София 1849, кв. Кремиковци„ тел./факс: +359 2 994 70 18, +359 882 82 66 83, +359 876 24 82 04
www.chrdri.net

МОДЕЛИ НА ЖИЗНЕНИЯ ЦИКЪЛ

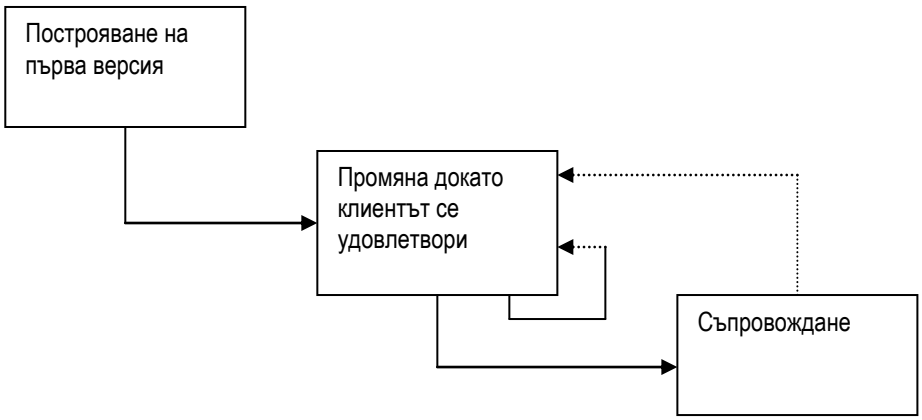
Основните дейности на процеса на разработване обикновено се реализират последователно във времето. Често обаче се налага някои от тях да се повтарят неколкократно. Възможно е също така някои дейности да се реализират паралелно във времето, а други да се пропуснат. Това се определя от специфичните особености на разработваната система.

Последователността на дейностите определя логиката на процеса. Под модел на жизнения цикъл се разбира съвкупността от всички стъпки при производството на даден продукт. Първоначално то се отнася само за разработването на програмно осигуряване. В последствие разширява своя обхват в областта на разработването на информационни системи и реорганизацията на управленски системи.

Изобщо модел е апроксимирано/ приближено представяне на даден обект от действителността. Цел на моделирането е възпроизвеждането на съществени свойства на изследвания обект, за да могат те да се изучат с оглед решаването на определена задача – управленческа, производствена, икономическа.

При жизнения цикъл за разработване основните дейности се обособяват в *етапи или фази.* На тази база става възможно управлението и контрола върху тяхното изпълнение. Следва да се отбележи, че има различни мнения относно броя и наименованията на етапите на жизнения цикъл за разработване.

Build and Fix Model



При него се разработва целият продукт и се доставя на клиента. Последният посочва кое не му харесва и трябва да се преработи. Модификации се правят докато клиентът стане доволен. След това продуктът започва да се използува.

Проблеми:

1. Скъпа цена на поправки след като продуктът окончателно се изгради и се използува върху клиентски компютър.
2. Липсват дейности като спецификация, планиране, проектиране и поддържането

става кошмар. Липсата на план води до изграждането на модули, които нямат никаква организация помежду си.

Кога да се използува този модел?

При малки проекти, когато вероятно продуктът няма да се използува особено много – курсови задачи, домашни.

Този модел е НЕПРИЛОЖИМ при разработването на софтуер.

Каскаден модел (Боем 81 г. – Waterfall model)

Неговите основни характеристики може да бъдат обобщени като:

- последователно изпълнение на етапите във времето;
- верификация и формално потвърждаване на всеки етап.

При този модел на процеса фазите са разположени линейно. Той има различни варианти, обикновено започва с изследване на възможностите (пред проектно проучване). След като се докаже, че проектът е възможен се преминава към анализ на изискванията и планиране на проекта (няма да го разглеждаме, планът се изготвя успоредно анализа на изискванията и трябва да е готов преди останалите фази да започнат, планирането е критична дейност за успеха на проекта). Проектирането започва след като завърши фазата Анализ на изискванията, а кодирането - след края на фазата Проектиране. При успешно тестване системата се инсталира, започва нейното действие и съответстващата му поддръжка (фиг.3.1):

При каскадния модел редът на дейностите при разработването на даден проект е:

- анализ на изискванията;
- проектиране на системата;
- кодиране и тестване на отделните единици;
- интегриране на системата
- тестване
- инсталиране;
- съпровождане / използване

Линейното подреждане на дейностите налага проверка на изхода от всяка фаза преди започването на следващата. Това се прави чрез верифициране (тестване в края на фазата) и валидиране (тестване за съвместимост с изискванията на цялата система).

Изходи за всеки проект при каскадния модел са следните:

- документ за изискванията;
- план на проекта;
- документ за проекта на системата;
- документ за детайлния проект;
- план на тестване и отчети от него;
- краен вид на кода;
- ръководства за софтуера;
- отчети от прегледа.

Всички без последния от тях представляват изходи от отделните фази, които се преглеждат, прави се съответен отчет и се преминава към следващата фаза.

Каскадният модел е целесъобразно да се прилага при системи, за които изискванията са ясно формулирани и са устойчиви във времето. Анализът и проектирането се осъществяват

от системни аналитици и проектанти. Потребителите са пасивен обект на изследване.

Преимущества:

1. Моделът е document-driven, т.е. всяка фаза завършва с одобрен документ. Основна причина за порвала на много проекти е неподходящата, лошата, некоректна документация или изобщо липсата на такава.
2. Обратна връзка – грешките се коригират при откриването им.

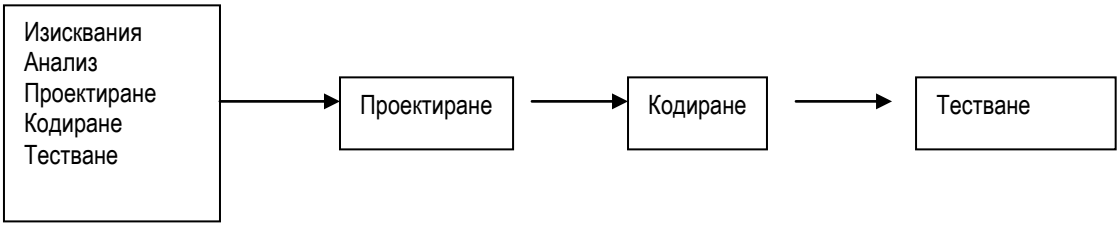
Ограниченията на каскадния модел са следните:

1. Допуска замразяване на изискванията преди започване на проектирането. Това е възможно при системи, които се разработват за да автоматизират съществуващи ръчни такива. За нови системи окончателното определяне на изискванията в една фаза е много трудна задача. Изискването да не се променят повече изискванията е нереалистично за подобни системи.
2. Замразяването на изискванията налага избор на хардуера. При големи проекти може да минат години докато се завършат и окончателно избрания хардуер може да се окаже остарял.
3. Каскадният модел налага определяне на изискванията преди да започне каквато и да е друга работа по системата. За някои проекти е по-удачно да се разработи изцяло част от системата, а след това тя да се подобрява поетапно. Такива проекти са свързани с разработване на софтуер предназначен за пазара, а не за определен клиент, който може да определи точно изискванията си.
4. Каскадният модел е процес, който се управлява от документи, понеже изисква всяка фаза за завършва с определен документ. Това прави процеса на документиране тежък и неподходящ за интерактивни приложения, където трудно се описващ потребителските интерфейси. Освен това, ако за разработването се използват езици от четвърто ниво или други средства, разработването на детайлна спецификация преди реализацията се оказва ненужна стъпка.

Каскадният модел притежава редица недостатъци. Той е трудоемък и бавен процес. Връзката с потребителя е прекъсната (индиректна). Той придобива реална представа за системата едва след нейното внедряване. Често потребителят не одобрява резултата, оказва се, че много от изискванията не са реализирани или не са интерпретирани правилно. При промяна на изискванията е необходимо да се повторят всички етапи на разработването, а това е много скъпо.

Прототипиране

Цел на процеса на разработване чрез прототипиране е да преодолее първите две ограничения на каскадния модел (фиг.3.2).



Основната идея е вместо да се замразяват изискванията преди да се продължи с проектиране и кодиране да се направи прототип на системата, който да спомогне за разбирането на изискванията. Този бърз прототип се разработва като се използват известните изисквания, клиентът може да се запознае с бъдещата система, а след това прототипът се изхвърля. По този начин изискванията се уточняват по-добре с участието на клиента. Тогава се прави Спецификацията. Това отлага останалите фази, но след това е вероятно изискванията да се променят по-рядко.

Прототипирането е атрактивно при големи и сложни системи където не съществува друга система, с чиято помощ да се определят изискванията. При такива ситуации работата на клиента с прототипа спомага силно да се разберат изискванията към системата. Това също е ефективен начин да се покаже, че даден подход за решаването на проблема е възможен.

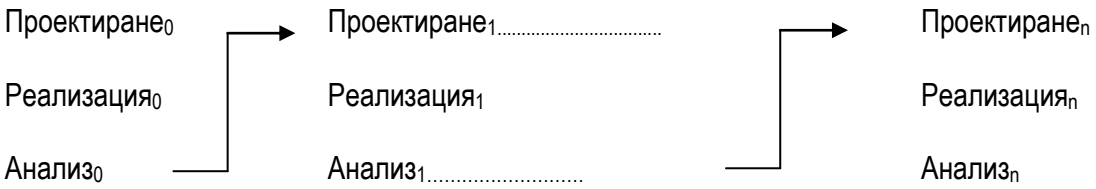
Разработването на прототипа със събраните от клиента първоначални изисквания. След като прототипа е готов, той се предоставя на клиентите и крайните потребители за да го проиграят. Те осигуряват обратната връзка с разработчика като указват какво да се промени, добави, премахне и т.н. разработчикът променя прототипа и процесът се повтаря докато изискванията към системата бъдат окончателно уточнени.

Прототипът се прави набързо без да се изпипва всичко за да бъде евтин. Понеже в крайна сметка той се изхвърля в него не се реализират онези от изискванията, които са ясни на разработчика. Набляга се върху бързината, а не върху качеството, не се прави документиране, тества се минимално.

Прототипирането не се използува много често поради страх от оскъпяване на проекта. То се прилага при големи проекти, за които е вероятно да се променят изискванията с времето и за които натрупването на опит с разработването на прототип поевтинява следващите фази понеже се натрупва опит.

Итеративно подобряване (iterative enhancement)

Моделът на процеса наречен итеративно подобряване преодолява третото ограничение на каскадния модел като се опитва да комбинира преимуществата на Прототипирането с каскадния модел (фиг.3.3).



Основната идея е, че софтуерът трябва да се разработва на стъпки, постепенно, като всяка стъпка добавя някаква функция на системата. Преимуществото на този подход е в по-доброто тестване на системата, което се прави на всеки етап. Освен това постепенното разработване осигурява, както при Прототипирането, обратна връзка с клиента, като дава възможност за доуточняване на изискванията към системата.

В първата стъпка на този модел се прави реализация на малко подмножество на задачата. Реализацията съдържа някои от основните характеристики на системата и може да се използува самостоятелно. Създава се списък за управление на проекта, който съдържа редът на задачите, които трябва да се реализират за да се изгради системата изцяло. Той описва с колко се доближава всяка от стъпките до окончателния вид на системата.

Всяка стъпка се състои в избор на задача от списъка за управление на проекта и нейното реализиране (проектиране, кодиране, тестване на получената система). Получената система се анализира и се обновява списъкът със задачи. Трети фази в рамките на всяка стъпка са: проектиране, реализиране, анализ. Процесът продължава докато се изчерпи списъкът със задачи.

Този модел се използува успешни при разработването на продукти където разработчикът сам определя изискванията. Повече от продуктите използуват този процес на разработване. Първо се предлага версия (release), която има някакви възможности. След това като се използува обратната връзка с потребителите се създава списък с допълнителни възможности, които водят до подобряването на софтуера и се включват в следващите версии на продукта.

При разработване на софтуер по поръчка на клиент този модел трудно се прилага, защото има договор за разработка.

Спираловиден модел

Това е съвременен модел предложен от Боем 1988 г. (фиг.3.4), който оценява рисковите фактори при разработването на софтуер. При него се провежда анализ на риска преди началото на всяка фаза. Разработчиците вземат предпазни мерки за да се осигурят, че няма да работят на загуба. В момента, в който се прецени, че загубата от продължаване на работата по проекта е твърде голяма, то той се спира незабавно. Името на модела идва от неговата схема, на която са изобразени отделните фази във вид на спирала. Всяка фаза се състои от анализ на риска, основна дейност, проверка (фиг. 3.4).

фиг.3.4

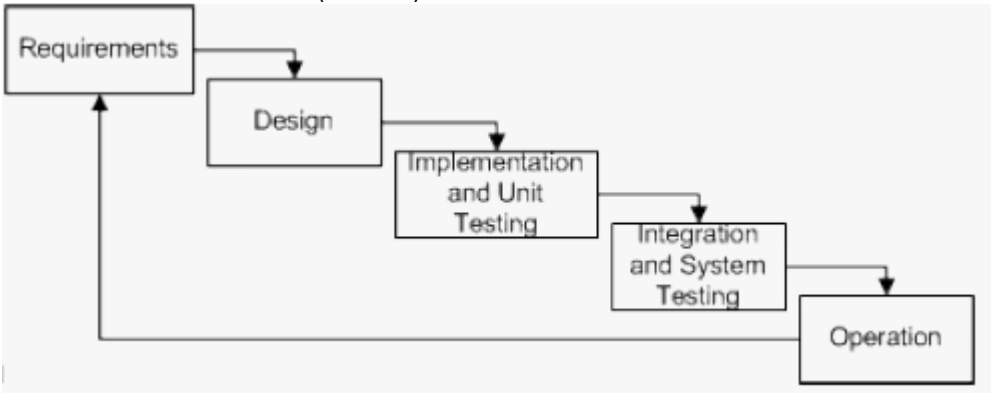
Основните цели са:

- предоставяне на възможност за корекция на грешките;
- подобряване на качеството на продукта чрез осъществяване на обратна връзка с потребителя;
- контрол на ефективността.

Според Боем този модел може да се използува, ако се разработва голям софтуерен продукт в рамките на дадена организация, т.е. няма външен клиент и договор между две страни. Тогава по-лесно се преценява дали си струва да се разработва продукта или е по-добре хората да се заемат с друга работа.

Постъпателен (incremental) модел

При постъпателния модел (инкрементален) отделни елементи от софтуера се разработват през различно време, в различна степен на завършеност и, когато са готови, се интегрират в цялостната система. Основна характеристика е разработването по части. След разработването на всяка част, тя се добавя към вече готовите части и по този начин софтуерът постепенно се разширява и усъвършенства. Преди да започне работата по следващото разширяване трябва да се предвиди време за оценка на системата и да се направи анализ за определяне на новите изисквания (Фиг.3.5).



фиг.3.5

Наборът от функции на разработвания софтуер се разширява постепенно. На първата итерация се реализират част от функциите на системата – получава се първа работна версия на продукта. Те се тестват от потребителите и, ако е необходимо, се правят корекции. След тяхното приемане от потребителите се преминава към реализирането на нови функции.

При постъпателния модел се осигуряват възможности за допълване и разширяване на изискванията към системата. Така значително се намалява рискът за разработване на система, която не удовлетворява потребностите на потребителите.

Разгледаните модели са хронологични доколкото представят процеса като последователност от фази, които се редуват във времето. Съществуват модели, които определят основните функции в процеса на разработването на продукта - наричат се функционални.

Освен това съществуват двумерни модели - на Гънтър, при който едното измерение - фази (хронологичен модел), второ измерение - функции. Има и тримерни модели, които са сложни са използуване.