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

АНАЛИЗ И СПЕЦИФИКАЦИЯ НА СОФТУЕРНИТЕ ИЗИСКВАНИЯ

Софтуерните изисквания

IEEE дефинира изискването като "условие или възможности на системата за удовлетворяване на договор, стандарт, спецификация или друг формално наложен документ". В тази фаза се разглеждат възможностите на системата, която подлежи на разработване и това прави задачата за обхващане на изискванията сложна. Тя завършва винаги с документа "Спецификация на софтуерните изисквания", който описва какво трябва да прави предложения софтуер без да казва как ще го прави. Цел на тази фаза е създаването на този документ, който описва пълното поведение на предлагания софтуер.

На пръв поглед това не изглежда трудна задача, но практическите трудности се свързват с промяната на обкръжаващата среда, в която ще работи бъдещата система. Това важи особено силно за сложни приложения, при които функциите на бъдещата система не могат да се обхванат от всички разработчици. Освен това често срещано явление е в процеса на уточняване на изискванията, на клиента да му хрумнат нови идеи относно функциите на системата. Така, че при сложни системи често се получава промяна на изискванията още след създаването на документа Спецификация. Всичко това може да вгорчи отношенията между клиент и разработчик, но трябва да се знае, че промяната на изискванията е нещо неизбежно особено при големи проекти.

Защо е необходимо разработването на документа Спецификация на софтуерните изисквания?

Произходът на повечето системи е свързан с удовлетворяването на някакви нужди на клиент, който или иска да автоматизира някаква ръчна система, или желае разработването на нова такава. В този процес има три заинтересувани страни: клиент, разработчик, потребител. Разработчикът комуникира както с клиента за да разбере неговите изисквания, така и с потребителя на системата, когото трябва евентуално да обучи да работи с нея. Основен проблем в това общуване е, че клиентът не разбира нищо от разработването на софтуер, а разработчикът често не разбира проблемната област на клиента. Възниква комуникационна пропаст между двете страни. Затова основна цел на спецификацията на софтуерните изисквания е да се намери общ език между двете страни като се дефинират точно потребителските нужди. Специфицирането на софтуерните изисквания е основа на разработването. Друга цел на този процес е да подпомогне клиента да разбере собствените си нужди.

Преимущества на процеса Спецификация на софтуерните изисквания:

1. Представлява основа за съгласие между клиента и разработчика относно функциите на разработваната система. Тук се сключва договор за това какво ще прави разработваната система. И винаги въпреки договора клиентът често не е удовлетворен!!
2. Спецификацията служи за валидиране на крайния продукт. По този начин клиентът може да определи дали софтуера, който му е доставен и за който е платил отговаря на изискванията.
3. Качествената Спецификация означава качествен софтуер, което е една от

основните" цели на всеки проект. Боем доказва, че около 25% от грешките се въвеждат по време на фазата Спецификация и другите ранни фази. Всяка грешка в Спецификацията означава грешка в крайната система, чието отстраняване струва много скъпо.

4. Качествената Спецификация намалява цената на разработване.

Фазата Изисквания се състои от три основни дейности и завършва с произвеждането на документа Спецификация:

1. Анализ на проблема или анализ на изискванията. Това е най-трудната и неясна дейност, която обхваща разбирането на проблема, неговите ограничения и цели. Основна цел е да се разбере какво трябва да прави софтуерът.
2. Спецификация на изискванията. Изискванията се оформят в документ като се описват подходящо.
3. Валидиране на изискванията. Проверява се дали са зададени всички изисквания и се проверява качеството на документа Спецификация.

Очевидно дейностите не са подредени линейно, а често се припокриват и между тях има обратна връзка.

Анализ на проблема или анализ на изискванията

Цел: да се разберат ясно нуждите на клиента и потребителите на системата, какво точно се изисква от софтуера и какви ограничения съществуват върху решението. Хората, които извършват анализ на изискванията се наричат аналитици (analyst) и отговарят за специфициране на изискванията.

Анализът включва интервюта с клиентите и крайните потребители. Тези интервюта, както и съществуващите документи са основен източник на тази дейност. Събира се огромно количество информация. Тя трябва да се анализира с помощта на клиента за да се проведи обхванати ли са всички функции, които трябва да изпълнява системата и дали няма противоречиви изисквания. Последните могат да възникнат поради различното разбиране на целите на системата от клиента и крайните потребители. Добре е информацията да се структурира за да се провери дали тя е пълна. Аналитикът трябва да умее да общува с различни хора.

Подходите използувани за анализ на изискванията са:

1. Неформални подходи, използуващи общуване
2. Подходи, анализиращи функциите на системата - структурен подход
3. Подходи, анализиращи обектите в системата – ОО моделиране
4. Прототипиране

Неформален подход

При него няма ясно дефинирана методология. Информация за разработваната система се получава чрез разговори с клиента, крайните потребители, въпросници, изучаване на съществуващи документи и т.н. Не се прави формален модел на системата. Първоначално аналитикът се явява в ролята на слушател, докато разбере същността на бъдещата система. Той може да документира неформално данните по някакъв начин. След като осмисли събраната информация аналитикът обяснява на клиента какво ще прави бъдещата система за да провери дали тя действително ще отговаря на неговите нужди. Този подход е широко разпространен и е доста полезен при системи където концептуалното моделиране е неподходящо.

Анализирайки проблемната област задачата, която ще решаваме може да се разбие на подзадачи по два начина: по отношение на функционалност - структурен подход, по отношение на обекти - обектно-ориентирано моделиране.

Структурен анализ

Използува функционално-базирано декомпозиране при моделирането на проблема. Обръща се внимание върху отделните функции характерни за разглежданата проблемна област и входно- изходните данни необходими на всяка от тях. Структурният анализ представлява подход за проектиране отгоре- надолу с уточняване (top-down refinement approach). Той помага на аналитика да реши какъв тип информация му е необходима и да я организира подходящо. Подходът използува два основи метода: Data Flow Diagrams - диаграми за потока на данните и Data Dictionary - речник на данните.

Диаграмата на потока на данните показва потокът на данните чрез система, представлява функция, преобразувала входните данни в желани изходи. Всяка сложна система не прави подобна трансформация на една стъпка и данните са подложени на множество от трансформации преди да се получи изхода от системата. Диаграмите описват именно тези трансформации, които се наричат процеси или мехурчета. По този начин диаграмите на потока на данните показват тяхното движение между отделните трансформации или процеси. Всяка трансформация или процес се бележи с окръжност, а потоците се отбелязват със стрелки. Правоъгълникът представя източник и може да бъде генератор или консуматор на данни.

Пример – виж презентация

Тази диаграма показва абстрактна дефиниция на разработваната система. Особености:

- Не личи каква е системата - автоматизирана или не.
- Не са показани детайли от пътя на данните
- Това не е блок-схема. Диаграмата представя само потока на данните и не дава никаква информация за процедурите, т.е. как ще се реализират трансформациите.
- Потоците на данни се идентифицират с различни имена

Точната структура на всеки поток се описва в речника на данните.

След като се построи диаграмата на потока на данните и съответния речник, те се проверяват за коректност от аналитика и чрез разговор с клиента.

Описание на метода

Според структурния анализ всяка система се разглежда като трансформиращи функции опериращи в дадена среда, които получават вход от нея и произвеждат съответни изходи. За да се открия функциите на системата се анализира потокът на данните като се построяват съответните диаграми. Методът се състои от следните стъпки:

Стъпка 1: Изучаване на физическата среда

Създава се диаграма на потока на данни на съществуващата неавтоматизирана система - например за склад. Нарича се контекстна диаграма, като цялата система се разглежда като единичен процес за дадената физическа среда. Идентифицират се всички входове, изходи, източници и др. Целта е потребителя да представа за функционирането на системата. Фиг.- виж презентация.

Стъпка 2: Създаване на логически еквивалент на системата

Създава се логически еквивалент на създадената в първата стъпка диаграма като се оставят само онези процеси, при които има трансформиране на данните.

Фиг. – виж презентация

С тези две стъпки се моделира текущата система.

Стъпка 3: Разработване на логически модел.

Разработва се логически модел на новата система и се създава съответна диаграма. През тази стъпка аналитикът работи на логическо ниво, т.е. указва какво трябва да се направи, а не как да стане това. Няма разлика между автоматизиран и неавтоматизиран процес или трансформация. Няма и правила при разработването на диаграмата на потока на данните за новата система, всичко зависи опита на аналитика и неговата представа за бъдещата система.

Стъпка 4: Установяване на границата човек-машина

Указва се коя част от системата ще бъде автоматизирана и коя ще остане ръчна. В нашия пример готвенето, вземането на поръчките и сервирането остават ръчни процедури, а останалите процеси т.е. трансформации на данните могат да се автоматизират.

Стъпка 5: Оценка на различните възможности

Стъпка 6: Представяне на изискванията.

За създаването на диаграма на потоците на данните се препоръчва подходът отгоре-надолу започвайки от контекстна диаграма. Тази диаграма в последствие се уточнява и се включват повече детайли. Получават се множества от диаграми на няколко нива, всяка от които описва различна част от потока на данните. Задача на аналитика е да запази всички входни и изходни потоци, както и да няма противоречие при уточняването.

Структурният анализ осигурява методи за организиране и представяне на информация за системите. Той осигурява и ръководство за проверка точността на информацията. Той е подходящ за анализ и разбиране на съществуваща система. Повечето от правилата му са приложими само в първите две стъпки когато се построява диаграма на потока на данните за съществуваща система. Трудно се построява диаграма на бъдещата система.

Обектно-ориентирано моделиране

При обектно-ориентираното моделиране или обектно-ориентирания анализ системата се разглежда като съвкупност от обекти, които взаимодействуват помежду си чрез услуги. Цел на анализа е идентифицирането на обектите, които съществуват в проблемната област, определянето им и задаване на връзките помежду им. Целият модел осигурява желаните от потребителя услуги.

В последните години обектно-ориентираното моделиране се прилага доста, защото се счита, че с негова помощ се създават системи, които лесно се изграждат и поддържат. Обектите винаги остават едни и същи дори и да се смени решаваната задача, което не може да се каже при функционално-ориентирано моделиране. Съществуват много подходи за обектно-ориентирано моделиране, които си приличат.

Основно понятие е обектът, който притежава определен брой атрибути. Подобните обекти се групират и образуват клас. Класът може да се разглежда като шаблон за група от обекти със сходни свойства и определя операциите (услугите) върху тях. Стойностите на атрибутите са достъпни единствено чрез операциите на класа. Обектите комуникират помежду си чрез съобщения. Когато даден обект получи съобщение, той изпълнява определена операция. Капсулирането в едно на атрибути и операции заедно в един обект е основно свойство на обектно-ориентираното моделиране, което го отличава от други подходи за моделиране на данните (например ER .-модела). Обикновено обектите се изобразяват графично чрез диаграми, най-често се използува правоъгълник, разделен на три части. В първата част се задава класа, към който обектът принадлежи, във втората - атрибутите, а в третата - списък на операциите.

За моделиране на връзките между класовете и обектите се използуват три вида структури:

- Обобщение/специализация - използува се от даден клас за да наследи някои или всички атрибути и операции от друг по-общ клас и за да добави собствени такива. Тази структура се моделира чрез наследяване. Новополученият клас се нарича специализация. Получават се йерархии от класове, представят се връзки от вида “това е”.
- Агрегация – структурата моделира връзка от вида “част от”. Обектът може да бъде изграден от много други обекти и това се моделира чрез агрегация.
- Асоциация - тя моделира множества от обекти. На диаграмите се бележи с плътна линия между двата класа. Всяка асоциация има множественост multiplicity (сравни с кардинално число), която показва по колко обекта от един клас се свързват с обекти

от другия клас и се бележи точки от двете страни. Множествеността може да бъде 1:1, т.е един обект от класа се свързва с точно един обект от другия клас, 1:N, M:N, или някакъв друг специален тип. Асоциацията моделира връзки между отделните обекти.

Основните компоненти на обектно-ориентирания модел са:

1. Обекти;
2. Класове - за класификация;
3. Наследяване - за моделиране на обобщение/специализация;
4. Агрегация - за моделиране на връзки "част-от"
5. Асоциация - за моделиране на връзки между отделните обекти.

Описание на метода

Построяването на обектен модел на дадена система минава през следните стъпки:

1. Идентифициране на обектите. Анализира се описанието на проблемната област и се отделят всички съществителни имена. Те представляват вероятни обекти, ако за тях трябва да се пазят данни, ако те трябва да изпълнява някакви услуги и имат собствени атрибути. В системата не се представят други обекти от действителността.
2. Идентифициране на структурите. Структурите представят йерархии, които могат да съществуват между класовете от обекти. Всички сложни системи имат йерархии. При обектното моделиране те представят връзки от вида "това-е" и "част-от". При връзка от вида "това-е" класът, на който се прави специализация се нарича суперклас, а неговите специализации подклас. Пример с EMPLOYEE. От всички възможни се разглеждат само смислените за дадено приложение специализации. Получената структура трябва да отразява йерархия в проблемната област. При връзка "част-от" трябва да се идентифицират сложните обекти и техните компоненти. Проверява се дали системата пази данни за отделните компоненти; ако да - те се моделират като отделни обекти.
3. Идентифициране на атрибутите. Атрибутите отразяват свойствата на обектите и представляват хранилища на данните за тях. Достъп до техните стойности имат само операциите върху обекта. За тяхното идентифициране се анализират свойствата на отделните обекти и се определя местоположението на всеки атрибут като се използуват структурите. Ако атрибутът е общ, той се слага в суперкласа, ако е специфичен - в подкласа.
4. Идентифициране на асоциациите. Те моделират връзки между отделните обекти по подобен начин на ER-модела. Връзките биват 1:1,1:N, M:N.
5. Дефиниране на услугите (операциите). Те определят поведението на системата и чрез тях се описват функциите, които системата осигурява.

Диаграмата на обектите става доста голяма при сложни системи. Затова се препоръчва тя да се разбие по нива, като всяко ниво обхваща част от диаграмата.

Съществуват и други подходи, които анализират функциите или обектите в системата.

SADT - Structured Analysis and Design Technique
PSL/PSA - Problem Statement Language/ Problem Statement Analyzer
ER - modeling

Прототипиране

При този подход системата се построява частично като прототип, който се използува от

клиент, разработчик и потребители за да разберат проблема и изискванията към евентуална софтуерна система. Използува се когато системата е нова и няма предишна такава. Тогава разработването единствено по написани на хартия изисквания е трудно. Целта е да се визуализира действието на софтуера. Тогава характеристиките на бъдещата система могат ясно да се обобщят.

Съществуват два подхода при прототипирането:

1. Throwaway - при него прототипът се изхвърля след като изпълни предназначението си и системата се разработва отначало. Затова в него не се реализират онези елементи от системата, които са добре разбрани. Разработването е "бързо и мръсно", т.е. не се държи на качеството. Експериментирането с прототипа се използува за да се определи дали са разбрани точно изискванията към системата.
2. Evolutionary – при него прототипът се разработва като се има предвид, че ще се преобразува в разработваната система. Затова в него се реализират онези части от системата, които са разбрани добре като се следи за качеството. По този начин работейки със стабилна и надеждна система, може да се определи какви още характеристики са необходими. Подходът се използува когато са ясни функциите на системата.

Първият подход throwaway води до прототипиране като процес на разработване, а втория – до итеративно подобряване. От гледна точка фазата Анализ на изискванията първият подход е подходящ и приложим при нови системи. Вторият подход се използува когато се разработва нова версия на съществуваща система.

Ще разглеждаме първия подход.

Възниква въпросът кога да се прави прототип за даден проект и кога не? Изискванията за една система могат да се разделят на три групи:

- Добре разбрани;
- Лошо разбрани
- Неизвестни.

В прототипа се реализират лошо разбраните изисквания, които от своя страна могат да се разделят на още две групи: критични за проекта и некритични за проекта. В прототипа се реализират критичните лошо разбрани изисквания. Следователно, ако множеството от лошо разбрани, критични изисквания е значително се прави прототип.

Характеристики на throwaway прототипирането:

1. Прототипът трябва да е евтин.
2. Прототипът се разработва бързо
3. Почти не се документира.
4. Не се следи за качество.
5. Не се следи за ефективност на решението.
6. Прототипът се използува от клиент и потребители за да се допълнят изискванията.
7. Тестването е минимално.

Да не се преобразува прототипа с реалната система, поради горните съображения.

Спецификация на изискванията

При тази дейност се използува натрупаното от анализа знание за системата. Преминаването от анализ към специфициране не е директно, дори и ако е използувано някакво формално моделиране. Една добра Спецификация изисква задаването на повече подробности, отколкото са събрани чрез моделиране, а може да се разработи и ако такова не се прилага.

Характеристики на Спецификацията (според IEEE):

1. Коректна - всяко изискване в Спецификацията е изискване и на крайната система
2. Пълна - описва всички функции на софтуера
3. Недвусмислена - всяко изискване има единствена интерпретация
4. Проверяема - всяко изискване може лесно да се провери от клиент, потребител и разработчик
5. Непротиворечива - няма конфликтни изисквания
6. Подлежаща на промяна - да се променя лесно, защото самото написване на Спецификацията е итеративен процес
7. Лесно проверяема – да е ясен произхода на всяко изискване.

От всички тези изисквания най-трудно се осигурява пълнота на Спецификацията.
Основни елементи на Спецификацията:

1. Функционални изисквания - задават какви изходи трябва да се получат от входните данни. Описват се всички операции върху данните, логически и формален контрол, използувани формули и др. НЕ СЕ задават алгоритми, тяхното разработване се оставя на проектанта. Описват се всички ненормални ситуации като невалидни входни данни, грешки в изчисленията и какви действия предприема системата. Извод: описва – се поведението на системата при всички възможни случаи.
2. Изисквания за производителност- задава ограничения свързани с производителността на системата, съществуват два типа изисквания за производителност: статични и динамични. Статичните изисквания не налагат ограничения върху параметрите на изпълнение на системата. Те включват изисквания като брой терминали, брой едновременно работещи потребители, брой файлове, които системата обработва и техния размер. Динамичните изисквания задават ограничения върху изпълнението, например време за отговор, брой транзакции, които се обработват за една минута и др. Всички тези изисквания се задават в измерими единици, а не "времето за отговор трябва да бъде добро".
3. Ограничения при проектирането - биват:
   - Изисквания да се спазва определен стандарт
   - Хардуерни ограничения
   - За надеждност и защита на системата.
4. Изисквания за външния интерфейс - задават се всички.възможни интерфейси - с потребители, хардуера и друг тип софтуер.

Езици за специфициране:
Естествени езици – преимуществото е, че клиент и разработчик се разбират. Недостатък |- двусмисленост и неточно формулиране на изискванията.
Таблици на решенията
Крайни автомати

Структура на документа Спецификация:

Стандартите на IEEE признават, че различните проекти могат да имат различни изисквания и че няма един единствен метод за организиране на изискванията, който да е подходящ за всички видове проекти. Те предлагат следната структура:

1 .Въведение
1.1.Цел
1.2 Обхват

1.3 Дефиниции, съкращения
1.4 Литература
1.5 Преглед
2.0писание на системата
2.1 Перспектива на продукта
2.2 Функции на продукта
2.3 Характеристики на потребителите
2.4 Общи ограничения
2.5 Допускания и зависимости
3. Специфични.изисквания - например
3.1.Изисквания към външния интерфейс
3.2 Функционални изисквания
3.3 Изисквания за производителност
3.4 Ограничения при проектиране

Горното е само ръководство за организиране на документа Спецификация.

Валидиране

Поради естество на фазата Спецификация на изискванията е възможно погрешно разбиране и допускане грешки. Възможно е също така да не са отразени някои от клиентските изисквания. Основна цел на дейността валидиране е да се осигури точността и верността на документа Спецификация.

Какви видове грешки могат да се допуснат в една Спецификация?

- Липса на изискване
- Противоречивост
- Неточен факт
- Двойственост

Изследвания показват, че всяка една от тях е значителна и затова цел на дейността валидиране трябва да бъде тяхното откриване.

Методи за валидиране на Спецификацията:

- Преглед на Спецификацията - прави се от работна група, включваща представители на клиента и потребителите. Това е най-често срещания метод за Валидиране. Изследвания показват, че метода е доста ефективен и се откриват 83% от грешките.
- Четене на Спецификацията - чете се не от автора, а от друг човек
- Построяване на сценарии - описват се различни ситуации, при които ще работи действуващата система и се проследява връзката с потребителите.
- Прототипиране - използува се за проверка .на някои от изискванията, т.е. за отговор на въпроса "може ли това да бъде направено?"