е планирана , направена , проверена (во рецензијата и ретроспективноста), и сите договори кои се договорени се постапиле.

* Овој итеративен пристап поддржува производ, а не начин на размислување на проектот . Ова обезбедува поголема флексибилност во целиот процес на развој; додека на проектите барањата се дефинирани и заклучени од самиот почеток, поради што е тешко да се сменат подоцна.
* Итеративниот развој на производот му овозможува на софтверот да се развива како одговор на промените во деловното опкружување или барањата на пазарот.

Поради краткиот итеративен стил на агилниот развој на софтвер, тој исто така има силни врски со концептот за посно стартување.

## 4.3. Код наспроти документација

Во писмото до IEEE Computer , Стивен Ракитин изрази цинизам за агилниот развој на софтвер, нарекувајќи го "уште еден обид за поткопување на дисциплината на софтверското инженерство" и преведување на "работниот софтвер преку сеопфатна документација" како "сакаме да го поминеме нашето време на кодирање. Запомнете, вистинските програмери не пишуваат документација. "

Ова е оспорено од поддржувачите на агилниот развој на софтвер, кој тврди дека програмерите треба да напишат документација, ако тоа е најдобриот начин да се постигнат релевантните цели, но дека често пати има подобри начини за постигнување на тие цели отколку пишување на статичка документација. Скот Амблер тврди дека документацијата треба да биде "едвај доволно добра" (JBGE), дека премногу или сеопфатна документација најчесто ќе предизвика отпад, а програмерите ретко имаат доверба во детална документација, бидејќи тоа обично не е синхронизирано со кодот, додека премалата документација може исто така да предизвика проблеми за одржување, комуникација, учење и споделување на знаења.

**Алистер Кокберн напишал за методот Кристално јасен :**

“Кристал размислува за развој на серија кооперативни игри и има намера документацијата да биде доволна за да му помогне на следната победа во следната игра.

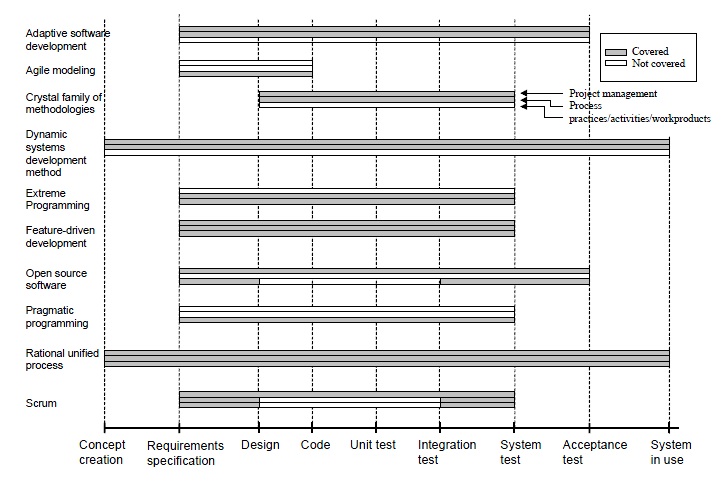
Работните производи за Кристал вклучуваат случаи на употреба, листа на ризици, план за итерација, основни модели на домени и белешки за дизајн за да се информираат за изборите ... но не постојат шаблони за овие документи и описите се неопходни нејасни, но целта е јасна, само доволно документација за следната игра.

Јас секогаш се трудам да го карактеризирам ова со мојот тим како: што би сакале да знаете дали ќе се приклучите на тимот утре“.

# 5. АГИЛНИ МЕТОТИ ЗА РАЗВОЈ НА СОФТВЕР

Агилни методи за развој на софтвер поддржуваат широк опсег на животниот циклус на развој на софтвер. Некои се фокусираат на практиките (на пример, XP, прагматично програмирање, агилно моделирање), додека некои се фокусираат на управување со протокот на работа (на пример, Scrum, Kanban).

Некои активности за поддршка за спецификација на барања и развој (на пример, FDD), додека некои се обидуваат да го покријат целиот животен циклус на развојот (на пример, DSDM, RUP ).



1. Поддршка на животниот циклус за развој на софтвер

**Популарни агилни рамки за развој на софтвер вклучуваат:**

* + Адаптивен развој на софтвер (ASD)
  + Агилно моделирање
  + Агилниот унифициран процес (AUP)
  + Дисциплинирана агилна испорака
  + Метод на развој на динамички системи (DSDM)
  + Екстремно програмирање (XP)
  + Функционален развој (FDD)
  + Посно развој на софтвер
  + Kanban
  + Рапид развој на апликации (RAD)
  + Scrum
  + Scrumban

## 5.1. Агилни практики за развој на софтвер

Агилниот развој на софтвер е поддржан од голем број конкретни практики, покривајќи области како што се барања, дизајн, моделирање, кодирање, тестирање, планирање, управување со ризик, процес, квалитет, итн. Некои значајни агилни практики за развој на софтвер вклучуваат:

* Прифаќање на тест-управувано развој (ATDD)
* Агилно моделирање
* Агилно тестирање
* Заостанати (производ и Спринт)
* Развој управуван од однесување (BDD)
* Метод на бизнис аналитичар (BADM) [44]
* Континуирана интеграција (CI)
* Крос-функционален тим
* Домен-ориентиран дизајн (DDD)
* Информациски радијатори (табла, табла, визуелен управувачки одбор, табела за запалување )
* Итеративен и поединечен развој (IID)
* Ниско-код развојни платформи
* Програмирање на пар
* Планирање на покер
* Рефакторинг
* Ретроспектива
* Scrum настани (спринт планирање, секојдневно scrum, спринт преглед и ретроспектива)
* Моделирање управувано од приказни
* Тест-управувано развој (TDD)
* Timeboxing
* Корисничка приказна
* Мапирање на корисни приказни
* Следење на брзина

## 5.2. Метод за прилагодување

Во литературата, поинакви термини се однесуваат на поимот адаптација на методот, вклучувајќи го и методот за прилагодување, методот за фрагмент прилагодување и методот на ситуациона метода.

**Методот на прилагодување е дефиниран како:**

Процес или способност во која човечките агенти го одредуваат пристапот за развој на системот за специфична проектна ситуација преку реактивни промени и динамички интерплакции помеѓу контекстите, намерите и фрагментите на методот.

* Мехмет Нафиз Ајдин и др., Метод на развој на ефикасен информациски систем во употреба

Прифатливоста на ситуацијата треба да се смета како карактеристична карактеристика помеѓу агилни методи и повеќе планови за развој на софтверски методи, со агилни методи кои им овозможуваат на тимовите за развој на производи да ги адаптираат работните практики според потребите на одделни производи.

Потенцијално, повеќето агилни методи би можеле да бидат погодни за прилагодување на методот, како што е DSDM скроена во контекст на CMM и XP се прилагодени со техниката на пракса на практикување правила (RDP). Сепак, не се согласуваат сите агилни противници, но Швабер забележува дека "на прво место сме влегле во проблеми, мислејќи дека проблемот не е совршена методологија. Напорите [треба] да се фокусираат на промените што [се потребни] во претпријатието".

Бас Водде го зајакна ова гледиште, сугерирајќи дека за разлика од традиционалните, големи методологии кои бараат од вас да ги одбирате и избирате елементите, Scrum ги обезбедува основите на кои додавате дополнителни елементи за локализација и контекстуализирање на неговата употреба. Практичарите ретко ги користат методите за развој на системот, или агилни методи посебно, од книгата, честопати избирајќи да ги изостави или прилагодат некои од практиките на методот за да создадат метод во куќа.

Во пракса, методите може да се прилагодат со помош на разни алатки. Општите јазици за моделирање на процесот како што се Unified Modeling Language може да се користат за да се прилагодат на методите за развој на софтвер. Меѓутоа, постојат и посветени алатки за инженерство на методи, како што е Теоријата за суштина на софтверското инженерство на SEMAT .

## 5.3. Големи размери, префрлани и дистрибуирани

Агилниот развој на софтвер е нашироко се смета за многу погоден за одредени типови на средини, вклучувајќи ги и малите тимови на експерти кои работат на гринфилд проекти и предизвиците и ограничувањата со кои се соочуваат при усвојувањето на агилни методи за развој на софтвер во голема организација со наследствена инфраструктура е добро документирана и разбрана.

Како одговор на тоа, голем број стратегии и модели се развија за надминување на предизвиците со големи напори за развој (> 20 програмери) или дистрибуирани (не-колоцирани) тимови за развој, меѓу другите предизвици ; и сега постојат неколку признати рамки кои се обидуваат да ги ублажат или избегнат овие предизвици.

* Сместена агилна рамка (SAFe), Дин Лефингвел, меѓу другото
* Дисциплинирана агилна испорака (DAD), Скот Амблер, меѓу другото
* Големи размери (LeSS), Крег Ларман и Бас Водде
* Nexus (намалени професионални Scrum), Кен Швабер
* Scrum на скала, Џеф Сатерленд , Алекс Браун
* Enterprise Scrum, Мајк Бидл
* Setchu (лесна рамка базирана на Scrum), Michael Ebbage
* Xscale
* Агилна патека
* Холистички развој на софтвер

Постојат многу конфликтни гледишта за тоа дали сите овие се ефективни или навистина се вклопуваат во дефиницијата за агилен развој, и ова останува активна и тековна област на истражување.

Кога агилниот развој на софтвер се применува во дистрибуирана поставка (со тимови дисперзирани низ повеќе деловни локации), тоа најчесто се нарекува дистрибуиран агилен развој. Целта е да се искористат уникатните придобивки што ги нуди секој пристап. Дистрибуираниот развој им овозможува на организациите да изградат софтвер со стратешки поставување тимови во различни делови на светот, практично градење на софтвер круг-часовник (почесто се нарекува следен-на-сонце модел). Од друга страна, агилниот развој обезбедува зголемена транспарентност, постојан фидбек и поголема флексибилност при реагирање на промените.

## 5.4. Регулирани домени

Агилни методи за развој на софтвер најпрво се сметаа за најдобри за развој на некритични производи, а со тоа се исклучени од употреба во регулирани домени како што се медицински помагала, фармацевтски, финансиски, нуклеарни системи, автомобилската индустрија и авионските сектори, итн. Сепак, во последните неколку години, има неколку иницијативи за прилагодување на агилни методи за овие домени.

Постојат бројни стандарди кои можат да се применат во регулираните домени, вклучувајќи ISO 26262 , ISO 9000 , ISO 9001 и ISO / IEC 15504 . Голем број клучни загрижености се од особена важност во регулираните домени:

* Обезбедување на квалитет (ОК): Систематско и инхерентно управување со квалитетот засновано на контролиран професионален процес и сигурност и исправност на производот.
* Безбедност и сигурност: Формално планирање и управување со ризикот за да се ублажат безбедносните ризици за корисниците и безбедно да се заштитат корисниците од ненамерна и злонамерна злоупотреба.
* Следливост: Документација која обезбедува ревизорски докази за усогласеност со регулативата и олеснување на следливоста и испитување на проблемите.
* Верификација и валидација (V & V): вградени во текот на процесот на развој на софтвер (на пр. Спецификација на кориснички барања, функционална спецификација, спецификација на проект, преглед на кодот, единица тестови, интеграциски тестови, системски тестови).

# 6. ИСКУСТВО И УСВОЈУВАЊЕ НА АГИЛНИОТ СОФТВЕР

Иако агилни методи за развој на софтвер може да се користат со било која програмска парадигма или јазик во пракса, тие првично биле тесно поврзани со објектно-ориентирани средини како што се Smalltalk и Lisp и подоцна Јава. Првичните посвојувачи на агилни методи обично беа мали и средни тимови кои работеа на системи без преседан со барања кои беа тешки за финализирање и најверојатно да се променат додека системот се развиваше. Овој дел ги опишува заедничките проблеми со кои се соочуваат организациите кога се обидуваат да усвојат агилни методи за развој на софтвер, како и разни техники за мерење на квалитетот и ефикасноста на агилните тимови.

## 6.1. Мерење на агилност

Најдобрите агилни практичари секогаш нагласуваат темелни инженерски принципи. Како резултат на тоа, постојат голем број на најдобри практики и алатки за мерење на перформансите на агилниот развој на софтвер и тимови.

* **Внатрешни проценки**

Индексот за мерење на агилност , меѓу другото, ги оценува случувањата во однос на пет димензии на развој на производи (времетраење, ризик, новина, напор и интеракција). Други техники се базираат на мерливи цели и една студија сугерира дека брзината може да се користи како мерка на агилност. Исто така постојат и агилни самооценувања за да се утврди дали тимот користи агилни практики за развој на софтвер (тест на Nokia, тест Karlskrona, 42 поени).

* **Јавни анкети**

Една од првите истражувања за подобрување на квалитетот, продуктивноста и задоволството на бизнисот со користење на агилни методи за развој на софтвер беше истражување спроведено од Shine Technologies од ноември 2002 до јануари 2003 година.

Слична анкета, Државата на ефикасно , се спроведува секоја година, почнувајќи од 2006 година, со илјадници учесници од целата заедница за развој на софтвер. Ова ги следи трендовите за придобивките од агилноста, научените лекции и добрите практики. Секоја анкета објави зголемени броеви велејќи дека агилниот развој на софтвер им помага да испорачаат софтвер побрзо; ја подобрува нивната способност да управува со промените на клиентските приоритети; и ја зголемува нивната продуктивност. Истражувањата, исто така, постојано покажаа подобри резултати со агилни методи за развој на производ во споредба со класичното управување со проекти. Во рамнотежа, има извештаи дека некои сметаат дека агилните методи за развој се уште се премногу млади за да овозможат детално академско истражување за нивниот успех.

## 6.2. Заеднички агилни развојни проблеми во софтверот

Организации и тимови кои спроведуваат агилен развој на софтвер честопати се соочуваат со потешкотии при транзиција од повеќе традиционални методи, како што е развојот на водопади , како што се тимовите кои имаат агилен процес присилен врз нив. Тие често се нарекуваат агилни анти-модели или почесто агилни мириси. Подолу се наведени некои општи примери:

* **Недостаток на целосен дизајн на производи**

Целта на агилниот развој на софтвер е да се фокусира повеќе на производство на работен софтвер и помалку на документацијата. Ова е во спротивност со моделите на водопади каде што процесот е често контролиран, а помалите промени во системот бараат значителна ревизија на придружната документација. Сепак, ова не го оправдува целосното извршување без анализа или дизајн на сите. Неуспехот да се обрне внимание на дизајнот може да предизвика тим да продолжи брзо на почетокот, но потоа да има значителни преправки потребни кога се обидуваат да го зголемат системот. Една од клучните карактеристики на агилниот развој на софтвер е дека е итеративна. Кога се прави правилно, дизајнот се појавува кога системот е развиен и се откриваат заеднички и можности за повторна употреба.

* **Додавање приказни за итерација во тек**

Во агилниот развој на софтвер, приказни се користат за дефинирање на барања и итерација е краток временски период во кој тимот се обврзува на одредени цели. Додавањето приказни за итерација во тек е штетно за добар проток на работа. Овие треба да се додадат во заостанатиот дел на производот и да се даде приоритет за последователна повторување или во ретки случаи повторувањето на повторувањето.

Ова не значи дека една приказна не може да се прошири. Тимовите мора да се справат со нови информации, кои можат да создадат дополнителни задачи за приказна. Ако новите информации ја спречуваат приказната да биде завршена за време на итерацијата, тогаш таа треба да се пренесе на следната итерација. Сепак, тоа треба да биде приоритет против сите останати приказни, бидејќи новите информации можеби го сменија првичниот приоритет на приказната.

* **Недостаток на поддршка од спонзори**

Агилниот развој на софтверот често се спроведува како обичен напор во организациите од тимови за развој на софтвер кои се обидуваат да ги оптимизираат нивните развојни процеси и да обезбедат конзистентност во животниот циклус на развој на софтвер. Со тоа што немаат спонзорска поддршка, тимовите може да се соочат со тешкотии и отпор од деловните партнери, други тимови за развој и менаџмент.

* **Недоволна обука**

Истражувањето спроведено од VersionOne покажа дека испитаниците не посочиле доволно обука како најзначајна причина за неуспешните агилни имплементации. Тимовите паднале во стапицата за преземање на намалени процеси на агилен развој на софтвер во споредба со другите методологии како што е водопадот значи дека не постојат вистински правила за агилен развој на софтвер.

* **Тимовите не се фокусирани**

Агилниот развој на софтверот бара од тимовите да ги исполнат обврските на производите, што значи дека тие треба да се фокусираат само на работата за тој производ. Сепак, членовите на тимот кои се чини дека имаат слободен капацитет често се очекува да преземат други работи, што им отежнува да им помогнат да ја завршат работата што им ја извршил нивниот тим.

* **Прекумерно подготвување / планирање**

Тимовите може да паднат во стапица за трошење премногу време за подготовка или планирање. Ова е вообичаена замка за тимови кои се помалку запознаени со агилниот развој на софтвер, каде што тимовите се чувствуваат обврзани да имаат целосно разбирање и спецификација на сите приказни. Тимовите треба да бидат подготвени да се движат напред само со оние приказни во кои имаат доверба, а потоа за време на итерацијата продолжуваат да откријат и да подготват работа за идните повторувања

# 7. АГИЛНО УПРАВУВАЊЕ

Терминот агилно управување се применува на итеративен, инкрементален метод за управување со проектирање и градење на активности на инженеринг, информатичка технологија и други деловни области кои имаат за цел да обезбедат нов развој на производ или услуга на високо флексибилен и интерактивен начин, врз основа на изразените принципи во Манифестот за агилен развој на софтвер.

Агилни Х техники, исто така, може да се нарече екстремно управување со проекти. Тоа е варијанта на итеративен животен циклус каде испораките се поднесуваат во фази. Главната разлика помеѓу агилниот и итеративниот развој е дека агилните методи ги завршуваат малите делови од испораките во секој циклус на доставување (итерација), [додека итеративните методи со текот на времето го развиваат целиот збир на испорачувања и ги завршуваат во близина на крајот на проектот. Итеративните и агилни методи беа развиени како реакција на разни пречки кои се развија во повеќе секвенцијални форми на организацијата на проектот. На пример, со оглед на тоа што технологичните проекти растат во сложеност, крајните корисници имаат тенденција да имаат тешкотии во дефинирањето на долгорочните барања, без да бидат во можност да ги видат прогресивните прототипови. Проектите кои се развиваат во итерации постојано можат да соберат повратни информации за да помогнат во усогласувањето на тие барања.

Агилниот менаџмент, исто така, нуди едноставна рамка за промовирање на комуникација и размислување за минатото работење меѓу членовите на тимот. Тимовите кои користеле традиционално планирање на водопади и го усвоиле агилниот начин на развој, обично одат низ фаза на трансформација и често ја користат помошта од агилни тренери кои помагаат во водењето на тимовите преку мазна трансформација. Постојат типично два стили на агилно тренирање: притисни-базирани и повлечете-базирани агилни тренери . Пристапот за прифатливо управување е исто така вработен и прилагоден на деловните и владините сектори. На пример, во рамките на сојузната влада на САД , Агенцијата на САД за меѓународен развој(УСАИД) користи пристап за управување со проекти за соработка кој се фокусира на инкорпорирање на стратегии за соработка, учење и адаптирање (ОАК) за итерација и прилагодување на програмирањето.

# ЗАКЛУЧОК

Често очекувањата и реалноста не се координирани, често и софтверот можеби ќе биде креиран во предвидениот временски рок и со предвидените средства но, нема да го има очекуваниот ефект и очекуваната корист на пазарот на побарувачка. Најголемите проблеми при развојот на нов продукт произлегуваат од непредвидливоста на ризикот, можноста за креирање на софтвер сосема поинаков од оној кој се очекува да се достави, неможноста за прилагодување кон промени во барањата на клиентите на почетокот, во текот, а често и кон крајот на развојниот процес, при што сето тоа се сведува на изгубена инвестиција, време и труд. За да се избегне можноста за таков неуспех мора мудро да се направи избор на соодветна методологија и алатка за работа.

Кога е неопходно да се промени проектот и адаптивниот тим исто така се менува со неа. На адаптивниот тим има проблем точно да се опише што ќе се случи во иднина. Што е датумот на кој што се вели понатаму, тоа ќе бидат оние кои се занимаваа со некои прилагодливи методи за да бидат нејасно на детали кои треба да се објасни тогаш што ќе се случи со проектот. Прилагодливиот тим точно ќе опише што ќе биде следната недела, но следниот месец ќе има само идеја дека карактеристиките потоа треба да се додадат во програмата. Извештаи адаптивни тим на следните шест месеци може да содржи само информации за очекуваните резултати во зависност од услови и барања, како и официјални изјави во врска со планот за ослободување на следната верзија.

Спротивно на тоа, предвидливи методи се фокусираат на детално планирање во иднина. Овој тим точно знае какви карактеристики и задачи се планираат во секое време во текот на целото времетраење на процесот на имплементација на софтверски проекти. Меѓутоа, тешко му е да го промени правецот на дејствување. Планот е генерално оптимизиран за главната цел и оваа промена на насоката може да предизвика целосно отфрлање на претходната работа и повторно да започне на нов начин. Предвидните тимови често формираа специјална комисија за контрола на промените со цел да ги разгледаат само најважните.