ПЛАНИРАНЕ НА ТРУДА НА ПРОГРАМИСТИТЕ ПРИ ИЗГРАЖДАНЕ НА ИНФОРМАЦИОННИ СИСТЕМИ

Моника Цанева*

1. ВЪВЕДЕНИЕ

Съгласно проучване на Forrester Research една от основните тенденции в развитие на бизнес софтуера е: „Оформя се нова визия за разпределение на ИТ разходите, насочена към придобиване на програмни продукти, които могат да се внедрят в кратки срокове и при ниски разходи, а същевременно да съдействат за постигането на максимален икономически ефект" [15]. Същевременно, Институтът за управление на проекти посочва, че „Най-важната задача на един проект е да зададе реалистични очаквания. Нереалистичните очаквания, базирани на неточни изчисления, са най-съществената причина за провал на софтуерните проекти" [14]

Основната част от изследванията в областта на управлението на софтуерни проекти е в посока как да бъде намалено времето за разработка [1], но не и как да оценим трудоемкостта на задачите или тяхната продължителност. Настоящата разработка си поставя за цел да предложи методика за оценка на плановата трудоемкост при системното и приложно програмиране на бизнес софтуера. За целта е необходимо:

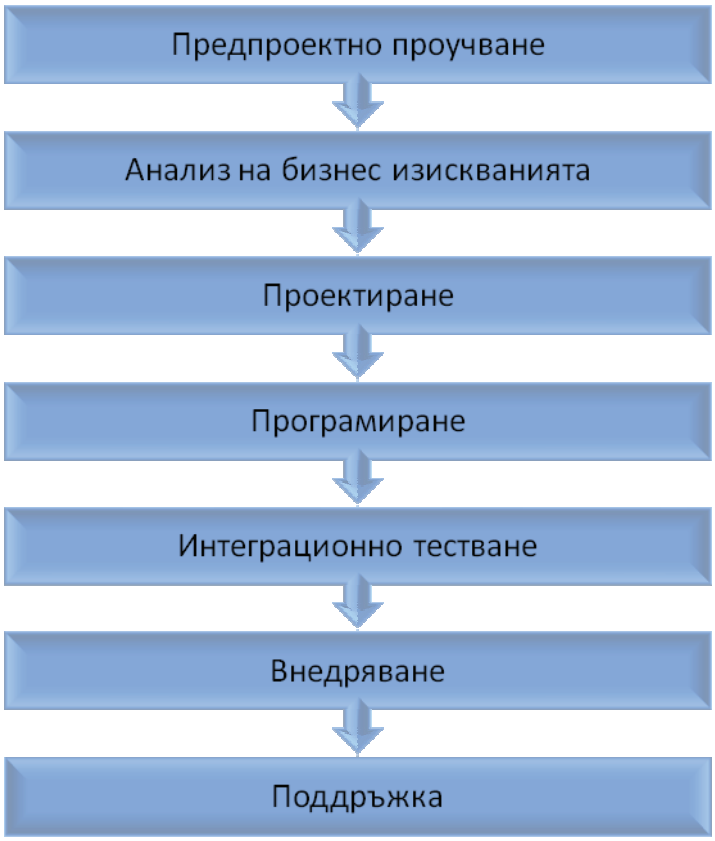
- да бъде анализиран процесът на разработка на софтуер като последователност от дейности;
- да бъдат дефинирани основните групи фактори, които оказват влияние на трудоемкостта на всяка от дейностите, съставящи процеса по разработка на софтуер;
- да бъде анализирано и оценено влиянието на най-съществените фактори върху трудоемкостта на разработката на софтуер както самостоятелно, така и във взаимната им връзка.

Като краен резултат е разработена методика за оценка на планираната трудоемкост и продължителността на разработката на софтуерни компоненти на бизнес приложения въз основа на дефинираните фактори.

* Моника Цанева е доктор по икономика, главен асистент в катедра "Информационни технологии и комуникации" в УНСС; тел: 089 876 62 89, e-mail: monika_tzaneva@yahoo.com.

2. ХАРАКТЕРИСТИКА НА ПРОЦЕСА ПО РАЗРАБОТКА НА СОФТУЕР

Жизненият цикъл на софтуерен проект обхваща следните основни фази (Фиг. 1):



Фиг. 1. Жизнен цикъл на софтуерен проект

Традиционните методологии за разработка на софтуер включват 3 основни модела – последователен, инкрементален и еволюционен [12].

В съответствие с неговото наименование, последователният подход изпълнява последователно всички фази от жизнения цикъл на проекта: т.е. всяка фаза започва след приключването на предходната, като се допуска връщане към предходни фази с цел отстраняване на грешки.

Инкременталният подход към разработката предполага, че фазите на проектиране, програмиране, тестване и внедряване се разпаралеляват винаги когато е възможно, при което съществено се намалява цялостният срок за изпълнение на проекта.

Еволюционният подход се състои от множество последователни жизнени цикли с припокриване на фазите по експлоатация, разработка и поддръжка.

През последните две десетилетия все по-широко се прилага ускорената методология за разработка на приложения (RAD- Rapid Application Development), която се осно-

вава на идеята, че приложенията трябва да се създават бързо и икономично, с минимум изменения между анализа и окончателното предаване. Характерно за ускорената методология на разработка е, че фазите на анализ и проектиране, от една страна, и на програмиране, от друга страна, се изпълняват циклично, вместо последователно.

Екстремните методологии за разработка на софтуер се основават на принципно различен подход. Тяхна основна презумпция е, че изменения в изискванията към софтуера могат да настъпят във всяка от фазите на разработка. Тяхната област на приложение все още е твърде тясна, а натрупаният опит в управлението на проекти, които се планират и изпълняват в съответствие с екстремния подход, е минимален. Екстремните методологии реализират подхода „проектиране чрез програмиране", при което отделните фази на разработката не са ясно разграничени и не могат да бъдат планирани поотделно. По тези причини изчисляването на трудоемкостта на разработката на софтуерни модули съгласно принципите на екстремните методологии остават извън обхвата на настоящия материал.

Независимо от възприетия подход към жизнения цикъл на разработвания проект фазата „Анализ на бизнес изискванията" винаги предшества фази „Проектиране" и „Програмиране" и завършва с изготвянето и приемането от всички заинтересовани лица на:

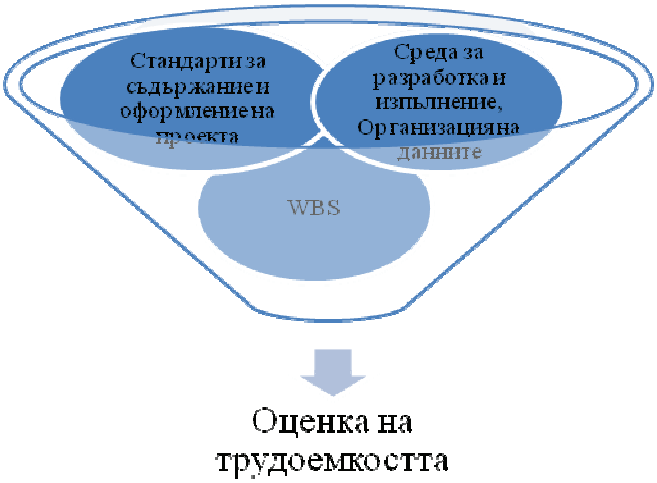
- пълна документация на обхвата и целите на проекта,
- подробна функционална спецификация на бизнес изискванията,
- спецификация на технологичните изисквания към проекта, вкл. операционна система, програмна(и) среда(и) за реализация, инфраструктура/и за изпълнение на СУБД,
- завършен план на проекта,
- определен екип на проекта,
- приета работна програма на проекта,
- изчислено предварително (грубо) разписание на проекта.

От това следва, че предварителното разписание на проекта трябва да се изчисли при условие, че са известни (Фиг. 2):

- Декомпозицията на задачите (WBS – Work Breakdown Structure), т.е. има обща представа за обема и сложността на всяка от тях.
- Програмната среда за реализация на модулите, при което може да се оцени доколко тя е подходяща за разработка на задачите от WBS.
- Технологията за организация на съхраняваните и обработвани данни (файлова система, база данни и СУБД, склад за данни).
- Инфраструктурата за изпълнение на създаваното приложение.
- Стандарти, процедури и техники за създаване на съдържанието и оформлението на документите от детайлния проект.

За да се получи гъвкаво структурирана система, създадена при ефективната организация на работата, е препоръчително като отделни задачи да се обособяват всички елементи на системата, които могат да бъдат създадени, тествани и използвани самостоятелно и то по възможност със средствата на една програмна среда. Така например разработката на модул за обработка на големи обеми от данни, който ще бъде реализиран с програмните средства на СУБД, и създаването на потреби-

телския интерфейс към него в среда за програмиране от 4-то поколение (или друга извън СУБД) би трябвало да се раздели на две задачи, които са по-лесно управляеми и могат дори да се изпълняват паралелно. В случай, че процедурата на БД, която ще извършва обработката, е съставена от сравнително обособено стъпки, които могат да се реализират като подпрограми и да се създават паралелно, то тези подпрограми (или част от тях) също могат да се обособят като отделни задачи. Подобен подход не е наложително да бъде приложен към проектирането, тъй като този процес по принцип изисква по-общ поглед върху функционалността на системата. Дори е препоръчително взаимосвързаните модули да се проектират заедно с цел да се синхронизират по-добре. Ако един проект е добре структуриран, той може да обхваща няколко задачи за разработка на софтуерни компоненти.



Фиг. 2. Условия, при които се планира трудоемкостта на системното и приложното програмиране

Обикновено в предварителното разписание не се отчита разпределението на задачите на участниците в екипа (виж т. 5), поради което на този етап квалификацията на програмистите не може да бъде фактор за оценка на плановата трудоемкост.

На фаза „Проектиране" се определя как ще бъде структурирана системата (нейната архитектура), как ще е организиран потребителският интерфейс (менюта, форми, справки) и как ще се реализира взаимодействието на приложението с други бизнес приложения (собствени за потребителя или външни).

Фазата включва следните основни етапи:

- Организационно проектиране на развитие на екипа – формират се екипи за създаване на обзор на проекта и за проектиране на отделните групи задачи.
- Създаване на резюме(та) на проекта. Тези резюмета определят архитектурата на системата, логическият и физическият модели на данните, разпределението на бизнес логиката, управлението и представянето на потребителския интерфейс между отделните слоеве на системния и приложния софтуер.

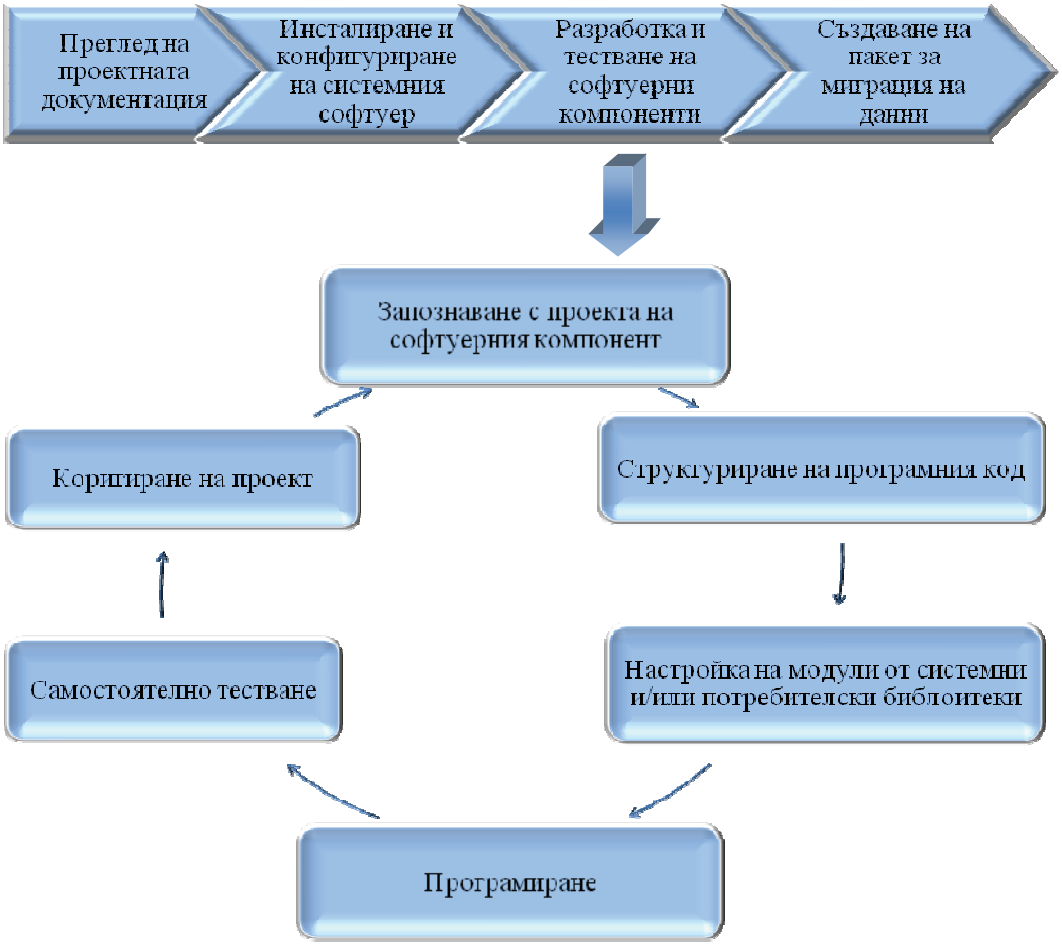
- Преглед и приемане на резюмето(тата) на проекта от екипа за преглед на проекта с цел интеграция и консистентност на системата и осигуряване на спазването на възприетите стандарти, процедури и техники.
- Изчисляване на точното разписание на проекта (вкл. с назначаване на ресурсите към всяка задача).
- Създаване на документите на детайлния проект. Тези документи се създават от всеки отделен член на екипа по детайлно проектиране и се съгласуват с всички останали членове на екипа.

По принцип фаза „Проектиране" завършва с одобрена проектна документация (пълен пакет), вкл. резюмета на проекта и детайлни проекти, предварителен план за тестване (може да се съдържа в детайлните проекти), но при прилагане на ускорена методология за управление на софтуерния проект програмната реализация на всеки модул на системата започва след приемането на неговия детайлен проект, без да се изчаква завършването и приемането на всички останали проектни решения .

На фаза „Програмиране" всяко проектно решение се реализира на избраните платформи със средствата на избраните СУБД и програмна среда. Фазата включва (Фиг. 3):

- Преглед на разписанието, работната програма и проектните документи с екипа. Необходимо е целта и общата функционалност на системата, изискванията към нея, технологията на нейната бъдеща експлоатация, критичните точки и датите за предаване подробно да бъдат разяснени на всички участници в екипа. Този подход гарантира, че те ще се отнесат с разбиране към възложената им работа, което ще повиши тяхната мотивация (виж. т. 4.4).
- Инсталиране и конфигуриране на програмната среда за разработка, на инфраструктурата за изпълнение, на схемата на базата данни, а при необходимост и попълване на служебните номенклатури.
- Реализация на програмното осигуряване на основата на детайлния проект и според вътрешните стандарти за разработка.
- Тестване на отделните програмни компоненти, включително преглед на програмния код за съответствие на стандартите и документиране на тестването.
- Създаване на пакет за миграция на данните от старата система.
- Фаза „Програмиране" завършва с приемане от всички заинтересовани лица на разработените програмни компоненти и на документираните резултати от тестването им, както и на пълния пакет за миграция на данни.
- Част от горепосочените задачи се изпълняват еднократно в началото или в края на фаза „Програмиране", а други имат цикличен характер (Фиг. 3).

При планиране на плановата трудоемкост на задачите във фаза „Програмиране" трябва да се отчитат спецификите и начинът на изпълнение на всяка една от посочените дейности, особено тези, които има цикличен характер, а не само „чистото програмиране". Разработената за целта методика не отчита спецификата на дейностите по локализация на софтуер, какъвто е случаят с внедряване на ERP системи [2]



Фиг. 3. Основни дейности по разработка на софтуерни компоненти

3. ЗНАЧИМОСТ НА ИЗЧИСЛЯВАНЕТО НА ТРУДОЕМКОСТТА НА РАЗРАБОТКАТА НА СОФТУЕР ПРИ ПЛАНИРАНЕ НА СОФТУЕРНИ ПРОЕКТИ

Задачите и дейностите за реализация на софтуерни проекти се характеризират с два взаимосвързани количествени измерители:

- Трудоемкост на изпълнението (Усилие – Effort);
- Продължителност на работата по задачата (Duration).

Трудоемкостта на задачата показва колко работа се изисква за нейното изпълнение и се измерва в работни човеко часове (човекодни, човекоседмици, човекомесеци и т.н).

Продължителността на задачата показва колко време е необходимо за нейното изпълнение. Измерва се в работни или календарни (включват се неработни и празнични) часове (дни, седмици и т.н).

Трудоемкостта е „най-чистият" количествен измерител на задачите, тъй като не се влияе нито от количеството човешки ресурси, които са им разпределени, нито от календара на проекта. Това е особено съществено поради факта, че при работа по един софтуерен компонент производителността на труда не е пропорционална на количеството назначени ресурси (двама програмисти няма да реализират този компонент два пъти по-бързо, отколкото би го направил всеки един от тях поотделно).

Изчисляването на трудоемкостта на разработката на програмните модули е един от аспектите на планиране на софтуерни проекти, които предлагат най-много предизвикателства пред ръководителя на проекта и неговия екип. Реалистичното изчисляване на трудоемкостта на отделните задачи е ключова предпоставка за следващото планиране на разходите и за бюджетиране на проекта. Прецизното изчисляване на трудоемкостта е важна предпоставка както за поддържане на морала на екипа, така и на коректните взаимоотношения с възложителите и с бъдещите потребители на софтуера, тъй като ясно регламентира очакванията на всички заинтересовани лица.

Изчисляването на трудоемкостта е сложна задача, тъй като зависи от множество фактори, част от които са неизвестни по време на планирането (рискови фактори), вкл. изменения в обхвата и функционалността на разработваните модули, наличие или отсъствия на планираните човешки ресурси по време на изпълнение на разпределените им задачи, неразбирателства и грешки, неочаквани събития и т.н. При извършване на реалистични изчисления на трудоемкостта следва да се отчита фактът, че никой човек не е продуктивен през 100% от работното си време.

В литературата са описани различни техники за планиране на трудоемкостта на разработката на софтуер, като най-разпространените се основават на исторически данни и натрупан опит при изпълнение на задачи с подобен обхват. Историческите данни обхващат както изчисленията за продължителност и трудоемкост на реализацията на разнообразни задачи по софтуерни проекти, така и кадровите характеристики на наличните човешки ресурси – проектанти и програмисти. Разработените модели, основани на софтуерни метрики [12], бяха приложими в епохата на процедурното програмиране, тъй като се основават на метрики като „брой програмни редове (в хиляди)", големина на база данни и др., които не са приложими в епохата на програмиране в среди от 4-то поколение при прилагане на многократно използваеми програмни конструкции и генератори на софтуерни компоненти от различен вид. Споменава се метриката „сложност", но без практически приложим подход за нейното изчисляване.

При отсъствие на подходящи исторически данни обикновено се препоръчва да се ползва експертно мнение на специалисти, които са ръководили или поне са консултирали проекти, подобни на разработвания.

Предлаганата методика дава възможност да се изчисли плановата трудоемкост на основните задачи, които съставят фаза „Разработка" на бизнес приложенията. Тя може да се прилага както при отсъствие на исторически данни за сходни проекти и/или на подходящо експертно мнение, така и за сравнение и/или корекции при тяхното наличие.

4. ФАКТОРИ, ОПРЕДЕЛЯЩИ ТРУДОЕМКОСТТА НА ЗАДАЧИТЕ ПО РАЗРАБОТКА НА СОФТУЕР

Факторите, които оказват влияние върху планираната трудоемкост на разработката на софтуерни модули, са разгледани в последователност, в която възникват и се проявяват, а не според своята тежест.

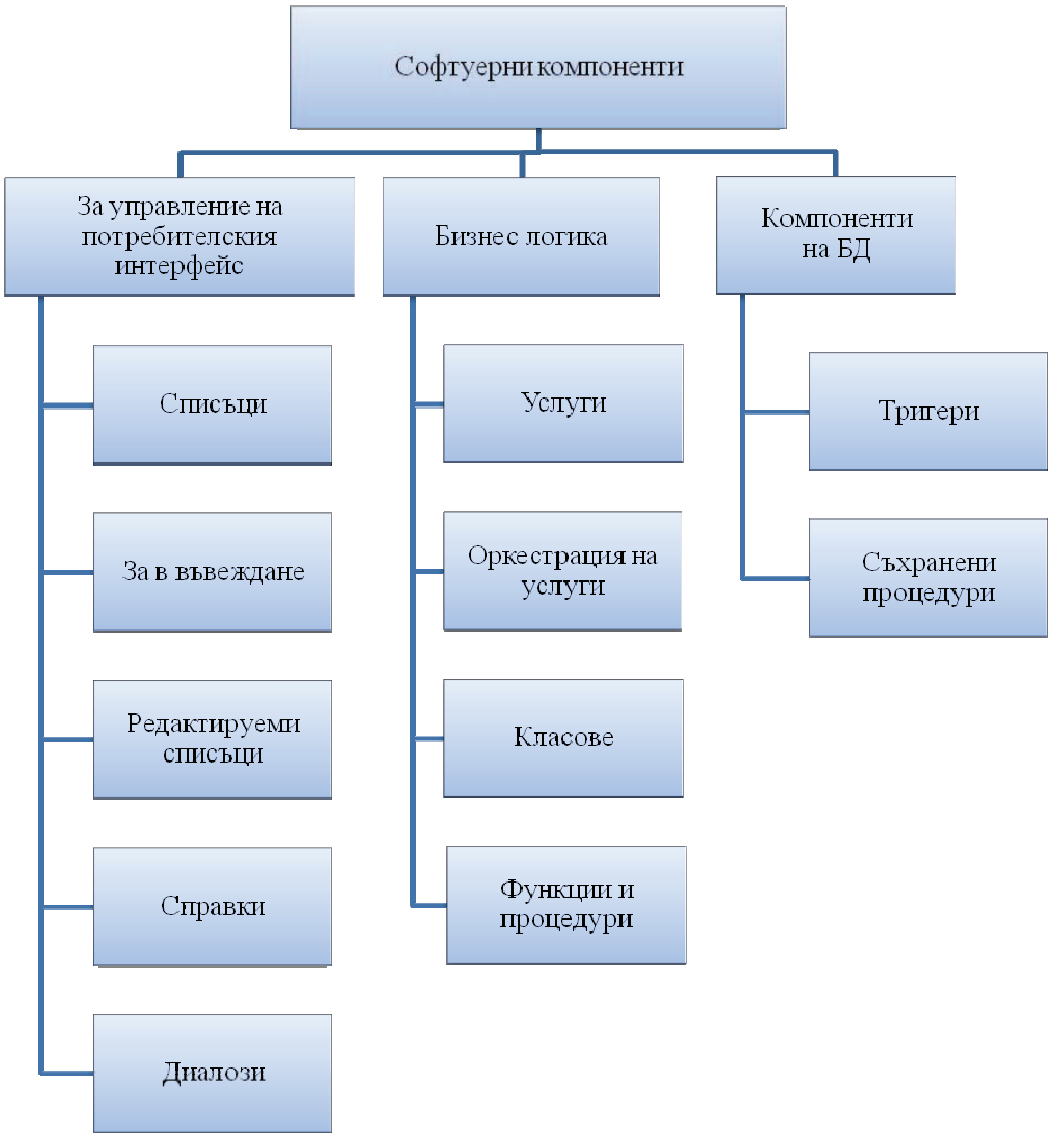
4.1. Вид и сложност на разработвания софтуерен компонент

За да се направи приблизителна оценка на сложността на софтуерните компоненти, които изграждат бизнес системи, е необходимо тези компоненти да бъдат класифицирани в зависимост от архитектурата на изграждaната система, средствата за тяхното създаване, изпълнение и тестване (Фиг. 4).

Типичните бизнес информационни системи включват три основни групи функции:

- *Потребителски интерфейс* – функции, реализиращи взаимодействието на ИС с потребителя.
- *Бизнес-логика* – функции по обработка на данните в съответствие със специфичните правила, характерни за съответната приложна област.
- Функции по съхранението на данните и извличане на конкретни данни от бази данни или отделни файлове.

През последните години се обособява специфична група системи, наречена Бизнес Интелигентни Системи, която също притежава йерархично организирана архитектура. Специфично за Бизнес Интелигентните Системи е, че в основата им стои склад за данни, вместо база от данни, както и че акцентът в тези системи пада на представянето на обработените данни във форма, подходяща за анализи [6]. В сравнение с останалите бизнес приложения, този вид системи имат по-различни изисквания към средата за разработка.



Фиг. 4. Класификация на основните типове софтуерни компоненти

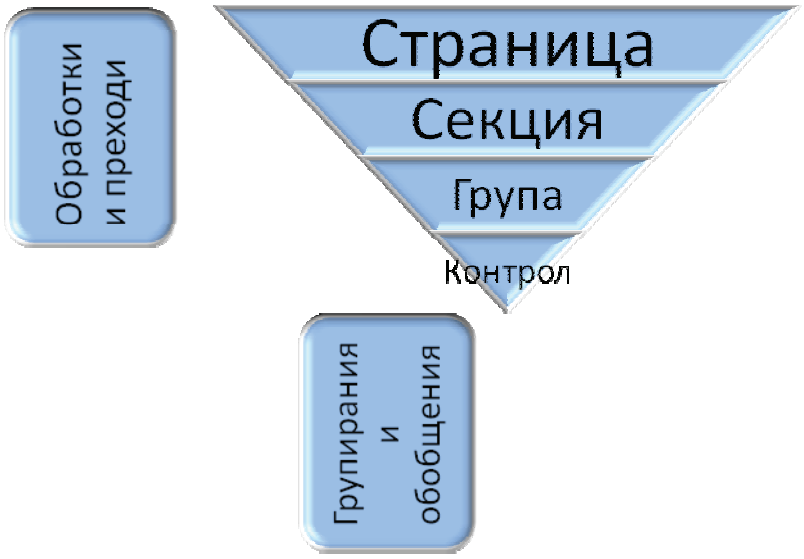
4.1.1. Интерфейсни софтуерни компоненти

Като интерфейсен може да бъде определен софтуерен компонент на компютърната система, който може да бъде видян (чут или възприет по друг начин) от потребителя, и команди и механизми, които потребителят използва, за да контролира действията на системата и нейните входни данни [16].

Средите за програмиране, които са популярни през последните няколко десетилетия, изцяло поемат грижата за визуализацията на интерфейсните елементи, като ос-

тавят на приложението само грижата за тяхното управление. В зависимост от начина на своето управление интерфейсните софтуерни компоненти на бизнес информационните системи могат да се класифицират условно в следните видове [5]:

1. *Форми-списъци* – предназначени са за търсене на елемент или извеждане на списък от елементи по зададен набор от критерии за търсене или филтриране на списъка (филтър). Типичната структура на формите-списък включва 2 секции – секция филтър, в която се въвеждат критериите за търсене на данни и секция-списък, в която се извежда резултатът от търсенето (филтрирането). В тези форми могат да се стартират операции по добавяне на нови елементи или редактиране и изтриване на данните за избран съществуващ елемент от списъка. Самите операции за манипулиране на данните се извършват в специални форми за въвеждане. Допълнителна функционалност, която може да се реализира включва операции със самия филтър (изчистване, запис, извличане и т.н). Специфичен случай на форма-списък са дървовидните списъци (tree view), които нямат специализирана секция за филтър (и съответни операции с нея), а организират достъпа до данните въз основа на тяхната естествена йерархия.
2. *Форми за въвеждане* – предназначени са за въвеждане и редактиране на данни за избран от списъка елемент. В структурата на формите за въвеждане се включват (Фиг. 5) – Tab страници, които се състоят секции. Секциите се състоят от групи, а групите от различни по вид контроли. Отделните елементи на формата (страници, секции, групи и контроли) могат да бъдат функционално или логически свързани или независими. Основните операции във формите за въвеждане са свързани с извличане и запис на данните. Често се налага реализацията и на допълнителни операции по печат на различни бланки, както и преходи към други форми.
3. *Форми-редактируеми списъци* – структурата им е аналогична на формите-списък, а функционалността им комбинира тази на форми за въвеждане и на форми-списък. Редактируемите списъци позволяват директна редакция на данните (без отваряне на специална форма за въвеждане), които са изведени в списъка и едновременно с това осигуряват изпълнение на операции с целия списък. За разлика от формите за въвеждане, редактируемите форми списък нямат йерархична структура (страници, секции и т.н), а се състоят само от контроли.
4. *Форми-справки* – като структура и функции са подобни на формите-списъци т.е. са предназначени за извеждане и отпечатване на списък с данни, филтриран по зададени критерии, но позволяват групиране и обобщаване на извлечените данни по различни признаци.
5. *Диалогови прозорци* – форми за въвеждане, които не позволяват работа с главното меню и другите отворени прозорци в работната област до затварянето им. Те са предназначени за въвеждане на критични параметри и данни, необходими за правилното функциониране на системата. Структурата и функционалността им са значително по-прости в сравнение с формите за въвеждане – нямат страници, а само ограничен брой секции и контроли. Операциите с този вид форми обикновено се свеждат до стартиране или отказ на дадена обработка, като не се допускат преходи към други форми.



Фиг. 5. Йерархия на интерфейсните елементи

Като цяло сложността на интерфейсните софтуерни компоненти зависи от факторите, показани на фиг. 6:

- Брой и разнообразие на съставящите ги елементи (страници, секции, групи и контроли), групировъчни признаци и обобщения, както и от възможните режими на работа на елементите.
- Брой и сложност на взаимните зависимости между елементите (напр. наличие на страници, секции и т.н., чието наличие или чието съдържание зависи от наличието и/или съдържанието на други страници, секции или контроли).
- Брой и разнообразие на операциите, които могат да се изпълняват с даден интерфейсен елемент
- Структура на данните, които се извличат, редактират и записват с помощта на формата. В колкото повече таблици от базата данни са разположени данните на формата (напр. структури master – detail на две или повече нива), толкова по-сложна е тя. Следва да се отчита и вида на връзките между таблиците (пряк или outer join).
- Необходимост от формален и логически контрол на въвежданите данни. По принцип в съвременните среди за разработка до голяма степен е автоматизиран формалният контрол на данните (по тип, по размер, по принадлежност към определен домен и т.н), докато логическият контрол трябва да се програмира ръчно. Една възможност за опростяване на интерфейсните модули е реализация на логическия контрол със средствата на СУБД (тригери и съхранение процедури), при което задачата на интерфейсните модули се свежда до прихващане и обработка на върнатите от процедурите грешки.

Фиг. 6. Фактори за определяне на сложността на интерфейсните компоненти

4.1.2. Софтуерни компоненти за реализация на бизнес логика

В тази група попадат само функциите, процедурите, класовете и услугите, които са предназначени за реализация на бизнес логика, а не тези, които изграждат потребителския интерфейс.

Процедурите и функциите са обособен програмен модул (част от цялостна програма), който реализира отделна конкретна задача. От гледна точка на трудоемкостта на своето създаване процедурите и функциите се разглеждат като синоними. Техните основни компоненти са:

- Тяло – програмен код, който се изпълнява при тяхното извикване;
- Параметри, които им се предават от точката на извикване;
- Резултат, който се връща към точката на извикване (незадължителен елемент).

Класът е кохерентен пакет, който се състои от конкретен тип метаданни и описва правилата на тяхното поведение. Класът се състои от структура данни и интерфейс към нея. Структурата от данни описва как се разпределят данните по полета (наречени още членове данни, свойства или атрибути) във всеки екземпляр на класа, докато интерфейсът описва как може да се взаимодейства с класа и с неговите представители.

Архитектурата, ориентирана към услуги (Service Oriented Architecture SOA), е стил, който ръководи всички аспекти на създаване и използване на бизнес дейности, пакетирани като услуги, както и дефинирането и осигуряването на съответна

инфраструктура, позволяваща обмен на данни между хетерогенни приложения и интегрирането им в единен бизнес-процес, който е слабо свързан с операционната система и използваната среда за програмиране [1]. Тази архитектура представя модел, в който функционалността на приложението е декомпозирана на обособени елементи (наречени услуги), които могат:

- да бъдат разпределени в рамките на цяла една мрежа от компютри, като обменят помежду си данни,
- да бъдат оркестрирани (координирани и комбинирани) в единен бизнес процес,
- да бъдат използвани многократно за изграждане на различни бизнес приложения.

От гледна точка на трудоемкостта на създаване е съществено, че в сравнение с класовете, които бяха основните градивни единици на приложенията, съгласно обектно ориентираната методология, услугите са между 10 и 1000 пъти по-големи и включват в себе си класове, функции и други програмни модули. В процеса на оркестрация услугите се свързват помежду си в нейерархични структури. Оркестрацията се извършва с помощта на специализиран инструментариум, който разполага с пълен списък на услугите, на техните характеристики и на средства за запис на проектните решения, които генерират код, директно използваем в процеса на изпълнение. За своето изпълнение услугите ползват „безопасни" инфраструктури [8], които управляват заделянето и освобождаването на паметта, динамичното свързване на компоненти, стандартизират типовете данни, което е важна предпоставка за намаляване на трудоемкостта на разработката

Основните типове услуги, които съставят бизнес приложенията биват:

- За предоставяне на данни (Data),
- За обработка на данни (Process),
- За реализация на бизнес логика (Business),
- Интеграционни (Integration),
- За управление на потребителския интерфейс.

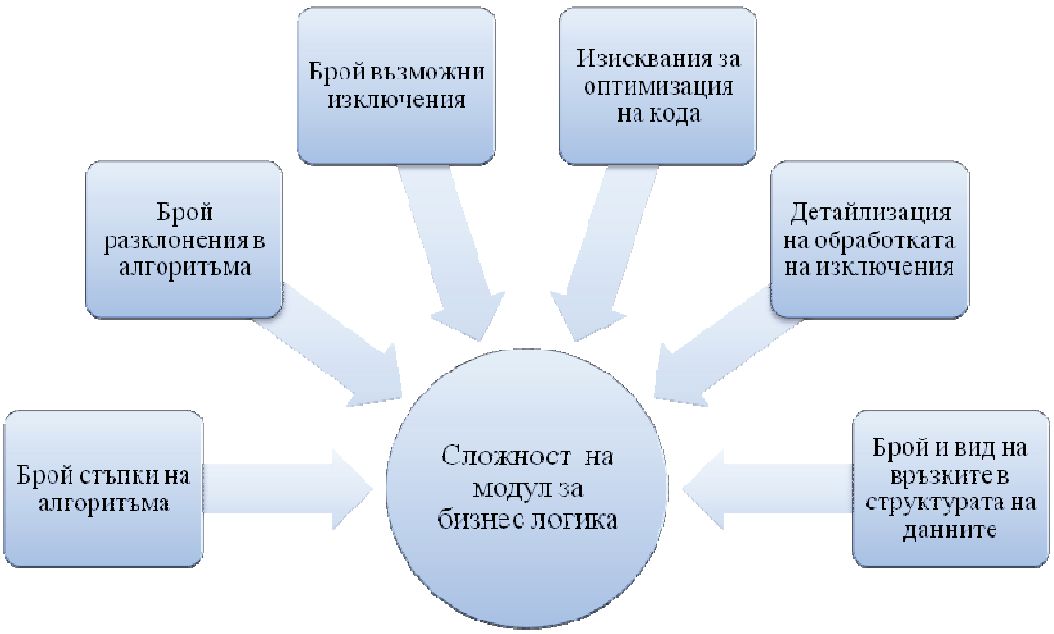
За целите на настоящата разработка услугите за управление на потребителския интерфейс са изключени от анализа, тъй като трудоемкостта на създаването им не се отличава съществено от трудоемкостта на създаване на интерфейсни компоненти, които работят в среда на контролирано изпълнение, без да са организирани като услуги.

Трудоемкостта на създаване на софтуерните компоненти за реализация на бизнес логика зависи предимно от следните основни фактори (Фиг. 7):

- Сложност на реализирани алгоритъм, по който се обработват данните. Не е от съществено значение дали тези данни се предават като параметър или са част от софтуерния компонент, реализирана под формата на член-данни (в класовете) или на локална структура в обхвата на процедура или функция. Сложността на алгоритъм може да бъде оценена на база на броя стъпки на обработката, броя на разклоненията в него, броя на потенциално възможните изключения (грешки), които могат да възникнат и следва да се обработят.
- Изисквания за оптимизация на кода по критерии бързодействие или обем на необходимата памет за изпълнение. Строгите изисквания за оптимизация налагат използване на по-сложни синтактични конструкции, което увеличава както

времето за „писане”, така и времето за тестване на модула. В редица случаи, особено когато се работи в среда, която не контролира изпълнението, се стига до цялостна преработка на модула, за да се постигне изискваното бързодействие.

- Изисквания към начина на обработка на евентуални грешки и изключения (exception handling). Колкото по-детайлно е необходимо да се анализират и обработват възникналите изключения, толкова по-трудоемка става реализацията на програмния модул. Пропорционално се увеличава и времето за тестване.
- Структура на обработваните данни. Най-малко усилия изисква обработката на линейно структурирани данни. Трудоемкостта на разработката се увеличава пропорционално на броя взаимни връзки между обработваните данни (отношения master-detail)



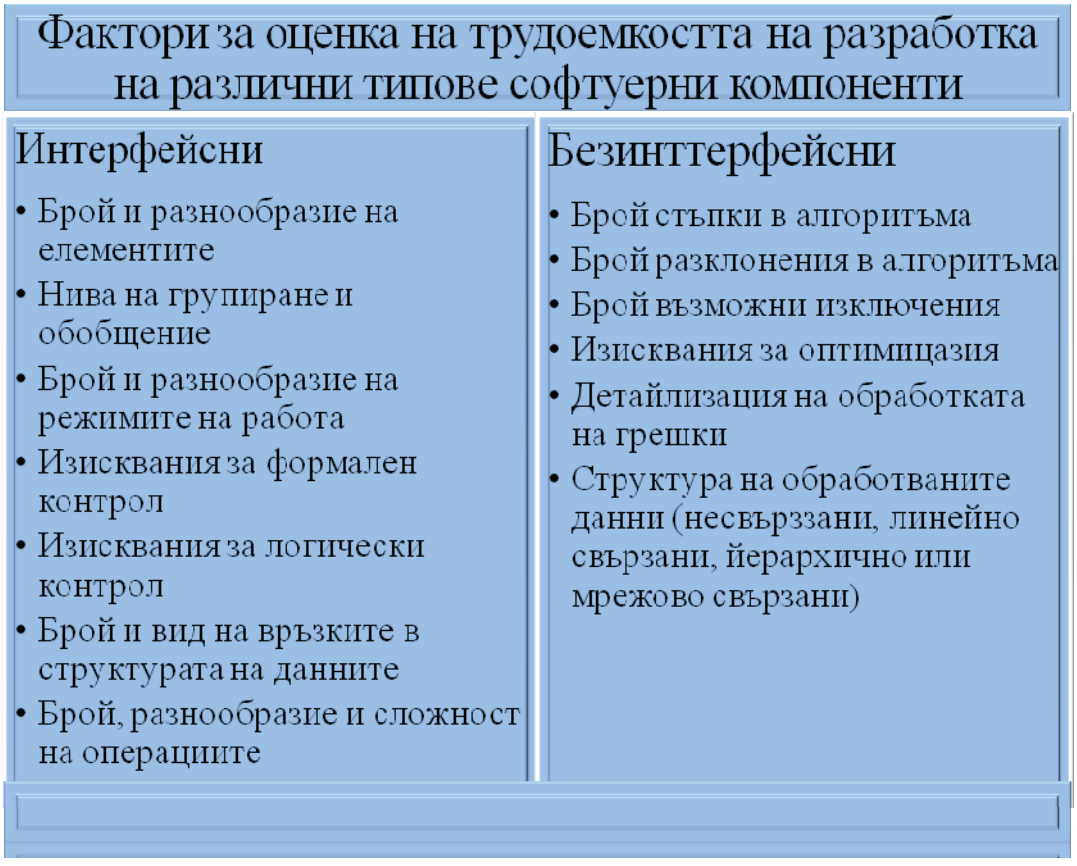
Фиг. 7. Фактори за определяне на сложността на софтуерните компоненти за реализация на бизнес логика

4.1.3. Съхранени процедури и тригери на БД

Съхранената процедура е програмен модул, който се съхранява в релационна база данни и се използва от приложенията, които осъществяват достъп до нея [4]. Типичната им употреба е за голяма и сложна обработка на данни, която изисква изпълнение на множество SQL оператори, за валидация на данни, която е интегрирана в базата, за контрол на достъпа за консолидиране и централизиране на бизнес логика. От гледна точка на трудоемкостта на създаване съхранените процедури приличат на потребителските функции, тъй като особеностите им се проявяват главно в начина на извикване и вида на връщания резултат.

Тригер на базата данни е процедурен код, който се изпълнява автоматично в отговор на определени събития, настъпили в дадена таблица или база данни. Използват се предимно за организация на потребителски журнали на измененията в данните, за ограничаване на достъпа до определени данни и т.н. По своята сложност и техника на създаване тригерите са близки до съхранените процедури и потребителските функции. Техните свойства (не приемат параметри и аргументи, не могат да завършват или да отменят транзакции, освен ако не работят в автономни такива) оказват влияние по-скоро върху трудоемкостта на тестване, отколкото на създаване. Единствено особеността им да предизвикват мутации на таблици увеличава трудоемкостта на реализация в сравнение със съхранените процедури.

Резултатите от направения анализ на факторите, които указват влияние върху трудоемкостта на разработка на различните видове софтуерни компоненти, могат да бъдат обобщени по следния начин (Фиг. 8):



Фиг. 8. Основни фактори за оценка на трудоемкостта на различни типове софтуерни компоненти

При равни други условия (разгледани в следващите точки) подходът към оценка на трудоемкостта на разработката на различни видове софтуерни компоненти зависи най-вече от това дали компонентът е интерфейсен или безинтерфейсен. Трудоемкостта за разработка на интерфейсни компоненти може да бъде оценена доста по-точно, тъй като структурата им е ясно дефинирана, докато сложността на безинтерфейсните компоненти е до голяма степен относителна.

4.2. Функционалност на използваната програмна среда за разработка и платформа за изпълнение

Различните поколения и типове среди и инфраструктури в различна степен автоматизират процеса на програмиране, предоставяйки или не на програмиста разнообразен набор от готови или полуготови градивни блокове. Поради това, средата за разработка и изпълнение на създавания софтуер е от изключително важно значение за трудоемкостта на разработката. Наред със сложността на изграждания модул тя предопределя обема на работата, която трябва да бъде свършена

4.2.1. Наличие на адекватен набор синтактични конструкции, богатство на стандартни и потребителски библиотеки според вида на създавания софтуер

Основните конструкции, които автоматизират приложното програмиране, могат да бъдат обобщени в следните основни групи:

- Синтактични конструкции от ниско ниво;
- Синтактични конструкции от високо ниво;
- Обща система от типове;
- Библиотека(и) със стандартни контроли за организация на потребителския интерфейс и редактор на ресурси (евентуално включващи стандартни обекти за създаване и манипулиране на 2D и 3D графика);
- Библиотека(и) стандартни контроли за извличане, манипулиране и представяне на данни;
- Библиотеки за създаване и обработка на документи;
- Библиотеки за управление на комуникацията между софтуерни компоненти;
- Разработени потребителски библиотеки за многократно използване;
- Инструментариум за реализация на бизнес интелигентност.

Влиянието на отделните групи конструкции е различно в зависимост от типа на разработвания програмен модул (виж. фиг 4 и т. 4.1). Наличието на синтактични конструкции от високо ниво, обща система от типове и наличието на подходящи потребителски библиотеки е съществено за всички видове софтуер; библиотеките със стандартни контроли за организация на потребителския интерфейс, редакторите на ресурси и графичните библиотеки са съществени при разработката на интерфейсни модули; инструментариумът за реализация на бизнес интелигентност подпомага изграждането на бизнес-интелигентни системи и т.н.

4.2.2. Подходяща организация на данните

Основните видове организация на данните, които стоят в същината на бизнес информационните системи, могат да бъдат обобщени по следния начин:

1. Файлова система – използва се основно в интерфейсни модули за комуникация между приложения.
2. База данни – обикновено съхранява цялата оперативна информация на предприятието – потребител на информационната система. От гледна точка на трудоемкостта на разработка е важно дали използваната СУБД поддържа:
   - декларативна референциална цялост,
   - транзакционен механизъм,
   - многопотребителски достъп,
   - обекти за обработка на множества,
   - съхранени процедури и различни видове тригери,
   - потребителски функции,
   - интерфейс за приложно програмиране,
   - аналитични функции,
   - мениджър на събитията,
   - оптимизатор на производителността при извличане на данни.
3. Склад за данни – използва се предимно като основа за работа на бизнес интелигентните системи и се захранва с данни от оперативната база. По отношение на трудоемкостта на създаване и използване от съществено значения са:
   - Средствата за описание на структурата на склада и измеренията на данните;
   - Средствата за извличане, трансформиране и зареждане на данни от оперативната база в склада.

Бизнес приложенията се отличават от останалите видове софтуер по големите обеми данни и множеството пакетни обработки, изпълнявани върху тях. Този вид софтуер използва предимно база данни за оперативната работа, поради което функционалните възможности на използваната СУБД са от решаващо значение за трудоемкостта на разработката, особено в случаите, когато е необходимо слоят за обработка на данни да реализира бизнес логика. При изграждане на системи от тип „предприятие” следва да се обърне специално внимание на наличието на ефективен оптимизатор на производителността при извличане на данни, тъй като при обработката на големи обеми от данни (милиони редове) необходимостта от оптимизация коства много големи разходи на труд.

4.2.3. Инфраструктура за изпълнение

В зависимост степента, в която извършва разпределението и управлението на компютърните ресурси, инфраструктурата може да бъде:

- Операционната система, която осигурява стандартно управление на компютърните ресурси (процесорно време, памет, периферия, изпълнение на програми и

достъп да данни). Създаването и изпълнението на софтуерни компоненти, работещи директно под управление на операционната система налага включването на програмен код за заделяне и освобождаване на памет, за контрол на достъпа до обектите и т.н, което увеличава значително усилията за разработка и за тестване, ангажирайки програмистите с реализация на технологични механизми, които нямат пряка връзка с функционалността на модула.

- Междинен слой системно програмно осигуряване (инфраструктура), която доразвива функциите на операционната система, осигурявайки контролирано изпълнение на приложенията. Инфраструктурата за контролирано изпълнение управлява паметта, изпълнението на нишките, изпълнението на кода, верификацията на безопасността на кода, компилацията и други системни функции, освобождавайки приложенията от грижата за тях. Общата трудоемкост на разработката намалява допълнително поради по-малкия брой грешки, които могат да бъдат допуснати, което съществено съкращава и улеснява тестването на модулите, които се изпълняват контролирано

Влиянието на средата за разработка и изпълнение на програмен модул върху трудоемкостта може да бъде обобщено по следния начин (Фиг. 9) – трудоемкостта на разработката на софтуер намалява пропорционално на разнообразието и функционалността на заложените готови програмни модули в стандартни и потребителски библиотеки, СУБД и системна програмна среда за изпълнение на приложението.

4.3. Характеристики на проекта на софтуерните компоненти

Процесът на разработка на софтуер (виж т. 2) съществено зависи от предварително създадените условия за работа, а именно:

- Подробна функционална спецификация, която описва логиката на използване и изискванията към създавания софтуер;
- Пълен проект на базата данни и разработваните компоненти;
- Инсталирана и подходящо конфигурирана среда за разработка;
- Създадена инфраструктура за изпълнение;
- Генерирана база данни с евентуално попълнени служебни номенклатури.

Както се вижда от фиг. 3, същинската работа по заданието за програмиране започва от запознаването с проекта на програмния модул.

4.3.1. Съдържание и степен на детайлизация на проекта

Проектът на софтуерен компонент има два основни аспекта:

- функционален,
- технологичен.

Фиг. 9. Влияние на средата за разработка и платформата за изпълнение върху трудоемкостта

Функционалният аспект включва описание на съдържанието и поведението на разработвания модул, вкл.:

1. Описание на интерфейсните елементи:
- тип и евентуално размер,
- разположение и формат,
- съдържание и особености на извличане на стойността от база данни или файл,

- стойност по подразбиране,
- правила за формален контрол,
- правила за логически контрол,
- режими на използване (добавяне, редакция, изтирване и т.н),
- зависимост от други интерфейсни елементи,
- влияние върху други интерфейсни елементи,
- особености на записа на новата стойност в таблица на базата данни или във файл,
- набор от сценарии за тестване на интерфейсния модул и неговите елементи.

2. Описание на процедури, функции, съхранени процедури и услуги:

- декларация (тип на връщания резултат, наименование, параметри, тип на услугата и т.н),
- пълен алгоритъм вкл. всички стъпки на изпълнение и възможни разклонения,
- възможни грешки (изключения), които трябва да бъдат прихванати и обработени,
- изключения, които трябва да бъдат генерирани, при определени условия или при настъпване на различни събития,
- множество от сценарии за тестване на програмния модул.

3. Описание на класове:

- място на класа в йерархията от класове – преки и непреки предшественици и наследници,
- членове данни – вкл. тип, размер и особености при задаването, изчисляването и извличането на стойности,
- членове функции, описани по подобие на процедурите и функциите. В описанието следва да се разграничат наследяваните функции, като се уточни които от тях се използват директно, кои се разширяват и кои се припокриват,
- набор от сценарии за тестване на класа.

4. Описание на тригери:

Това описание също е подобно на функционалните модули, като допълнително е необходимо да се опишат:

- вид на тригера (за всеки отделен ред или за множество редове),
- събитие, което го стартира (непосредствено преди или след добавяне на ред, непосредствено преди или след редакция на данни, непосредствено преди или след изтриване и т.н. в зависимост от особеностите на конкретната система за управление на база от данни),
- евентуално необходимите временни таблици, които се използват за избягване на мутациите.

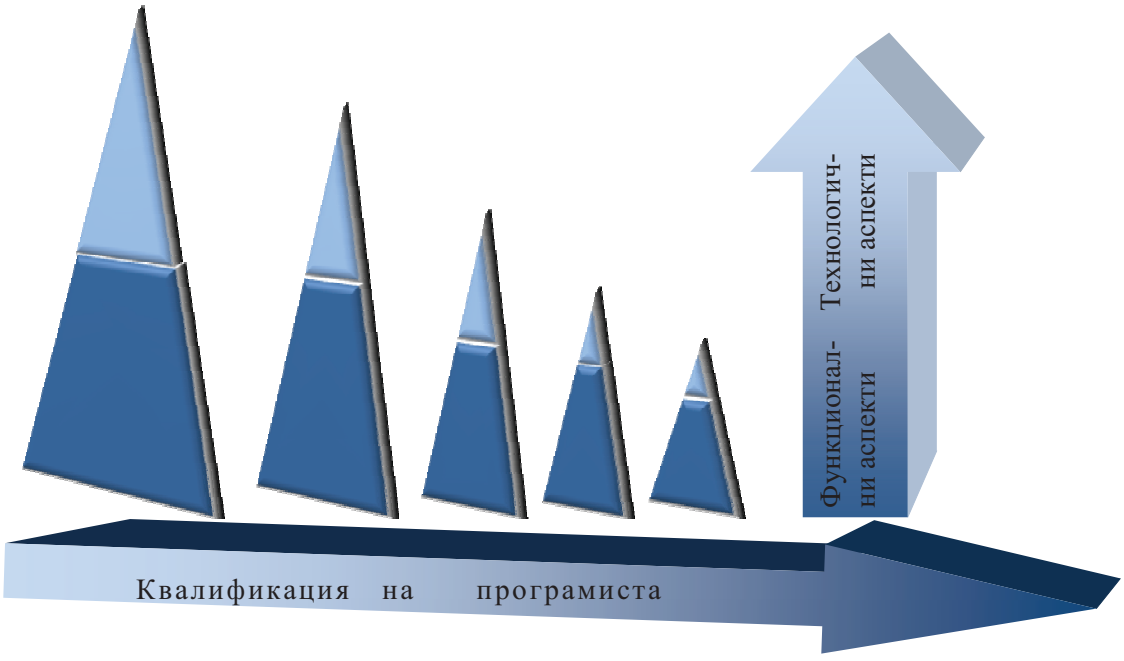
Технологичният аспект на проекта засяга структурирането на програмния код на изграждания компонент и може да включва:

- Организация на структурите от данни и йерархията от класове в рамките на компонента.
- Технология на използване на точно определени елементи от стандартни и/или потребителски библиотеки.
- Указания за разполагане на потребителски код за обработка на данните на ниво отделни събития.

- Указания за структуриране на програмния код на подпрограми (функции или процедури) в рамките на модула.
- Изисквания към използваните синтактични конструкции.
- И др. – според спецификата на използваната среда за разработка.

Практиката показва, че могат да се допускат сериозни отклонения в обхвата и детайлизацията на проектното решение, особено по отношение на неговите технологични аспекти. Колкото по-неопитен е програмистът (виж т. 4.4), който ще реализира даден софтуерен компонент, толкова по-подробно описание на технологичните аспекти на проекта му е необходимо. Опитните програмисти структурират програмния код по-добре от проектантите, без това да се отразява съществено на трудоемкостта на програмната реализация. Нещо повече – прекалено детайлизираният проект (особено в технологичната му страна) води до демотивиране на квалифицираните програмисти.

За разлика от технологичните аспекти на проекта, основната част от функционалните аспекти, особено тези, които засягат алгоритмите на обработката, трябва да бъдат разгледани съвсем подробно. В противен случай на програмистите им се налага да „доизмислят" части от проекта, което значително увеличава трудоемкостта на реализацията. Изключение може да се направи за тези елементи от описанието, които са очевидни от моделите на данните (типове и размери на полета, както и произтичащия от това формален контрол), както и за рутинната обработка на изключенията, тъй като рутинираните програмисти имат изградени навици в тази област.



Фиг. 10. Детайлизация на проектното решение в зависимост от квалификацията на програмиста с оглед постигане на максимална ефективност

Като цяло се налага изводът, че когато даден програмен компонент ще бъде реализиран от неопитен програмист, всеки пропуск в описанията, които съставят проекта, води до рязко увеличаване на трудоемкостта на реализацията. От друга страна, проектните решения, предназначени за опитни програмисти, трябва да бъдат функционално ясни, но да са изчистени от технологични описания на рутинни програмистки техники.

4.3.2. Структура и оформление на проектното решение

Проектните решения могат да се оформят като:

- свободен текст,
- структуриран текст въз основа на предварително възприет шаблон,
- модел, който директно се използва в програмната реализация.

Последният тип проектни решения се създават с помощта на специализирани CASE (Computer Aided System Engineering) инструменти.

От гледна точка на трудоемкостта на разработката на софтуерни компоненти, са съществени две негови характеристики на детайлния проект:

- четимост – т.е. доколко оформлението на проекта улеснява процеса на възприемането му от програмистите;
- използваемост в процеса на програмиране – т.е. доколко е възможно части от проектното решение да се включат директно в програма, разработвана в избраната програмна среда за реализация.

Колкото по-строго е стандартизирана структурата на текста и колкото по-богато е илюстриран проектът, толкова по-лесно се възприема от програмистите, особено в случаите, когато те работят многократно по проекти на еднотипни модули и са свикнали с организацията им.

Фиг. 11. Влияние на структурата и оформлението на проекта върху трудоемкостта на реализацията

Проектите, генерирани с помощта на CASE инструменти, включват набор от диаграми (напр. use-case, activity, sequence, package, class и т.н), които описват различни аспекти на програмния модул като обхват, алгоритъм на обработката, структура на програмния код. Като краен резултат от генерирането на такъв проект се получават набор от стандартизирани описания на модула и множество от декларации на софтуерни компоненти за реализация на заложената бизнес логика (вкл. йерархия на класове с членовете данни и интерфейсите към тях, услуги и т.н), записани в синтаксиса на избраната програмна среда за реализация. По този начин програмирането става естествено продължение на проектирането – запълва се със съдържание генерираният „скелет" на програма и се добавят софтуерни компоненти за реализация на някои технологични изисквания. Този тип модели най-съществено намаляват трудоемкостта на реализацията. На обратния полюс са слабо структурираните текстови документи, в които е трудно да се отдели съществената информация, а последователността на изложението не съответства на технологията на разработка (Фиг. 11).

4.4. Квалификация и заинтересованост на програмистите

Квалификацията и заинтересоваността на всеки отделен програмист като член на разработващия екип са от съществено значение за неговата производителност и следователно пряко влияят върху продължителността на изпълнение на възлаганите им задачи.

Квалификацията на програмистите се определя от два взаимосвързани показателя:

- опит в използването на съответната програмна среда за разработка и изпълнение на софтуера;
- опит в създаването на близки по тип програмни компоненти в същата или сходна като архитектура програмна среда;
- мотивация за работа по конкретното задание за програмиране.

Запознаването на програмистите с тънкостите на определена среда за програмиране (вкл. документираните и недокументираните й грешки), както и с възприетите в екипа технология и стандарти на използване на средата и разработените към нея потребителски библиотеки, изисква период на обучение от минимум няколко месеца. Обикновено практиката "learning by doing" в рамките на реален проект не дава добри резултати от гледна точка на качеството на разработените програмни модули – те са лошо структурирани и тежки за поддръжка.

Практиката показва, че трудоемкостта за разработка на първия програмен модул в дадена среда е на порядък по-голяма от тази на следващите няколко проекта, след което програмирането става рутина.

Аналогично стоят проблемите, свързани с опита в създаването на определен тип софтуерни компоненти – интерфейсни, за реализация на бизнес логика или за манипулиране на данни: създаването на първия програмен модул от даден тип отнема на порядък повече време, отколкото е необходимо за следващите няколко подобни модула, след което производителността на програмиста става сравнително постоянна. Тази зависимост се проявява особено силно, когато усвояването на особеност-

тите в технологията за разработка на даден вид софтуерен компонент се комбинира с прехода към непозната или слабо позната информационна технология.

Мотивацията на програмистите се определя преди всичко от:

- интереса към разработвания проблем.
- организацията на екипа и самочувствието на всеки отделен негов член.

Интересът на програмиста към разработвания проблем зависи както от степента на разбиране на неговата значимост в съответната предметна област, така и от чисто субективното положително или отрицателно отношение към използваната информационна технология. Основна грижа на ръководителя на проекта е да създава и поддържа интереса към него, а следователно и мотивацията на програмистите на необходимото ниво. В следващото изложение се има предвид по-скоро вторият аспект на заинтересоваността на програмистите, а именно този, който е свързан с тяхното място в йерархията на разработващия екип.

Организацията на екипа за разработка може да бъде 2 основни типа:

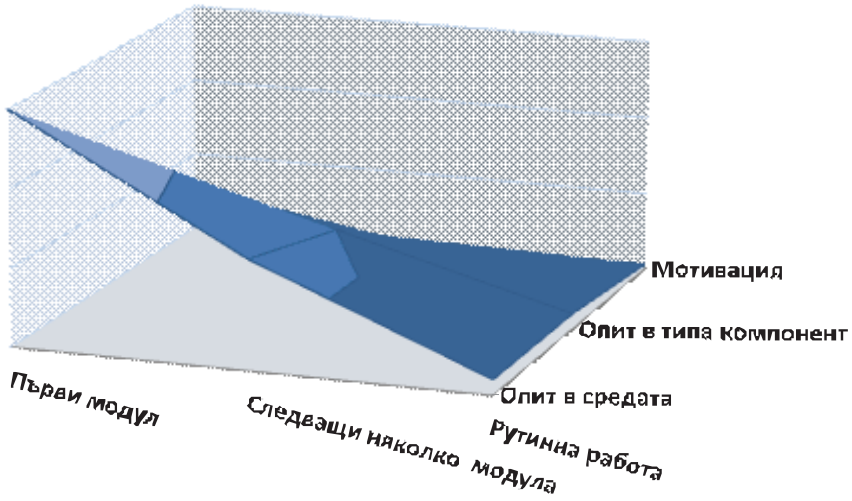
- адаптивна с виртуален екип,
- традиционна с постоянен екип.

Екипите от тип „виртуален" или „ad-hoc" [7] са съставени от участници, външни за организацията или са ангажирани само в рамките на този проекти за определено време. Понякога единственият „истински заинтересован'" участник в такива екипи (зает на пълен работен ден и работещ само по тази задача) е ръководителят на проекта. Този тип екипи се отличават със слаба лоялност на участниците, „доверието" и взаимоотношенията в екипа са в рамките на професионалните изисквания към работата, налице е слабо споделяне на натрупан опит. В допълнение на това следва да се отбележи, че оценката на квалификацията на външни членове на екипа е изключително ненадеждна.

Традиционният екип (наречен още whole-of-life team [7]) възниква в среда, която е в значителна степен стабилна и постоянна по отношение на ролята на всеки участник, изискванията към неговата компетентност и времевата му заетост. Това е предпоставка са формиране на екипи, които се отличават с: висока степен на лоялност на членовете към екипа, споделяне на опит и идеи, доверие между членовете в екипа и ефективност. Голямо значение за производителността на екипа има специализацията на неговите членове в различните типове софтуерни компоненти (интерфейсни, за бизнес логика, компоненти на БД), стига тя да не е толкова строга, че да ограничава тяхното развитието и по-този начин да ги демотивира.

Управлението на човешките ресурси в рамките на проекта и тяхната дългосрочна мотивация, вкл. перспективите им за професионално и материално израстване са съществени и за производителността на труда по всяка конкретна задача (влияят на продължителността на нейното изпълнение), но са извън обхвата на настоящия материал.

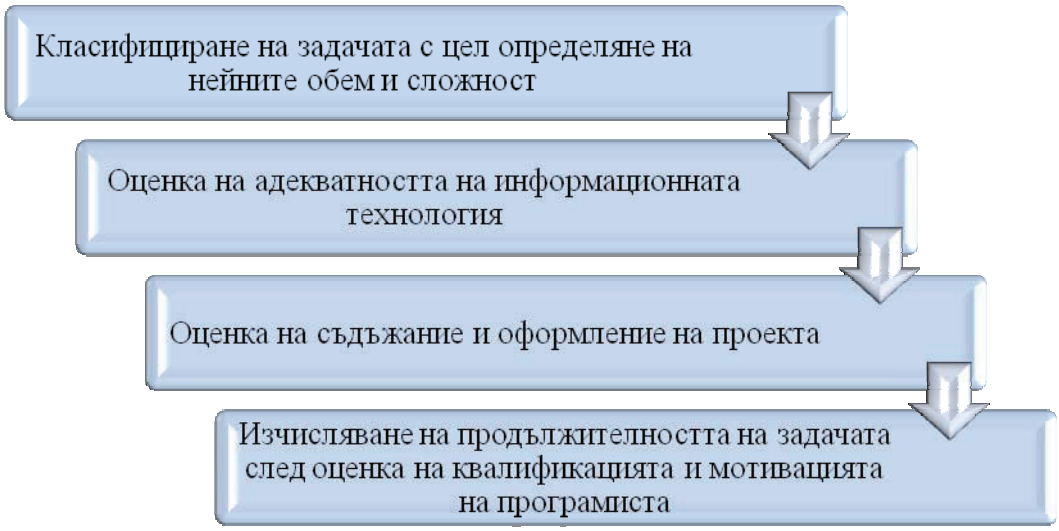
Влиянието на квалификацията и заинтересоваността на програмистите върху продължителността на разработката на софтуер може да се обобщи в тримерната графика, която е представена на фиг. 12. Направената оценка е приблизителна и се основава на натрупания опит на автора като ръководител или участник в екип от ръководители на повече от 10 софтуерни проекти.



Фиг. 12. Влияние на квалификацията и заинтересоваността на програмиста върху продължителността на разработката

5. МЕТОДИКА ЗА ОЦЕНКА НА ПЛАНОВАТА ТРУДОЕМКОСТ И ПРОДЪЛЖИТЕЛНОСТТА НА ЗАДАЧИТЕ ПРИ РАЗРАБОТКАТА НА СОФТУЕРНИ МОДУЛИ ВЪЗ ОСНОВА НА ГОРЕПОСОЧЕНИТЕ ФАКТОРИ

Прецизното отчитане на всички фактори, влияещи върху трудоемкостта на разработката на софтуер, които са разгледани в предходното изложение, би довело до абсурдната ситуация времето за оценка на трудоемкостта на дадена задача да стане съпоставимо с времето за нейното изпълнение. По тази причина се предлага методика за оценка на плановата трудоемкост, която отчита само факторите, които влияят най-силно върху трудоемкостта (Фиг. 13).



Фиг. 13. Методика за оценка на плановата трудоемкост и продължителност на задачите

1. Класифицира се задачата с цел да се определят нейните обем и сложност. На тази основа се определя базова трудоемкост на задачата ($\mathbf{T}_{базова}$).

При класифициране на задачите според техния обем и сложност се използват факторите, влияещи върху трудоемкостта на разработката на различни видове софтуерни компоненти (интерфейсни и безинтерфейсни), които бяха подробно разгледани в т. 0. В допълнение следва да се има предвид, че:

- При програмирането обемът и сложността на работата не са линейно свързани и не влияят еднакво върху трудоемкостта на задачите.
- Като цяло, факторите, които се измерват като брой (напр. брой стъпки в алгоритъм, брой елементи във форма, брой операции и т.н), влияят по-скоро върху обема на задачата, отколкото върху нейната сложност. Единствено изключение от това правило е факторът „брой разклонения в алгоритъма", който влияе както върху обема, така ъ върху сложността на задачата.
- По принцип, факторите, които касаят „разнообразие" – например разнообразие на видовете компоненти, различните режими на работа, различните операции върху тях, връзки между структури от данни и т.н, влияят пряко върху сложността на задачите, т.е. увеличават трудоемкостта, без да е задължително да увеличават обема на работата.
- Най-съществено повишават трудоемкостта на програмиране на бизнес логика високите изисквания за оптимизация на кода по отношение на бързодействие и обем на заеманата памет. Тази зависимост се проявява толкова по-силно, колкото повече итерации включва реализираният алгоритъм и колкото по-голям е обемът на извличаните и обработвани данни. Практиката познава редица случаи, в които, за да се постигнат заложените параметри за бързодействие и разход на памет, се извършва цялостна преработка на програмния код, а в някои случаи се

налага дори промяна на проектното решение (избор на по-сложен за реализация, но по-ефективен алгоритъм).

- Сложността на интерфейсните модули най-силно нараства при наличието на множество разнообразни взаимни зависимости между компонентите на диалога с потребителя. Реализирането на такива зависимости може да се налага като част от логическия контрол на въвежданите данни или по „чисто естетически причини", т.е. подобряване на прегледността на представянето. В случаите, когато сроковете на проекта са кратки или човешките ресурси са недостатъчни, се препоръчва вторият тип зависимости да бъдат избягвани от проектното решение. За разлика от този подход към зависимостите между интерфейсни елементи, които са наложени по естетически причини, със зависимостите, които са функционално обусловени и предпазват потребителите от въвеждане на некоректни данни, по-трудно се правят компромиси. „Спестяването" на програмната реализация на логическите зависимости води до намаляване на полезността на модула за крайните потребители [3], поради което не трябва да се практикува освен при крайно ограничени срокове и ресурси по проекта.

В крайна сметка класифицирането на задачите според възможните комбинации на техните обем и сложност може да се извърши по следната примерна таблица:

Таблица 1.

Оценка на трудоемкостта в зависимост от класификацията на задачите според техния обем и тяхната сложност

Обем / Сложност	Много голям	Голям	Среден	Малък	Много малък
Много голяма	20	19	19	17	17
Голяма	14	13	13	12	12
Средна	9	8	8	7	7
Малка	5	5	4	4	3
Много малка	3	3	2	1	1

Коефициентите в таблицата ($\mathbf{К}_{задача}$) са ориентировъчни и могат да бъдат попълнени с други стойности, получени на базата на исторически данни, експертно мнение или комбинация от двете.

По-горе беше подчертано, че съществена особеност на разработката на софтуер е трудоемкостта на задачите да зависи в много по-голяма степен от тяхната сложност, отколкото от обема на работата. Тази зависимост се проявява толкова посилно, колкото по-развита среда за програмиране се използва. По тази причина, предложените в таблицата коефициенти ($\mathbf{К}_{задача}$) намаляват стръмно при намаляване на сложността на задачата и намаляват значително по-плавно (а в някои случаи запазват стойностите си) при намаляване на обема на работата.

За целите на настоящата методика се приема, че минималната стойност на базовата трудоемкост (за елементарна задача с много малък обем) е 1, а максималната

(за много сложна и много голяма задача) е 20. Счита се, че софтуерен компонент, чиято базова трудоемкост надвишава 20 работни човекодни трябва да бъде декомпозиран на по-малки и по-лесно управляеми модули, за да не се усложни прекомерно неговата разработка, тестване и поддръжка.

2. Оценява се адекватността на средата за разработка, инфраструктурата за изпълнение и използваната СУБД на вида на задачата (Таблица 2).

Таблица 2.

Оценка на трудоемкостта според адекватността на информационната технология спрямо типа на софтуерния компонент

Тип на задачата / **Необходим инструментариум**	**Потребителски интерфейс**	**Бизнес логика**	**Обработка на данни**	**Бизнес интелект**
Синтактични конструкции от високо ниво	X	X	X	X
Обща система от типове	X	X	X	X
Стандартни и потребителски елементи на потребителския интерфейс	X			
Стандартни и потребителски обекти за извличане, визуализиране и запис на данни	X	X	X	X
Графични библиотеки за 2D и 3D графика	X			X
Инфраструктура за контролирано изпълнение	X	X		
Транзакционен механизъм, управление на многопотребителския достъп до данните			X	X
Тригери, съхранени процедури, потребителски функции		X	X	
Оптимизатор на производителността		X	X	X
Средства за извличане, преобразуване и зареждане на данни				X

Съществено необходимите средства за разработка на съответен тип софтуер са обобщени в следната таблица (по-ярко са оцветени компонентите, чието наличие е критично за трудоемкостта на разработката, а по-бледо, тези, чието наличие е препоръчително).

Отсъствието на критичен за трудоемкостта инструмент може да я увеличи до 3 пъти, докато отсъствието на препоръчителен компонент увеличава трудоемкостта с 40 – 60%.

Трудоемкостта на разработка на интерфейсни модули зависи най-съществено от

наличието в средата на библиотеки със стандартни и потребителски елементи на потребителския интерфейс. За реализация на бизнес логика най-съществени са синтактичните конструкции от високо ниво, както и стандартни и потребителски обекти за извличане, визуализиране и запис на данни. Трудоемкостта на изграждане на слоя за управление на данните се определя в най-голяма степен от мощта на избраната система за управление на базата от данни. При обработката на големи обеми от данни критично може да се окаже и наличието на оптимизатор на производителността. Прави впечатление, че наличието на обща система от типове и на среда за контролирано изпълнение е съществено (макар и в различна степен) за всички типове софтуерни компоненти.

Въз основа на направената оценка на адекватността на средата за програмиране и изпълнение, базовата трудоемкост на задачата се коригира (увеличава) със съответен коефициент ($\mathbf{К}_{среда}$). В случай, че избраните среда за разработка и инфраструктура за изпълнение на програмния модул съдържат целия необходим инструментариум за програмиране, тестване и изпълнение на програмния модул, коефициентът $\mathbf{К}_{среда}$ би трябвало да има стойност 1, т.е. да се запазва базовата трудоемкост на задачата.

3. Оценяват се съдържанието, оформлението и използваемостта на проекта, след което се коригира трудоемкостта, изчислена в предходната точка.

Вариант за оценка на влиянието на съдържанието и оформлението на проекта ($\mathbf{К}_{проект)}$ върху трудоемкостта на разработката на проектирания модул може да се обобщи в следната таблица (Таблица 3):

Таблица 3.

Оценка на трудоемкостта според съдържанието и оформлението на проекта на разработваната задача

Съдържание / Оформление	Много подробно	Достатъчно подробно	Минимално необходимо
Неструктурирано	1,8		2
Структуриран текст			
Модел на модула			
Скелет на модула	1,1	1	1,1

Коефициентите в таблицата могат да бъдат попълнени въз основа на исторически данни или експертно мнение. Възможно най-малкият коефициент (предложена е стойност 1) на увеличаване на трудоемкостта се използва, когато в резултат от проекта е получен скелет на разработвания модул, а съпътстващото го описание е „достатъчно подробно". Най-голяма трудоемкост имат разработките, които се извършват въз основа на неструктурирани описания. Следва да се обърне внимание на факта, че скалата за оценка на съдържанието на проекта започва от „Минимално необходимото", тъй като обикновено проектите, които не съдържат необходимото

функционално описание, не отговарят на изискванията и по тях не се възлага задача за програмиране.

Оценената по този начин трудоемкост се залага при изготвяне на предварителното разписание на проекта, което се създава преди разпределение на задачите между човешките ресурси. Предварителната оценка на трудоемкостта ($\mathbf{Т}_{предварителна}$) може да се извърши по следната формула

$$\mathbf{Т}_{предварителна}=\mathbf{Т}_{базова}*\mathbf{К}_{задача}*\mathbf{К}_{среда}*\mathbf{К}_{проект}$$

4. След назначаване на ресурсите по задачи се оценява квалификацията на съответния програмист и се изчислява продължителността за изпълнение на всяка конкретна задача.

Равнището на квалификацията на програмистите се определя според броя разработени модули от дадения тип и със средствата на съответната информационна технология

Коефициентът за изчисляване на продължителността за изпълнение на задачата в зависимост от квалификацията на програмиста ($\mathbf{К}_{квалификация}$) започва от 0,6 за програмистите, за които разработката на съответен тип софтуер със средствата на определена информационна технология е рутинна, и се увеличава до 1,5-1,6 за начинаещите програмисти. Използването на дробен коефициент (за намаляване на трудоемкостта) в този случай се препоръчва, за да може предварителното разписание на проекта (преди назначаване на ресурси) да бъде изготвено реалистично, като за всяка задача се разчита на програмист със средна квалификация (неговият коефициент е 1). Следва да се отбележи също така, че непознаването на информационната технология влияе по-силно на трудоемкостта на реализацията, отколкото липсата на опит в разработката на съответен тип софтуерни компоненти.

Таблица 4.

Оценка на квалификацията на програмиста и нейното влияние върху продължителността на изпълнение на възлаганите му задачи

Опит в типа компонент / **Опит в информационната технология**	**Голям**	**Малък**	**Никакъв**
Голям	0,7-0,8		
Малък		1	
Никакъв			1,5-1,6

Както беше посочено в т. 4.4, мотивацията на програмистите при участието им в конкретен проект може съществено да повлияе на продължителността на работата. Като правило следва да се приеме, че при равни други условия:

- мотивацията на участниците в традиционни екипи е по-висока в сравнение с мотивацията на ad-hoc екипите;
- колкото по-висока е позицията на даден участник на екипа, толкова по-мотивиран да работи би трябвало да е той.

Мотивацията на програмистите може условно да се класифицира на 3 степени – висока, средна и ниска ($\mathbf{К}_{мотивация}$), като на всяка степен експертно се определя съответен коефициент. Препоръчително е при средна мотивация коефициентът да е 1, за да може тя да съответства на планираната оценка на трудоемкостта ($\mathbf{Т}_{планова}$), която, за да бъде реалистична, се планира при средни стойности на влияещите фактори.

След оценка на всички посочени групи фактори продължителността на разработката на софтуерни компоненти може да се изчисли по формулата:

$$\mathbf{П}_{планова}=\mathbf{Т}_{планова}*\mathbf{К}_{квалификация}*\mathbf{К}_{мотивация}$$

Оценената по тази методика планова продължителност на работата може да се използва като реалистична основа за окончателно изчисляване на разписанието на софтуерен проект, както и за бюджетиране на разходите. За целта е препоръчително тя да се използва в специализиран софтуер за управление на проекти, който въз основа на въведената продължителност за изпълнение на задачата изчислява всички останали параметри на разписанието, вкл. общо количество на работата, начална и крайна дати за изпълнение на задачите и .т. н.

6. ЗАКЛЮЧЕНИЕ

Настоящата студия представя методика за оценка на плановата трудоемкост и продължителност на разработката на системни и приложни програмни модули.

Разгледан е жизненият цикъл на софтуерните проекти, както и възможните подходи към този жизнен цикъл в процеса на управление на проекта. Като резултат от това са изведени условията (известните величини), при които се извършва оценката на плановата трудоемкост на разработката на софтуерни компоненти. На тази основа са обособени основните групи фактори, от които зависи въпросната трудоемкост. Факторите, които са определящи за трудоемкостта на разработката на софтуер, са разгледани самостоятелно и във взаимната им връзка.

Като краен резултат от изследването е предложена цялостна методика за оценка на плановата трудоемкост и продължителност на разработката на системни и приложни програмни модули, която е приложима при управление на софтуерни проекти по традиционна, еволюционна, инкрементална и ускорена методология.

Разработената методика за оценка на плановата трудоемкост на системно и приложно програмиране може да се използва в следните случаи:

- Изготвяне на предварително и на окончателно разписание на софтуерни проекти;
- Разпределение и преразпределение на задачите между човешките ресурси при разработка на бизнес софтуер, особено при необходимост от балансиране на трудоемкостта на проектиране спрямо тази за проектиране, както и при реализиране на планирани и/или непланирани рискове по проекта;
- Изчисляване на стойността на разработка на софтуер при възлагането й на външни консултанти и подизпълнители;
- Оценка на стойността на работата по реализация на изменения и допълнения в обхвата на софтуерни проекти.

Предложената методика е формализирана в достатъчна степен, за да се използва като основа за разработка на софтуерни средства за автоматизация на този бизнес процес, което съществено би увеличило нейната полезност за ръководителите на софтуерни проекти, както и за останалите заинтересовани лица.

ЛИТЕРАТУРА

1. Велев Д., Е. Денчев, К. Стефанова, В. Лазарова, М.Цанева, А. Мурджева. SOA – основно направление в развитието на софтуерните технологии. В: „Бизнес информатика”- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г. стр. 78-85
2. Денчев Е., К. Стефанова, М. Цанева, Д. Велев, В. Кисимов, А. Мурджева, В. Лазарова. Проблеми и решения при избор на ERP система. В: „Бизнес информатика”- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г. стр. 183-190
3. Лазарова В., Е. Денчев, Д. Велев, А. Мурджева, В. Кисимов, К. Стефанова, М. Цанева. Критерии за определяне на потребителската ефективност на информационните системи в Интернет. В: „Бизнес информатика”- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г.,стр 223 - 230
4. Мурджева А., К. Стефанова, М. Цанева, В. Лазарова, Е. Денчев, Д. Велев. Разделяне на бизнес логиката в многослойни приложения и съвременните технологии. В: „Бизнес информатика”- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г. стр. 231-242
5. Мурджева А. и Моника Цанева. „Принципи за проектиране на потребителския интерфейс на бизнес информационни системи”, Управленски, информационни и маркетингови аспекти на икономическото развитие на балканските страни, Сборник доклади от международна научна конференция, София, Ноември 2004,стр. 319-331
6. Стефанова К., В. Кисимов, М.Цанева, В. Лазарова, А. Мурджева, Д. Велев, Е. Денчев, Д. Кабакчиева. Проектиране на център за компетентност по бизнес интелигентност. В: „Бизнес информатика”- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г., стр. 86-99

7. Цанева М. и Ал. Мурджева. „Управление на човешките ресурси на софтуерни проекти", Управленски, информационни и маркетингови аспекти на икономическото развитие на балканските страни, Сборник доклади от международна научна конференция, София, Ноември 2005,стр. 247-259

8. Цанева Моника, В. Кисимов, Е. Денчев, А. Мурджева, Д. Велев, К. Стефанова, В. Лазарова. Интегриране на бизнес приложения – основно предизвикателство към системното програмиране. В: „Бизнес информатика"- сборник доклади от юбилейна международна научна конференция по повод 40-та годишнина на специалност Информатика, С., 11 октомври, 2007 г., стр. 215-222

9. **CIO - бр. 12, 2007 / ИТ мениджмънт,** Планиране на времето за проектите или как да съкратим сроковете и същевременно да ги спазим, http://cio.bg/?call=USE%7Ecategory;&catid=8

10. **CIO - бр. 12, 2007/ Проучвания**, Световният пазар на корпоративен софтуер – под влиянието на динамични промени, http://www.cio.bg/?call=USE%7Ecategory;&catid=9

11. Alistair Cockburn, Characterizing People as Non-Linear, First-Order Components in Software Development

12. COCOMO, New COCOMO II Book, http://sunset.usc.edu/research/COCOMOII/

13. Duncan W., A guide to the project management body of knowledge, Project Management Institute, Four Campus Boulevard, Newtown Square, USA

14. Futrell, Shafer, Shafer, "Quality Software Project Management", Prentice Hall PTR; 1 edition (Jan 24 2002)

15. Simon Yates, Worldwide PC Adoption Forecast, 2007 To 2015,It's Time To Focus On Emerging Markets For Future Growth, June 11, 2007 http://www.forrester.com/Research/Document/Excerpt/0,7211,42496,00.html

16. http://www.comp.glam.ac.uk/Teaching/projectmanagement

17. http://www.hyperdictionary.com/

ПЛАНИРАНЕ НА ТРУДА НА ПРОГРАМИСТИТЕ ПРИ ИЗГРАЖДАНЕ НА ИНФОРМАЦИОННИ СИСТЕМИ

Резюме

Голяма част от изследванията в областта на управлението на софтуерни проекти са насочени в посока как да бъде намалено времето за разработка, но не и как да бъде оценена трудоемкостта на задачите и продължителността на тяхното изпълнение. Настоящата студия предлага методика за оценка на плановата трудоемкост и времето за изпълнение на задачите при системното и приложно програмиране на бизнес софтуера. За целта са дефинирани основните групи фактори, които оказват влияние на трудоемкостта и продължителността на разработката на софтуер, а именно:

- Вид и сложност на създавания системен програмен модул;
- Адекватност на средата за разработка, организацията на данните и платформата за изпълнение към вида на изгражданите софтуерни компоненти;
- Съдържание и оформление на проекта на възложения за разработка програмен модул;
- Квалификация и заинтересованост на програмистите.

Предложен е подход за оценка на сложността на програмните модули. Разгледани са характеристиките на системната програмна среда за разработка и на инфраструктурата за изпълнение, които са съществени от гледна точка на трудоемкостта на реализацията на всеки отделен тип софтуерни компоненти. Дефинирани са основните елементи на проектните решения, които оказват влияние върху трудоемкостта на програмната реализация. Набелязани са някои аспекти, които определят равнището на квалификация на програмистите, както и тяхната мотивация при изпълнение на възложените им задания.

Като краен резултат, въз основа на дефинираните фактори, е създадена методика за оценка на планираната трудоемкост и продължителност на разработката на софтуерни компоненти на бизнес приложения.

PLANNING OF PROGRAMMERS LABOR THROUGH DEVELOPMENT OF INFORMATION SYSTEMS

Abstract

Main part of researches in the area of software project management is concentrated in direction of development time reduction, but do not refer problems of estimation of efforts, needed for task implementation and their duration. This paper suggests a methodic for estimation of planned labor - consumption and of task duration of system and application programming of business software. For this purpose, main groups of factors influencing over software development labor-consumption and task performance duration have been defined as follows:

- Type and complexity of the software module, which is developed;
- Suitability of the development environment, data organization and execution framework to the software components type;
- Design contents and layout of the task assigned for development;
- Programmers qualification and motivation.

An approach for estimation of the software modules complexity is proposed. The characteristics of development environment and execution framework, which are essential to efforts needed for each type of software components development, are considered.

Main items of module design, which infer programming efforts, are defined.

Some aspects, which determine the level of programmers' qualification and their motivation during assigned task execution, are outlined.

As a final result, a methodic for estimation of planned labor – consumption and task duration of programming of business applications software components using the defined factors, is developed.