БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

Институт по математика и информатика

Николай Иванов Тодоров

Гъвкави методологии за разработка на софтуерни проекти и тяхното приложение, основано на утвърдени стандарти за управление и качество

АВТОРЕФЕРАТ

на дисертация за присъждане на образователна и научна степен

"ДОКТОР"

област на висше образование 4. Природни науки, математика и информатика;
професионално направление 4.6 Информатика и компютърни науки;

научна специалност 01.01.12. "Информатика"

Научен ръководител

проф. д-р Аврам Ескенази

София

2013

СЪДЪРЖАНИЕ

УВОД

Гъвкавите (agile) методологии започват да стават все по-актуални и използвани в съвременния процес на разработка на софтуер. Итеративният цикъл на изпълнение, отвореността към постоянна промяна, тясното сътрудничество с клиента и между разработчиците се превръщат във все потърсени характеристики и отговарят във все по-пълна степен на днешните бизнес нужди.

Съществуват няколко основни методологии като Екстремно програмиране (XP) [1], Scrum [2], Разработване по функционалности (FDD) [3], Адаптивно разработване на софтуер (ASD) [4] и други. В основата си те се опитват да намалят рисковете чрез разработване в кратки периоди от време, наречени итерации, които обикновено продължават от една до четири седмици. Всяка итерация сама за себе си е като малък софтуерен проект, включващ всяка от фазите, необходими за реализирането и предоставянето на нова функционалност - планиране, анализ, проектиране, кодиране, тестване и документация. Гъвкавите методологии наблягат на това да се създаде работещ софтуер като главен критерий за успех. Заедно с предпочитанието за комуникация лице в лице, те изискват и генерират много по-малко документация в сравнение с останалите методи.

От друга страна съществуват утвърдени стандарти за управление на проекти и тяхното качество (засягащи не само областта на софтуерната разработка). Такива са Системата от знания за управление на проекти (Project Management Body of Knowledge – PMBOK [5]) и Интегрираният модел за зрялост на възможностите (Capability Maturity Model Integration – CMMI [6]). Те предоставят цялостната рамка на процеса за разработка на софтуер, като дефинират пълен набор от области на знание, правила, практики и средства за изпълнението на проекта.

Днешните ръководители на софтуерни проекти са изправени пред много предизвикателства. Изискванията и натискът към тях постоянно се увеличават в резултат на повишената конкурентна среда, нуждата от комплексни решения, постоянно променящите се технологии и всичко това е допълнително усложнено от трудните икономически условия. За да се справят с тези предизвикателства, ръководителите на проекти трябва да преосмислят традиционните подходи за управление и да се опитат да бъдат още по-гъвкави. Ефективното управление изисква от тях не само владеене на стандартни техники, но също така знания и

умения да се адаптират, да променят или пренапишат правилата в определени граници, когато ситуацията го изисква.

Съществува мнение, че гъвкавите методологии не могат да претендират за пълнота и завършеност от гледна точка на традиционните и утвърдени похвати за управление [7]. Причината, която се изтъква, е, че те наблягат основно на крайните резултати и взаимодействието в екипа и често са критикувани като недисциплинирани. От своя страна PMBOK и CMMI разчитат на добре документиран план на проекта, стриктно следене и контрол на изпълнението му и качеството му. Но за да успее ръководителят на проекта да се адаптира към съвременните условия, е необходимо да разбере добре управленските философии, които стоят в основата на гъвкавите методологии и да намери точното им съответствие с добре познатите му процеси, практики и средства в утвърдените стандарти. Важно е да се спомене, че според проучванията на Gartner [8] в края на 2012-а година 80% от софтуерните проекти се изпълняват на основата на гъвкави методологии.

Когато говорим за управление на софтуерни проекти, съществуват четири основни променливи, между които трябва се постигне точният баланс за успешното изпълнение на проекта. Това са:

- Обхват на реализираните функционалности (scope)
- Времето и графика за изпълнение на проекта (time)
- Цената за реализация (cost)
- Качеството на крайния продукт (quality).

Същите тези четири променливи са дефинирани като основополагащи и в гъвкавите методологии от Kent Beck [1]. Още тук виждаме, че гъвкавите и традиционните подходи поставят един и същ общ фокус при изпълнението на проекта.

Софтуерните проекти обаче се характеризират с някои специфични неща, които оказват голямо влияние за реализацията им и това са често срещани и характерни проблеми, пред които могат да се изправят по време на изпълнението си. Стандартите и модели като PMBOK и CMMI разглеждат тези проблеми като рискове, но не се спират на техни конкретни изражения и особено на способи за предпазването или справянето с тях. Познаването им и правилните подходи в такива ситуации може да доведе до съществено предимство и оптимизиране на една или дори повече от гореспоменатите четири променливи. Поради тази

причина решихме да разглеждаме проблемите, характерни за софтуерните проекти, като пета променлива, която трябва да управляваме и ще наречем „специфики“. Нямаме данни подобен подход на надграждане на класическите дефиниции за управление и качество да е разглеждан в литературата досега.

Целта на настоящата дисертация е да дефинираме и развием гъвкав процес, който целенасочено да съчетава в себе си елементи на гъвкавите методологии, следвайки утвърдените стандарти за управление на проекта и качеството PMBOK и CMMI. Този процес следва да покрива всички аспекти от управлението на един софтуерен проект, на основата на дефинираните принципи и практики в двата стандарта. Тази цел ще бъде постигната чрез решаването на следните **задачи**:

1. Да се направи обзор на основните гъвкави методологии (XP, Scrum, FDD и ASD) чрез анализ на техния процес, роли и отговорности, практики и ограничения
2. Да се направи сравнителен анализ между гъвкавите методологии от една страна и утвърдените стандарти за управление на проекта и качеството PMBOK и CMMI от друга.
3. Да се покаже, че тези методологии и стандарти за управление предоставят необходимия набор от практики и правила за ефективното управление и изпълнение на софтуерни проекти, като се вземе предвид и оптимизира всяка една от петте променливи – обхват, време, цена, качество, специфични проблеми.
4. На тази основа да се дефинира гъвкав процес, който покрива в достатъчна пълнота жизнения цикъл на разработката на софтуер.
5. Да се приложи така дефинираният процес в определени реални проекти, за да се проверят и потвърдят така получените резултати.

1. ГЪВКАВИ МЕТОДОЛОГИИ

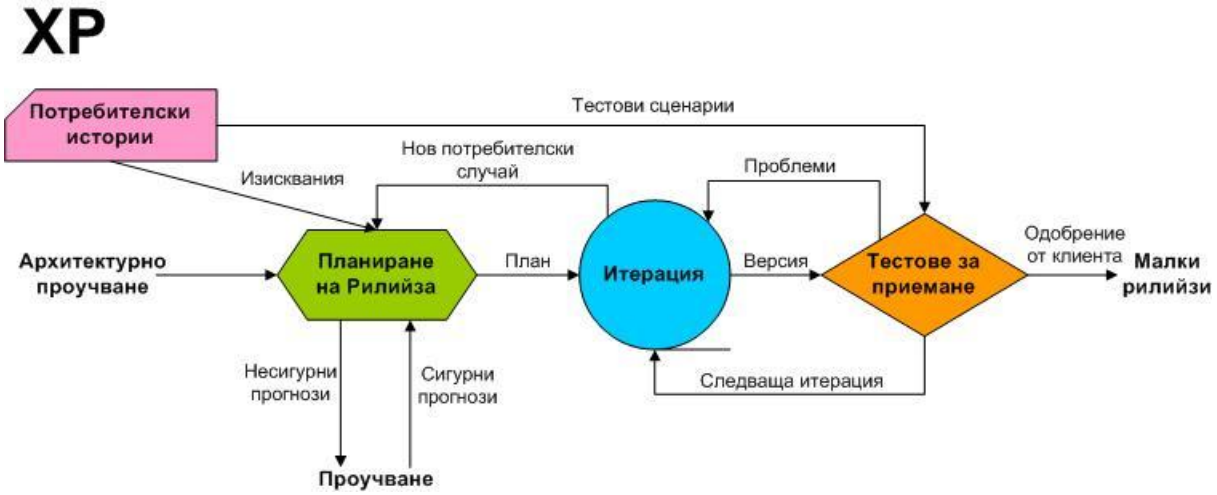
През февруари 2001 г. 17 разработчици на софтуер се срещат, за да обсъдят олекотяване на процесите за разработка на софтуер. Те публикуват манифест на гъвкавите методологии *[9]*, за да дефинират подхода, станал известен като гъвкави методологии за разработка на софтуер.

През следващите няколко години тези методологии започват да се налагат като естествена реакция на традиционните подходи за разработка на софтуер. Причина за това е осъзнатата нужда от алтернатива на тежките, основани силно на документацията процеси [10]. Съществуват множество гъвкави методологии за разработка на софтуер. Общото между тях е, че се стремят винаги да се адаптират към постоянно променящите се условия и изисквания, които съпътстват един софтуерен проект. Всички те са набор от добре познати практики, но комбинирани по такъв начин, който да доведе проекта до успешен завършек. Не всяка от тях покрива целия процес на разработка. Някои от тях се концентрират само върху определени фази. Не винаги една и съща методология е приложима универсално във всяка една област, в която се нуждаем от софтуерно решение. Дори не винаги една и съща методология е подходяща два пъти последователно. Успешното прилагане на комбинация от различни методологии с техните практики би довело до успешното изграждане на собствен процес в рамките на един софтуерен проект.

1.1. Екстремно програмиране (Extreme Programming - XP)

Екстремното програмиране (XP – Фиг.1) се развива в резултат на проблемите, които изникват в цикъла на разработка на стандартните модели на процеси. Стартира просто като “възможност да се свърши работата” с практики, които са доказали своята ефективност при разработката на софтуер в предишни години. След поредица от успешни проекти в практиката XP е “теоретизирано” в основните си принципи от Kent Beck през 1999 г. Въпреки че отделните практики в XP не са нови, те са събрани и обединени да функционират помежду си по нов начин, формирайки нова методология за разработка на софтуер. Тя е система от практики, която обществото от разработчици еволюционно развива, за да се справи с проблеми като бързо доставяне на качествени продукти, посрещащи нуждите на бизнеса [13]. Терминът “extreme” (екстремен) идва от това, че тези стандартни практики се прилагат на екстремно ниво [14]. На Фиг. 1 може да се видят

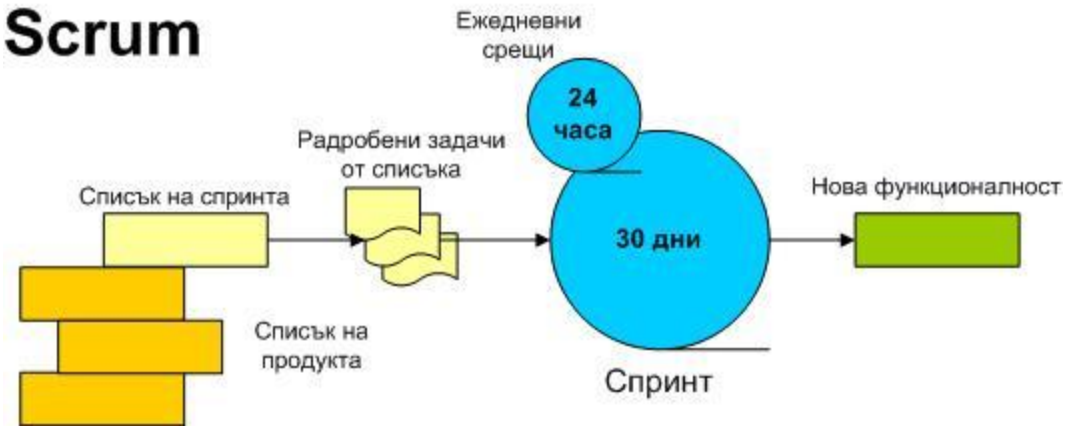
основните фази и практики, през които преминава един XP проект, които са разгледани по-подробно в дисертацията.



Фигура 1. XP процес

1.2. Scrum

Терминът Scrum произлиза от играта ръгби и означава скупчването на играчите около топката с цел придвижването й напред. Този подход (Фиг. 2) е създаден, за да управлява процеса по разработка на софтуер. Той е "емпиричен подход, прилагащ идеите на теорията за управление на индустриалните процеси при разработката на софтуер, в резултат на което се проявяват идеите за гъвкавост, адаптивност и продуктивност" (Schwaber and Beedle [2]). Процесът дава възможност на софтуерните компании да реализират проектите си бързо и незабавно да почнат да пускат в експлоатация нова версия на продукта си на всеки 30 дни [15]. Не се дефинират някакви специфични техники за разработка за фазата на реализация. Scrum се концентрира върху това как членовете на екипа трябва да работят заедно, за да изградят системата гъвкаво в постоянно променящата се среда. Scrum помага да се подобрят съществуващите практики в организацията, като набляга на честите намеси на мениджмънта, за да се идентифицират всякакви проблеми и пречки по време на процеса.

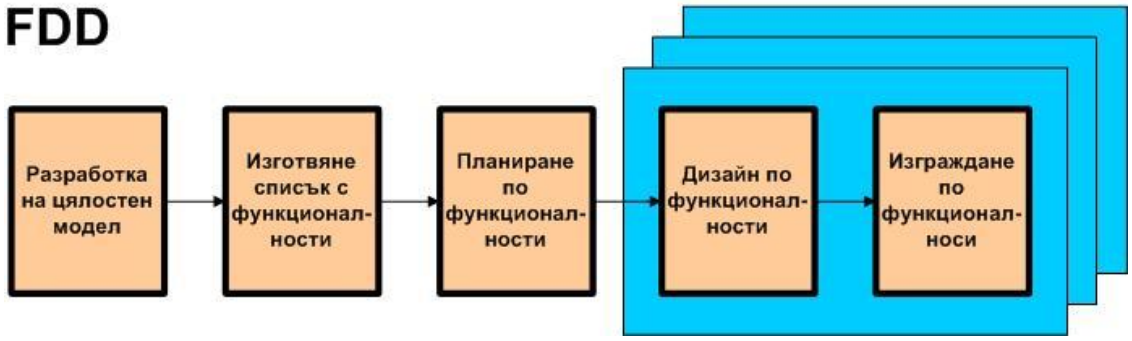


Фигура 2. Scrum процес

1.3. Разработване по функционалности (Feature Driven Development - FDD)

Разработването по функционалности (FDD – Фиг. 3) е гъвкав и адаптивен подход при разработката на софтуерни системи. Той не покрива целия софтуерен процес, а по-скоро се фокусира върху фазите на проектиране и реализация [3]. FDD съчетава итеративния метод на разработка с най-добрите практики, доказали се като ефективни в индустрията. Набляга на качеството по време на процеса и включва често предоставяне на функционалности на клиента, както и непрекъснато наблюдение на напредъка. Целта е всяка функционалност да бъде така описана и дефинирана, че да може клиентът да потвърди нейната стойност и да вземе решение да я включи в продукта [16].

FDD се състои от пет последователни процеса и предоставя методите, техниките и напътствията, необходими на участниците в проекта за изграждането на системата. Също така, FDD включва роли, цели и графици, необходими за проекта. Твърди се, че за разлика от някои други методологии, FDD е подходящ за разработка на критични за бизнеса системи [3].



Фигура 3. FDD процес

1.4. Адаптивно разработване на софтуер (Adaptive Software Development)

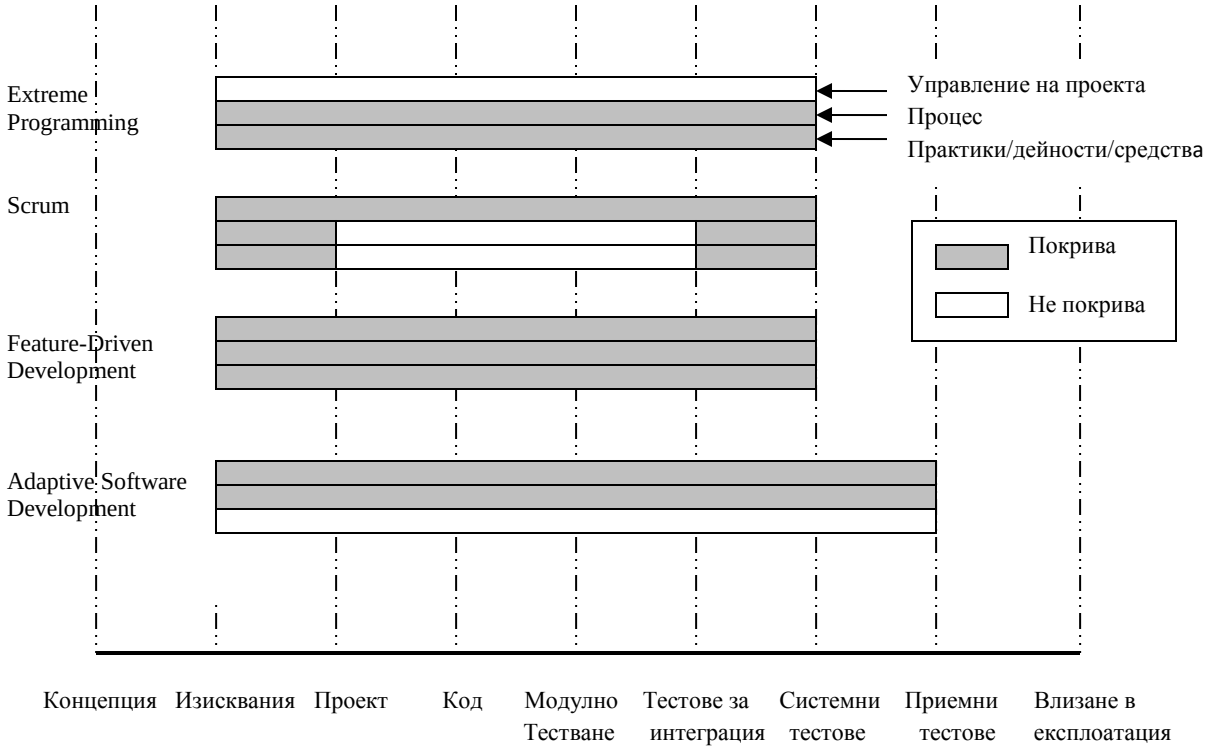
Адаптивното разработване на софтуер (Фиг. 4) е развито от James A. Highsmith III [4]. ASD се фокусира основно върху разработката на големи и сложни системи [17]. Методологията силно препоръчва надграждащо, итеративно разработване с постоянно съставяне на прототипи. Като цяло, ASD се опитва да "балансира на ръба на хаоса" – целта му е да зададе рамка с достатъчно напътствия, за да не изпадне проектът в хаос, но без да прекалява много, за да не потиска себеизявата и креативността.

Фигура 4. ASD процес

1.5. Сравнение на гъвкавите методологии

Всяка от разгледаните методоологии се намира в различна фаза от своето развитие. XP и Scrum са вече изградени методологии, които са добре документирани и съществуват множество материали и отзиви за тях. Има изградени общества, които се занимават с изучаването и прилагането им. Затова можем да ги определим като “Активни”. FDD и ASD ги определяме като “Изграждащи” се методологии, защото все още има твърде малко информация за тях и липсват конкретни отзиви за резултати от прилагането им. Въпреки че изхождат от сходна позиция и имат подобни принципи, всяка една от гъвкавите методологии поглежда под различен ъгъл към разработката на софтуер.

Гъвкавите методологии не винаги покриват всяка фаза от процеса на разработка на софтуер. Фиг. 5 показва коя фаза от кой метод се покрива [12]. Всеки метод е разделен на три части. Първата показва дали методът предлага начини за поддръжка на управлението на проекта. Втората – дали процесът, предложен от метода, е описан в самия него. Третата показва дали методът описва практиките, действията и средствата, които трябва прилагат и използват. В сиво са оцветени блоковете, които покриват съответната фаза, а в бяло – които не покриват.



Фигура 5. Покритие на софтуерния процес

Това, което е важно за целта на настоящата разработка е, че различните методологии покриват различни аспекти на процеса и **най-цялостният подход за управлението на един софтуерен проект би било тяхното целенасочено, обосновано и в крайна сметка – ефективно комбиниране**.

2. СТАНДАРТИ ЗА УПРАВЛЕНИЕ И КАЧЕСТВО

Съществуват утвърдени стандарти за управление на проекти и тяхното качество (засягащи не само областта на софтуерната разработка). Такива са Системата от знания за управление на проекти (Project Management Body of Knowledge – PMBOK) и Интегрирания модел за зрялост на възможностите (Capability Maturity Model Integration – CMMI). Те предоставят цялостната рамка на процеса за разработка на софтуер като дефинират пълен набор от области на знание, правила, практики и средства за изпълнението на проекта. Тези стандарти са изцяло дефинирани от международно признатите организации, които ги разработват. Ще направим кратък обзор на техните основни принципи и особености.

2.1. Система от знания за управление на проекти (Project Management Body of Knowledge – PMBOK)

„Система от знания за управление на проекти" и нейното ръководство (PMBOK® Guide) е признат стандарт за професионалния ръководител на проекти [18]. Стандартът е официален документ, който описва установените норми, методи, процеси и практики. Както и при другите професии, като право, медицина, счетоводство, съдържащото се знание в този стандарт се развива от признатите добри практики на управляващите ги ръководители на проекти, които са допринесли за развитието му. Ръководството PMBOK® Guide посочва насоките за ръководене на отделни проекти. То дава определение на управлението на проектите и свързаните с него понятия и описва жизнения цикъл на управлението и характерните за него процеси.

Нарастващото приемане на управлението на проекти показва, че прилагането на необходимите знания, процеси, умения, средства и методи може да има значително въздействие върху успеха на проекта. Ръководството PMBOK® Guide идентифицира системата от знания за управление като общоприети добри практики. „Общоприети" означава, че описаните знания и практики обикновено са приложими за повечето проекти и има консенсус за тяхната стойност и полезност. „Добра практика" значи, че има общо съгласие, че прилагането на тези умения, средства и методи може да подобри шансовете за успех в широк кръг от проекти. „Добра практика" не предполага, че описаното знание винаги трябва да се прилага

еднакво за всички проекти; организацията и/или екипът за управление на проекта е отговорен за определянето на това, какво е подходящо за даден проект.

Ръководството PMBOK® Guide също така предоставя и насърчава обща лексика в рамките на професионалното управление на проекти за обсъждане, писане и прилагане на концепции за управление на проекти. Такъв стандартен речник е основен елемент на професионална дисциплина. Институтът по управление на проекти (Project Management Institute – PMI, www.pmi.org) разглежда този стандарт като фундаментална концепция за управление на проекти за своите програми за професионално развитие и сертификация.

2.2. Интегриран модел за зрялост на възможностите (Capability Maturity Model Integration – CMMI)

Моделите на зрялост на възможностите (Capability Maturity Model® - CMM®), включително и интегрираният такъв (CMMI), се стремят да представят опростен модел на заобикалящия ни свят. CMM моделите съдържат основни елементи на ефективните процеси и качество. Тези елементи са базирани на концепции разработени от Crosby, Deming, Juran и Humphrey.

На български не съществува официален превод на съкращението CMMI – Capability Maturity Model Integrated. Най-често срещаният директен превод е „Интегриран модел за зрялост на възможностите“. Най-лесно обаче, е CMMI да бъде дефиниран описателно като модел за зрялост и качество на процесите. Подобен впрочем е и подходът на д-р Георги Шарков (водеща фигура по разпространението и внедряването на CMMI у нас), който в лична кореспонденция с нас застъпва „Модел за зрялост и качество на процесите в софтуерни и ИТ организации/фирми“. Трябва обаче моделът CMMI да се разграничава от модела CMM . Също така, най-често срещаната приложна област е ИТ организации и фирми, но самият модел се стреми да не се ограничава само в тази сфера. Поради тези причини оттук нататък ще използваме единствено английското съкращение CMMI.

CMMI е подход и ръководство за подобрение на процесите и качеството [6]. Това може да бъде използвано както в рамките на отделни проекти или отдели, така и в цялата организация. Процесите биват оценени съобразно тяхното ниво на зрялост, които са дефинирани като „Първоначално“, „Управляемо“, „Дефинирано“, „Количествено Управляемо“ и „Оптимизиращо“.

CMMI в момента се развива в три основни области:

- Разработка на продукти и услуги – CMMI за разработка (CMMI for Development – CMMI-DEV)
- Управление на услуги – CMMI за услуги (CMMI for Services – CMMI-SVC)
- Придобиване на продукти и услуги – CMMI за придобиване (CMMI for Acquisition – CMMI-ACQ)

CMMI е разработен от група експерти от индустрията, администрацията и SEI към Carnegie Mellon University. Моделът предоставя ръководство за развитие и подобрение на процесите с цел да се постигнат бизнес целите на организацията. CMMI също се използва като основа за оценка на нивото на зрялост на организацията.

Моделът произлиза от софтуерната индустрия, но през годините е обобщен в голяма степен, за да може да покрие и други области като производство на хардуер, доставка на всякакъв вид услуги и придобиването на продукти и услуги. Дори думата „софтуер“ не се среща в дефинициите на CMMI. Това обобщение на концепциите за подобрение прави моделът сравнително абстрактен. Той не е процес, по който да бъде разработван даден продукт или услуга, а рамка за качество, към която трябва да се стремим.

3. СРАВНИТЕЛЕН АНАЛИЗ НА ГЪВКАВИТЕ МЕТОДОЛОГИИ И СТАНДАРТИТЕ ЗА УПРАВЛЕНИЕ И КАЧЕСТВО

Както беше споменато в уводната глава, съществува мнение, че гъвкавите методологии не могат да претендират за пълнота и завършеност от гледна точка на традиционните и утвърдени похвати за управление и качество [7]. Причината, която се изтъква, е, че те наблягат основно на крайните резултати и взаимодействието в екипа и често са критикувани като недисциплинирани. От своя страна PMBOK и CMMI разчитат на добре документиран план на проекта, стриктно следене и контрол на изпълнението му и качеството му.

В следващите няколко секции ще направим точно такъв сравнителен анализ за основните променливи от управлението на един софтуерен проект.

- Обхват на реализираните функционалности (scope)
- Времето и графика за изпълнение на проекта (time)
- Цената за реализация (cost)
- Качеството на крайния продукт (quality)
- Характерни проблеми или специфики – които ние за първи път допълнително дефинираме, тъй като ги считаме за изключително важен елемент в софтуерната разработка, успоредно и заедно с първите 4 променливи. В следващите няколко секции ще направим точно такъв сравнителен анализ за основните променливи от управлението на един софтуерен проект.

За първите четири променливи ще разгледаме процесите и практиките, които дефинират утвърдените стандарти за управление и качество (PMBOK и CMMI) и ще потърсим еквивалентните практики, които ни предоставят гъвкавите методологии и ще направим категоризация на съпоставката на следните три нива:

- ДА – налично е пълно съответствие между елемента, дефиниран в PMBOK и дадена практика от гъвкавите методологии
- Частично – не всички аспекти от дефинираното в PMBOK се покриват от коя да е методология или съвкупност от тях. Това ще бъде обяснявано в по-големи детайли в съответната секция
- НЕ – не съществува (не сме открили) съответна практика в гъвкавите методологии, която да покрива описаното в PMBOK.

Софтуерните проекти обаче се характеризират с някои специфики, които оказват голямо влияние върху реализацията им и това са често срещани и характерни проблеми, пред които могат да се изправят по време на изпълнението си. Поради тази причина разглеждаме тях като пета променлива, която трябва да управляваме, и ги наричаме „специфики“. За тези специфики ще направим сравнителен анализ как и дали някоя от четирите разгледани гъвкави методологии в първата глава (XP, Scrum, FDD и ASD) дефинира начин да се справим с тях.

Гъвкавите методологии в своята изначална идеология разчитат в голяма степен в своите принципи и практики на реални предмети и визуални, физически начини за тяхното представяне [19]. Такива са картите за описване на потребителски истории (CRC cards), дъските с Диаграма на напредъка [20] (Burndown chart) или Kanban дъски [25] и много други. В наши дни съществуват обаче и множество системи за управление на проекти, които са изцяло проектирани да покриват процесите, принципите и практиките, дефинирани от гъвкавите методологии (Jira [23], Trac [24] и др.). Те предлагат в електронен вариант същите средства за описание и визуализация. При нарастване на обхвата и продължителността на един проект е силно препоръчително да се използва някоя от тези системи, за да се улесни управлението и проследяването на проекта, както и да се запазят исторически резултатите му в дългосрочен план. Поради тази причина за някои от сравненията ще използваме като пример за съпоставка предлаганите средства именно от тези системи за управление на проекти.

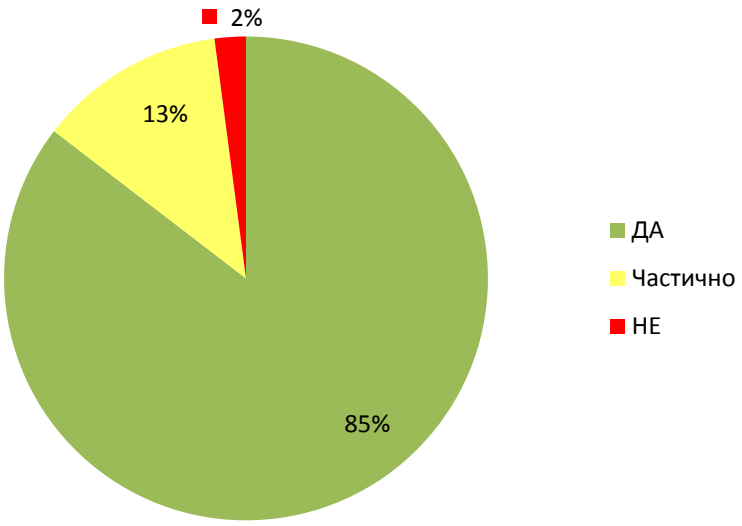
3.1. Управление на обхвата

Според PMBOK управлението на обхвата включва процесите, необходими за да се уверим, че в проекта ни е идентифицирана и включена цялата необходима дейност (и единствено тя), за да може той да бъде успешно завършен [5]. Те са общо 5 на брой и ще бъдат разгледани в следващите секции.

В гъвкавите методологии итеративните и надграждащи процеси са това, което управлява обхвата. Единствено това следва да бъде описано като процедура за управление на обхвата, освен ако нещо допълнително не е необходимо да бъде документирано с цел одит. Обхватът бива дефиниран и предефиниран постоянно по време на срещите за планиране на итерацията или версията чрез отразяването на необходимите елементи в списъка на продукта [21].

В резултат на направения анализ относно областта от знание за управление на обхвата в PMBOK и принципите и практиките дефинирани от гъвкавите

методологии в това отношение, можем да направим следното обобщение на степента на съответствие:



Фигура 6. Управление на обхвата

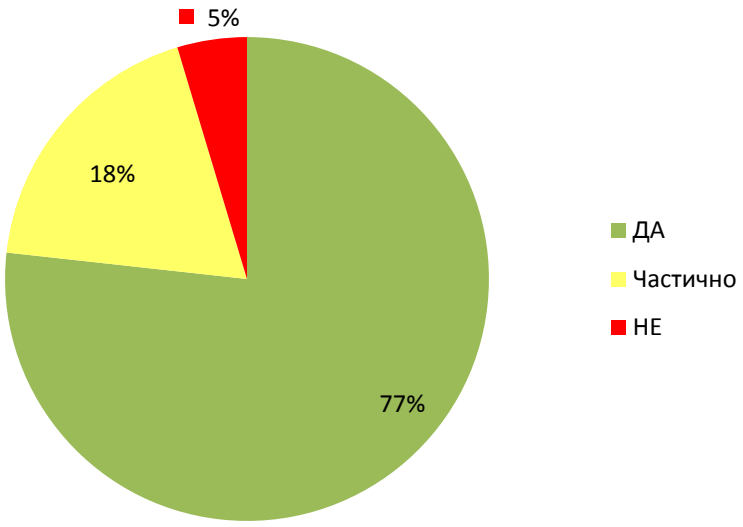
Виждаме, че в областта на управление на обхвата имаме почти пълно покритие от процеси и практики, дефинирани от гъвкавите методологии, които да изпълняват функциите, дефинирани в стандарта PMBOK.

3.2. Управление на времето

Според PMBOK областта от знание за управлението на времето включва процесите, необходими за навременното приключване на проекта. Заедно със своите средства и техники те са документирани в плана за управление на графика.

В гъвкавите методологии итерациите са винаги фиксирани за определен период (който обикновено е между 1 и 4 седмици [22]). Поради тази причина времето се управлява в тази рамка чрез използването на съответните практики. Това дава възможност екипът да бъде фокусиран върху най-важните цели и задачи.

В резултат на направения анализ относно областта от знание за управление на времето в PMBOK и принципите и практиките дефинирани от гъвкавите методологии в това отношение, можем да направим следното обобщение на степента на съответствие:



Фигура 7. Управление на времето

Виждаме, че в областта на управление на времето отново имаме голяма степен на съответствие между елементите, описани в PMBOK, и процеси и практики, дефинирани от гъвкавите методологии. Все пак имаме известно количество частични или липсващи съответствия. Повечето могат да се обяснят с това, че PMBOK се стреми да дефинира подходи, които да намалят риска от несигурност в един дългосрочен проект. Нещо, което в гъвкавите методологии се елиминира още от първоначалната им концепция за кратки итерации.

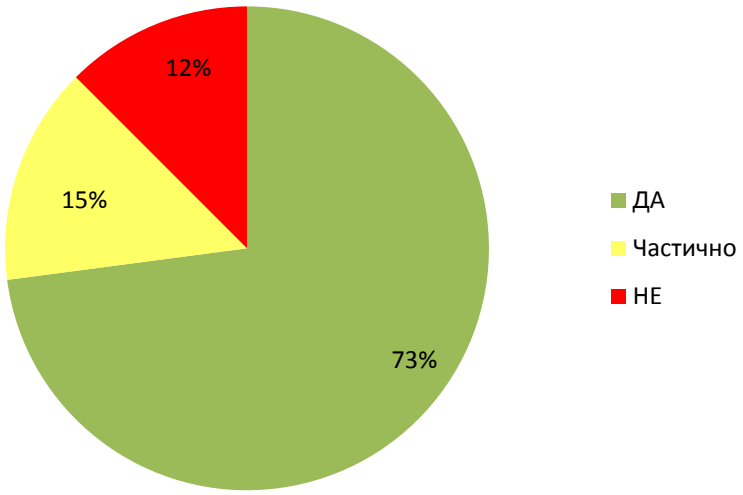
3.3. Управление на цената

Областта от знание за управление на цената в PMBOK включва процесите по оценка, определяне на бюджет и контрол на разходите, така че проектът да бъде изпълнен в одобрените рамки. За някои проекти, особено по-малките такива, оценката и определянето на бюджета са тясно свързани и дори могат да бъдат разглеждани като един процес. Така или иначе обаче средствата и техниките за единия и другия са различни.

При гъвкавите методологии итерациите са винаги фиксирани за даден период (между 1 и 4 седмици). Екипът също е фиксиран и изцяло фокусиран върху проекта [26]. Той оценява усилията, необходими за всяка функционалност, която ще влезе в итерацията и това директно се преобразува в нейната цена, на база на

ставката. Това дава възможност на клиента динамично да решава за обхвата и приоритетите на функционалностите, които иска да включи в дадена итерация.

В резултат на направения анализ относно областта от знание за управление на цената в PMBOK и принципите и практиките дефинирани от гъвкавите методологии в това отношение, можем да направим следното обобщение на степента на съответствие:



Фигура 8. Управление на цената

Виждаме, че в областта на управление на цената отново имаме голяма степен на съответствие между елементите описание в PMBOK и процеси и практики, дефинирани от гъвкавите методологии. Все пак имаме известно количество частични или липсващи съответствия. Повечето могат да се обяснят с това, че PMBOK обръща внимание и на договорните аспекти от управлението на проекта, докато гъвкавите методологии се фокусират повече върху процеса по изпълнение.

3.4. Управление на качеството

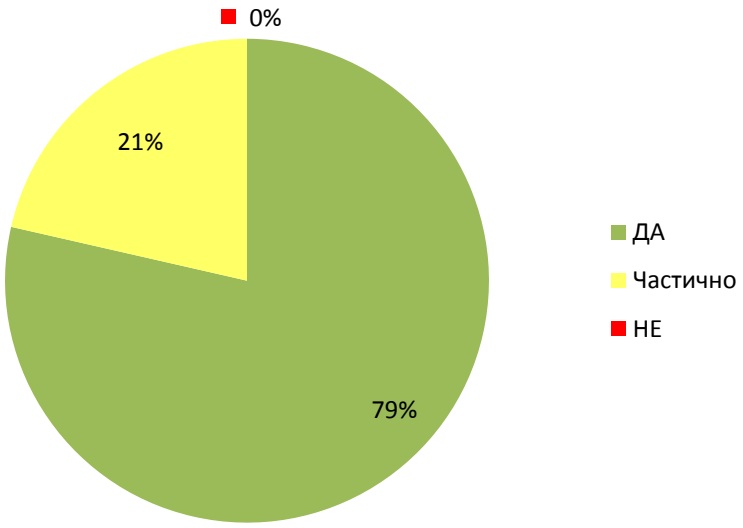
Качеството при разработката на един софтуерен проект от инженерна гледна точка засяга няколко основни аспекта, които трябва да бъдат добре дефинирани и следвани:

- Развиване на изискванията
- Техническо решение

- Интеграция на продукта
- Верификация
- Валидация

CMMI моделът предоставя рамка за това какво трябва да бъде изпълнено във всеки един от тях, за да бъде гарантирано качеството на проекта. Ще се спрем на една от разновидностите на CMMI, а именно за разработка на продукти и услуги (CMMI for Development – CMMI-DEV) [27].

В резултат на направения анализ относно инженерните области в CMMI, които обхващат качеството на продукта и принципите и практиките, дефинирани от гъвкавите методологии в това отношение, можем да направим следното обобщение на степента на съответствие:



Фигура 9. Управление на качеството

Виждаме, че в областта на управление качеството имаме почти пълно покритие от процеси и практики дефинирани от гъвкавите методологии, които да изпълняват функциите, дефинирани в CMMI модела.

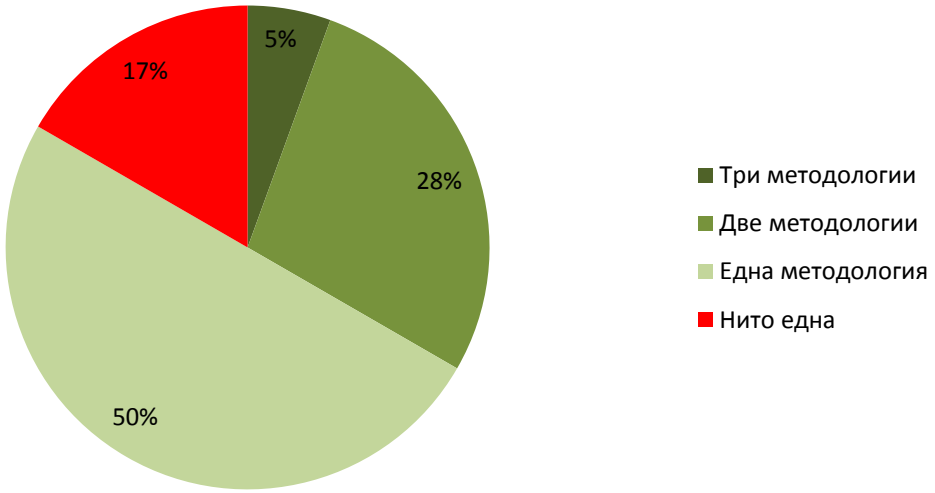
3.5. Управление на спецификите

Управлението на съвременните софтуерни проекти изисква умело контролиране на възможните специфични проблеми, както на очакваните, така и на неочакваните ситуации. Мощно средство, с което всеки ръководител на проекти

трябва да разполага, за да се справи с тези предизвикателства, е познаването на възможностите за предпазване или преодоляване на тези специфики, предоставяни му от гъвкавите методологии.

Ще декомпозираме разработката на софтуерни проекти в три фази – стартиране, изпълнение и приключване. За всяка от тях ще разгледаме потенциалните трудности, пред които можем да се изправим. По-горе също така разгледахме четири от основните методологии – XP, Scrum, FDD и ASD. Ще се опитаме да изложим систематичен подход, разглеждайки как всяка от тях се справя (и дали) с тези специфики, които изникват пред софтуерните проекти. Целта е да дадем конструктивни съвети в тази насока на ръководителите на проекти.

Частични такива анализи са правени от Ketuunen и Laanti [28] в областта на вградените (embedded) системи и Тодоров и Ескенази [29] за портални решения. Ще се опитаме да обобщим тези разработки цялостно за софтуерните проекти. В долната фигура ще покажем какъв процент от спецификите се покриват от съответно три, две, една или нито една от методологиите. Този анализ е развит подробно в дисертацията.



Фигура 10. Покритие на специфики

Виждаме, че почти всички от специфичните проблеми при разработката на софтуерни проекти се покриват от поне една от гъвкавите методологии. Доброто познаване и правилното управление на тези специфики може значително да оптимизира останалите класически четири променливи. Анализът ни отново

показа, че за над 80% от проблемите поне една от методологиите разполага със средства за тяхното разрешаване. Там, където такива липсват, е необходимо процесът да се надгради извън рамките на конкретната методология, за да може да се обърне внимание на всички потенциални проблеми.

3.6. Обобщение на анализа

В предходните няколко глави направихме анализ на четирите класически променливи при управлението на един софтуерен проект – обхвата, времето, цената и качеството. За всяка от тях преминахме през дефинициите какво трябва да включват, определени от признатите стандарти като PMBOK и CMMI. Потърсихме еквивалентни принципи или практики в гъвкавите методологии, които да ги покриват. Ще обобщим резултатите в долната таблица:

Променлива	ДА	Частично	НЕ
Обхват	85%	13%	2%
Време	77%	18%	5%
Цена	73%	15%	12%
Качество	79%	21%	0%

Таблица 1. Обобщена съпоставка

Добавихме още една променлива към горните класически четири, която считаме, че е изключително важна в областта на управление на софтуерни проекти и трябва да се вземе предвид. Това са техните специфики, за които също направихме анализ как гъвкавите процеси се справят с тях.

Обобщено се оказва, че приблизително 85-95% от аспектите на тези пет променливи могат напълно или поне частично да бъдат покрити чрез гъвкавите методологии. Тази съпоставка е валидна както за всяка една от разглежданите променливи, така и за цялостния жизнен цикъл на проекта.

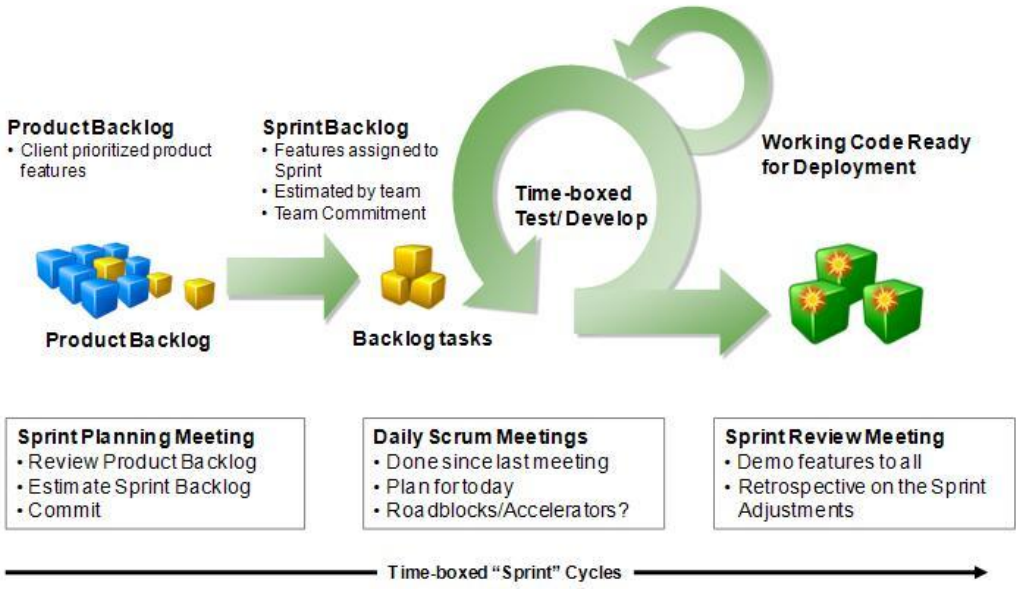
Това ни дава основание да считаме, че **може да се дефинира един изцяло гъвкав процес**, който да се основава на съвкупност от принципите и практиките на различните методологии. Същевременно той може да бъде достатъчно пълен, за да отговаря на нуждите на един софтуерен проект съобразно признатите стандарти и модели за управление и качество. Целта на следващите глави ще бъде да се дефинира този процес и той да бъде приложен в няколко реални софтуерни проекта.

4. ГЪВКАВ ПРОЦЕС ЗА РАЗРАБОТКА НА СОФТУЕРНИ ПРОЕКТИ

На базата на всичко развито досега ще дефинираме цялостен гъвкав процес за разработка на софтуерни решения, който да комбинира практиките и принципите на известните гъвкави методологии и да покрива всички аспекти, определени от петте променливи.

Гъвкавият процес, който разработваме, се базира на жизнения цикъл дефиниран от Scrum методологията, като развива в детайли процесната рамка, комбинирайки я с принципи и практики, описани в останалите методологии (като XP, FDD и т.н.). Този избор се определя от анализа, направен в първата глава и аспектите от софтуерния процес, които всяка методология покрива. Ще определим основните роли, събития и елементи, които трябва да бъдат спазвани или използвани по време на разработката на проекта. Процесът ще бъде основан на итерации (наричани още спринтове) от по две седмици. Обосновката на този избор се намира в дисертацията.

Целта на всяка итерация е да достави работещ продукт по такъв начин, че той да може да бъде пакетиран и доставен на крайния потребител с реализираните до момента функционалности. Важно условие за взимане на такова решение в края на итерацията е всички функционалности да са били тествани и да покриват формалните критерии за приемане на версията.



Фигура 11. Жизнен цикъл, базиран на Scrum

Не всяка итерация може да завършва с пускане в експлоатация на нова версия на продукта. В дългосрочна перспектива е необходимо да се изгради план за версиите, които ще бъдат доставени до крайния клиент. Обикновено това се обвързва с бизнес сроковете и набора от функционалности, които ще бъдат включени. Трябва да се планират 2-3 версии напред, което може да включва 5-6 итерации. Това включва и оценка и одобрение на необходимия бюджет на база периода и размера на екипа.

Както отбелязахме по-рано в анализа, съществуват множество системи за управление на проекти, които са изцяло проектирани да покриват процесите, принципите и практиките дефинирани от гъвкавите методологии (Jira[23], Trac [24] и др.). Те предлагат в електронен вариант всички необходими средства за описание и визуализация на елементите на процеса.

4.1. Екип, роли и отговорности

Екипът, който работи по проекта, се състои от няколко основни роли. Силно препоръчително е всяка роля да се изпълнява от точно един човек и да не се разпределя между няколко:

- Собственик на продукта
- Екип по разработката
- Scrum ръководител
- Мениджър
- Клиент
- Правила за разрастване на екипа (роене)

4.2. Стъпки на процеса и събития

По време на процеса и жизнения цикъл на проекта и итерациите се минава през определени стъпки и се случват определени събития. Те имат регулярен характер и по този начин оптимизират времето, изразходвано от екипа за тях, както и елиминират нужда от допълнителни, непланирани, трудно управляеми срещи. Всички събития трябва да бъдат фиксирани във времето, така, че да имат дефинирана максимална продължителност. Това фокусира самите участници по време на тези срещи да не изпадат в дълги дискусии или такива, които са по

странични теми, което в крайна сметка спестява от тяхното време. Стъпките и събитията са следните:

- Спринт
- Разработване на потребителски истории
- Среща за планиране на спринта
- Ежедневни срещи
- Преглед на спринта
- Ретроспекция на спринта

4.3. Елементи на процеса

Елементите на процеса представят работни продукти или правила и принципи, които се използват по време на процеса.

- Списък на продукта
- Списък на спринта
- Връзки и препратки
- Преглед на кода (ревю)
- Дефиниция за завършеност
- Наблюдение на напредъка на спринта
- Архитектурен и технологичен подход
- Постоянна интеграция
- Краен продукт

4.4. Тестване и осигуряване на качеството

Необходимо е да се дефинира план и процес за тестване по време на всеки спринт, за да се осигури качеството на крайния продукт. Повечето гъвкави методологии не дефинират явно такъв, затова ще взаимстваме популярни добри практики, съчетавайки ги с принципите на гъвкавите методологии, за да изградим такъв процес.

Като инструментариум отново може да бъде използван Trac и неговото разширение Тест Мениджър, чрез което да се изпълняват процедурите по тестване и да се записват изходните резултати.

4.5. Обобщение

Така дефинираният процес обхваща всички аспекти от разработката и управлението на един софтуерен проект. Той е основан изцяло на принципите на гъвкавите методологии, като комбинира техни практики, за се покрият всички елементи от жизнения цикъл, както и необходимите инженерни техники.

Процесът трябва послужи като рамка при разработката на дадено софтуерно решение. Съобразно принципите на гъвкавите методологии обаче, използващите го екипи трябва да се стремят постоянно да го оптимизират и детайлизират съобразно контекста на техния специфичен проект. Най-добрият източник за такива подобрения идва от срещите за преглед на спринта и особено ретроспекцията, където целият екип допринася с идеите си.

5. ПРИЛОЖЕНИЕ НА ПРОЦЕСА И АНАЛИЗ НА РЕЗУЛТАТИТЕ

За да потвърдим валидността на цялостния гъвкав процес, описан в предходната глава, приложихме го в няколко реални проекта, разработващи софтуерни решения. В практиката можем да имаме изключително голяма разнородност на типовете проекти, техните срокове и участници. Поради тази причина избрахме следните три съвсем различни проекта:

- Проект 1 – дългосрочна разработка на голяма софтуерна платформа, включваща няколко под-екипа. Този проект внедрява процеса още от самото си стартиране
- Проект 2 – дългосрочна разработка на продукт, която е започнала преди години и използва стандартен каскаден процес. Тук направихме преход към дефинирания гъвкав процес
- Проект 3 – екстремен проект за около месец, използващ нови технологии и много кратки срокове. Тук отново гъвкавият процес е въведен от самото начало

За всеки от тях в дисертацията предоставяме следната информация, примери и резултати в резултат от прилагането на дефинирания гъвкав процес:

- Кратко описание – цел и технологии
- Основни параметри – срокове, брой участници, обхват и др.
- Реални примери от изпълнението им – Списък на продукта/спринта, Диаграма на напредъка и др.
- Крайни резултати – брой реализирани версии, брой потребители, подобрения и др.
- Специфики – срещнати характерни проблеми и как процесът се е справил с тях

Тъй като става дума за реални комерсиални проекти, за които има договори за неразкриване на информацията, неизползваме истинските им имена и участници. Също така може се налага на места описаната по-горе информация за прилагането на процеса, която даваме, да бъде съкратена до това, което е допустимо да бъде публично предоставено.

След прилагането на процеса в описаните проекти резултатите, които бяха постигнати, недвусмислено говорят, че той е напълно приложим в практиката, поради следните причини:

- Всички проекти имат доказани резултати и продукти, които са пуснали в експлоатация.
- Проектите 1 и 2, които не са приключили, продължават да го използват и да доставят нови версии на продуктите си на кратки интервали, съответно и ползи за своите клиенти и потребители.
- Наблюдават се множество подобрения в проект 2 след преминаването от използвания дотогава каскаден модел към новия процес.
- Предизвикателен и екстремен проект като третия успя да завърши успешно за толкова кратки срокове.
- Всеки от проектите е разработван в различен контекст, което показва приложимостта на процеса в различни ситуации.

Трябва също да се отбележи, че за всички специфични проблеми, проявили се при проектите, дефинираният гъвкав процес предоставя начини да се справяме с тях или дори ги предотвратява.

6. ПРИНОСИ И БЪДЕЩО РАЗВИТИЕ

В настоящата дисертация направихме цялостен обзор на гъвкавите методологии за разработка на софтуер, основавайки се на официална литература (стандарти, формални ръководства), както и на научни статии и други публикации за тях. Спряхме се на техните основни принципи и практики и направихме сравнителен анализ между различните методологии.

Разгледахме основните модели за управление и качество PMBOK и CMMI, включвайки и преглед на научните разработки, свързани с тях и тяхната еволюция. Обобщихме най-важните принципи и изисквания, които тези модели дефинират, за да може да се твърди, че една софтуерна разработка се справя с всяка от основните четири променливи при ръководенето на проекта – обхват, време, цена и качество.

Въз основа на това, чрез разширяване на тематиката и последвалите анализи и изводи, достигнахме до следните резултати, които считаме за наши приноси в настоящата дисертация:

- Чрез сравнителния анализ в трета глава показахме, че в разгледаните гъвкави методологии съществуват дефинирани принципи и практики, които покриват в много висока степен предписанията на моделите за управление и качество PMBOK и CMMI за всяка от четирите класически променливи – обхват, време, цена и качество (3.1, 3.2, 3.3, 3.4, [b], [c], [d]).

- Към широко използваните четири променливи в управлението на софтуерното производство предложихме, структурирахме и използвахме една нова променлива, наречена „специфики“ (3.5, [a]).

- Чрез обобщения качествен и количествен анализ показахме, че за близо 90% от елементите на PMBOK и CMMI, както и за разгледаните специфики, съществуват практики в гъвкавите методологии, които да ги покриват (3.6, [b], [c], [d]).

- Изградихме нов цялостен гъвкав процес за управление на софтуерни проекти. Този процес съчетава в себе си елементи от различните гъвкави методологии, като чрез своите стъпки и елементи покрива в много висока степен всички аспекти на управлението и качеството, определени от стандартите PMBOK и CMMI. За всички части от този процес показахме с

кои от класическите четири променливи и техните детайлни дефиниции те се справят (4, [b], [c], [d]).

- Приложихме разработения гъвкав процес в три реални софтуерни проекта. За всеки от тях анализирахме и обобщихме данните от използването му. Обобщените резултати категорично показват успешната реализация на проектите, използващи така дефинирания процес (5).

(За всеки от горните приноси с цифра е обозначена съответната част от дисертацията, а с буква – съответната публикация, в която е развит приносът)

Един от основните принципи на гъвкавите методологии е постоянното подобрение. Поради тази причина една от естествените посоки на бъдещо развитие на настоящата разработка е оптимизацията на така дефинирания процес и елиминирането на евентуално показани негови недостатъци при използването му. Това възнамеряваме да осъществим чрез прилагането на процеса във все повече и по-разнородни софтуерни проекти. Заложените в него принципи на ретроспекция ще ни служат като надеждна постоянна основа за усъвършенстването му.

В духа на идеологията на гъвкавите методологии и ставащите все по-актуални напоследък принципи на Lean [30], друга посока на развитие е да интегрираме оперативното ръководене на софтуерните проекти в организацията с нейната мисия, бизнес цели и стратегическо управление. Казано с други думи, така дефинираният гъвкав процес, който е приложим в рамките на един проект, трябва да работи в синхрон с принципите на постоянна оптимизация в цялостната организация, които Lean проповядва.

Добавената от нас пета променлива „специфики“, която представлява характерни проблеми пред софтуерните проекти, може да бъде разгледана в по-широк аспект. Тя може да бъде развита и използвана и в други сфери заедно с класическите четири – обхват, време, цена и качество. Това би повишило ефективността при управление на проекти и извън областта на софтуерните технологии.

ЛИТЕРАТУРА

(Списъкът е оформен в съответствие с БДС 17377-96; от всички източници, посочени в дисертацията, по-долу са само цитираните в автореферата)

[1] Beck, K., Extreme Programming Explained: Embrace Change. Reading, Mass., Addison-Wesley, ISBN 0201616416, 1999

[2] Schwaber, K., Agile Project Management with Scrum, Microsoft Press, ISBN 073561993, 2004

[3] Palmer, S.R. and Felsing, J. M., A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall, 2002

[4] Highsmith, J. A., Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing, 2000

[5] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 4th Edition, ISBN 978-1-933890-51-7, 2008

[6] CMMI Institute, Capability Maturity Model Integration, http://cmmiinstitute.com, 2012

[7] Fitsilis P., Comparing PMBOK and Agile Project Management Software Development Processes, Advances in Computer and Information Sciences and Engineering, Springer Science+Business Media B.V., ISBN 978-1-4020-8740-0, 2008

[8] Murphy Th., Duggan J., Norton D., Prentice B., Plummer D., Landry S., Predicts 2010: Agile and Cloud Impact Application Development Directions, Gartner, 2009

[9] Agile Manifesto, 2001, http://agilemanifesto.org/, 30 March 2013

[10] Ilieva S., Ivanov P., Stefanova E., Analyses of an agile methodology implementation, Euromicro Conference 2004, ISBN 0-7695-2199-1, 2004, p. 326 – 333

[11] Systems Engineering Lecture Notes in Computer Science Volume 6051, 2010, p. 266-280

[12] Abrahamsson P., Salo O., Ronkainen J, Warsta J., Agile Software Development Methods Review and Analysis, ISBN 951-38-6009-4, 2002

[13] Beck K., Fowler M., Planning eXtreme programming, Addison-Wesley, ISBN 0-201-71091-9, 2001

[14] Jeffries R., Anderson A., Hendrickson C., eXtreme Programming Installed, Pearson Education, ISBN 201-70842-6, 2001

[15] Schwaber K., Beedle M., Agile Software Development with Scrum, ISBN 0-130-67634-9, 2001

[16] Anderson D. J., Feature-Driven Development, Microsoft Corporation, October 2004

[17] Paetsch F., Eberlein A., Maurer F., Requirements engineering and agile software development, Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE 2003, ISBN 0-7695-1963-6, 2003, p. 308-313

[18] Holtzman, J., Getting Up to Standard, PM Network, December 1999

[19] Sharp H., Robinson H., Petre M., The role of physical artefacts in agile software development: Two complementary perspectives, Interacting with Computers, Volume 21, Issues 1–2, January 2009, p. 108–116

[20] Sutherland J., Inventing and Reinventing SCRUM in Five Companies, Cutter IT Journal, 2001

[21] TechTarget – (http://media.techtarget.com/searchSoftwareQuality/downloads/Software_Project_Manager_Agility_CH05.pdf), Chapter 5 - Scope Management, May 2012

[22] Cohn M., Selecting the Right Iteration Length, InformIT Network, 2004

[23] Jira – система за управление на проекти, www.atlassian.com/JIRA

[24] Trac – система за управление на проекти, http://trac.edgewall.org/

[25] Anderson D.J., Kanban: Successful Evolutionary Change for Your Technology, Blue Hole Press, ISBN 978-954-92934-1-8, 2010

[26] Ackerson D., 2008, Successful Teams Are Small And Dedicated, http://www.agileweboperations.com/successful-teams-are-small-and-dedicated, 30 March 2013

[27] Software Engineering Institute, CMMI for Development, Version 1.3, 2010

[28] Kettunen P., Laanti M, "How to Steer an Embedded Software Project: Tactics for Selecting Agile Software Process Models ", ICAM 2005 International conference on agility, Helsinki, July 27-28, 2005

[29] Todorov, N. and Eskenazi, A. Specifics in applying agile software methodologies in portal solutions. Serdica Journal of Computing, Volume 5, Number 1, 2011

[30] Monden Y., Toyota Production System, An Integrated Approach to Just-In-Time, Third edition, Norcross, GA: Engineering & Management Press, ISBN 0-412-83930-X, 1998

ПУБЛИКАЦИИ

[a] Todorov, N., Eskenazi, A., Specifics in applying agile software methodologies in portal solutions. Serdica Journal of Computing, Volume 5, Number 1, 2011

[b] Kovatcheva T., Todorov N., Optimizing software development process: A case study for integrated Agile-CMMI process model, EUROCON 2011, Lisbon, ISBN 978-1-4244-7486-8, 2011

[c] Todorov N., Comparing Agile and PMBOK – Time Management, ATINER 2012, Athens, ISBN 978-960-9549-60-8, 2012, p. 137-152

[d] Todorov N., Comparing Agile and PMBOK – Cost Management, 42nd Spring Conference of the Union of Bulgarian Mathematicians, Borovetz, 2013

ДОКЛАДИ

- Доклад на годишната отчетна сесия на ИМИ 2010
- Доклад на тема „Сравнение между гъвкави методологии и PMBOK® стандарта в областта на управление на обхвата“, PM Day 2011, София, 2011
- Доклад на семинар на секция Софтуерни технологии ИМИ, Април 2012
- Доклад на годишната отчетна сесия на ИМИ 2012
- Доклад на тема „Три управленски капана преминавайки към гъвкави методологии“ на редовна среща на PMI Bulgaria Chapter, София, 2012
- Доклад на тема „Подобряване на качеството на реализация на софтуера“ на годишна конференция ISTA 2012, София, 2012