Структурен и обектноориентиран анализ

1.Структурен анализ

Структурния анализ (Structured Analysis, SA) се базира на методите за структурно програмиране. При него софтуерните системи се представят чрез абстрактни диаграми на потоците от данни. Данните и функциите се моделират като самостоятелни единици. Функциите отразяват изменението на потоците от данни.

Основната задача на структурния анализ е диаграмите на потоците от данни да се детайлизират на отделни нива и да се представят като йерархични структури. За целта се прави декомпозиция на процесите на подпроцеси, всеки от които се детайлизира и представя със собствена диаграма. Всяка диаграма се означава с идентификатор, състоящ се от абревиатурата DFD (Data Flow Diagram) и число, което показва нивото на абстракция в йерархията. Например, DFD0, DFD2, DFD2.5. Така се построява дърво на процесите. Ако един процес или подпроцес не може да бъде декомпозиран повече, той се представя с миниспецификация. Миниспецификациите (MiniSpecs) могат да бъдат таблици, псевдокод или дърво на решенията. За тях са валидни следните правила:

- трябва да описват как входните данни се преобразуват в изходни;
- не трябва да съдържат правила за изпълнение.

Описанието на интерфейсите на моделираната и изследвана система се прави с контекстни диаграми, които се съставят по следните правила:

- трябва да съдържат поне един интерфейс;
- трябва да представят само един процес;
- не съдържат хранилища на данни;
- между интерфейсите няма потоци от данни;
- всеки интерфейс се представя само веднъж.

Допуска се за по-добра разбираемост на диаграмите даден интерфейс да се представи почеве от един път.

Според правилата на CASE (Computer Aided Software Engineering) за диаграмите могат да се добавят още следните правила:

- Нивото на абстракция трябва да е едно и също за всички процеси и потоци в една диаграма;
- Номерирането на процесите започва от 0;
- Всяка диаграма има номер, който показва позицията ѝ в йерархията.
- Номерът на диаграмата се поставя пред номера на процеса, например DFD2.5 – е втора детайлизация на процес номер 5.

За построяване на DFD могат да се използват CASE средства.

Построените диаграми (контекстни и на потоците от данни), както и реализираната декомпозиция на дават възможност да се построй фунционално дърво на системата.

Структурният анализ има следните предимства:

- ✓ лесен за усвояване;
- ✓ позволява построянване на йерархии на системите на базата на различни нива на абстракция и детайлизация на процесите и функциите;
- ✓ удачен е за приложение при проектиране на системите от горе надолу или от общото към детайлното (top-down design).

2. Диаграми на потоци от данни

Диаграмите на потоци от данни (DFD)се използват за създаване на модели на изследваната информационна система. Моделът се представя като йерархия на

диаграми на потоци от данни. Той описва процеса на преобразуване на данните в информация за дадената система.

Основните компоненти на диаграмите на потоци от данни са:

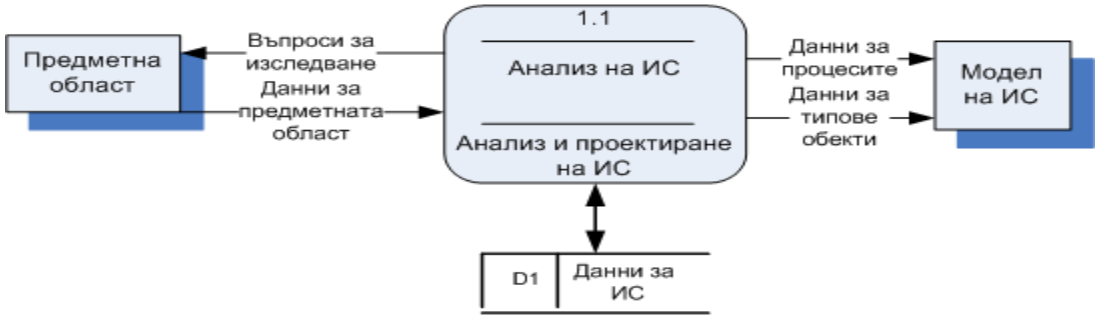
- ✓ Външен обект - физическо лице или материален предмет, които е източник или приемник на информация и е външен за иформационната система (читател, студент, склад, клиент);
- ✓ Процес - представлява преобразуване на входен поток от данни в изходен, в съответствие с определен алгоритъм. Описва с глагол, след който следват съществителни (заемане на книги, приемане на поръчки);
- ✓ Хранилище на данни - абстрактно устройство за съхранение на данни. Означава се с буквата "D" и произволно число (D1,D34).
- ✓ Системи и подсистеми – големите и слежни ИС се разбиват на подсистеми;
- ✓ Поток данни - определя информацията предавана от източника към приемника на данни. Всеки поток от данни има име.

Използват се два вида DFD.

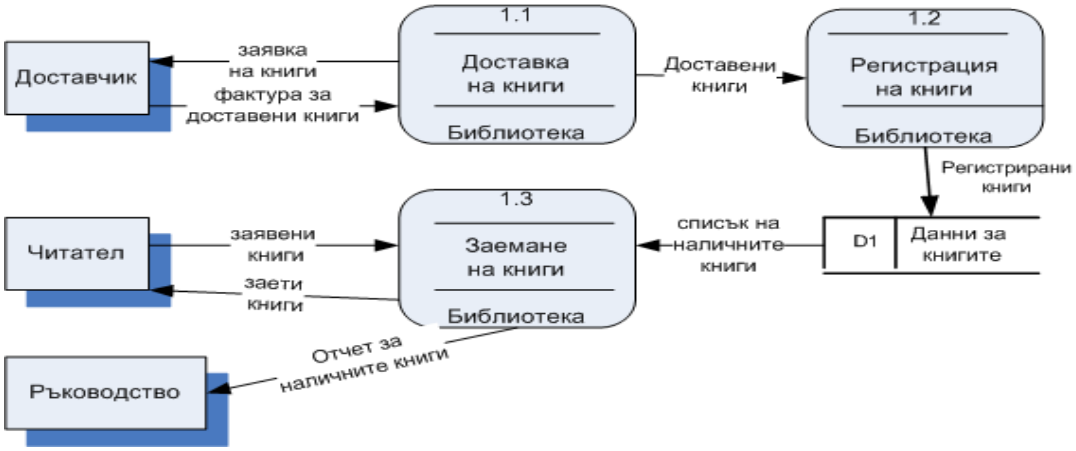
• **контекстни диаграми (диаграми от високо ниво)** – определят основните процеси или подсистеми на ИС с външните входове и изходи.

• **диаграми от ниско ниво** – детайлизират диаграмите контексните диаграми. Прилага се декомпозиция на процесите с цел да се получи йерархична структура. Разделянето продължава докато е необходимо или възможно.

Примери за диаграми от високо и ниско ниво са показани на фиг. 1 и 2.



Фиг. 1. Контекстна диаграма на процеса „Анализ на ИС”



Фиг. 2. Контекстна диаграма на второ ниво на ИС за библиотека

Източниците на информация (външни обекти) генерират информационни потоци, пренасящи информация към подсистеми или процеси. Те от своя страна преобразуват информацията и създават нови потоци, пренасящи информацията към други процеси или подсистеми, хранилища на данни или външни обекти.

3. Обектноориентиран анализ

Обектноориентираният анализ (ООА) се използва за създаване на формализиран модел на разработваните софтуерните системи, в това число и на ИС. По дефиниция той се определя като „метод, които представя изискванията към софтуерния продукт като класове и обекти, които могат да се синтезират от проблемната област". Този анализ се базира на обектноориентираното програмиране (ООП).

ООА може да бъде разгледан в различни аспекти: модел, език за моделиране, методология и средства.

ИС се декомпозира на отделни обекти и класове. Представят се два модела на системата – статичен и динамичен. Начинът на разработване на модела на ИС се определя от приложената методология. А най-често използвания език за моделиране е UML (Unified Modeling Language).

Обект е основно понятие, изплзвано във всички етапи и фази на проектиране и реализация на системите. То може да се дефинира по следните начини:

- Софтуерна единица, съдържаща множество от логически свързани помежду си данни и операции;
- Комбинация от сруктури от данни и поведение;
- Структура, съставена от памет и множество операции;
- Комбинация от данни, съхраняващи състоянията на обектите и операции, предоставящи механизмите за достъп и манипулация с тези състояния.

Клас е съвкупност от подобни обекти. Той е също е абстрактна единица, която има състояние, поведение и идентичност, като структурата и поведението на подобните обекти.

Основно средство за визуално представяне на моделиране при ООА са клас - диаграмите. Използваните модели могат да бъдат: базови, статични и динамични.

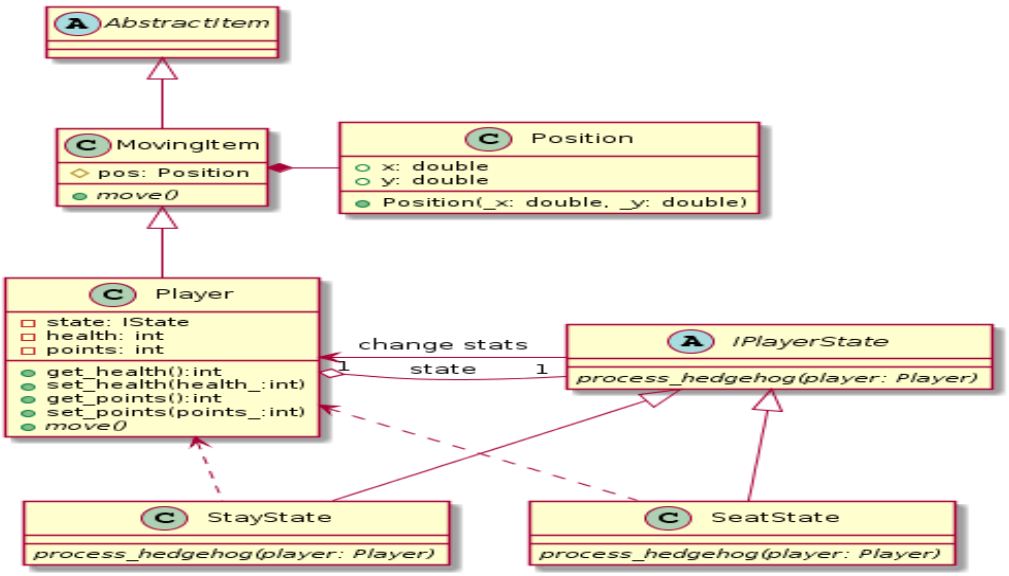
Базовият модел представя основните изграждащи елементи на модела – обекти, класове, атрибути, операции и полиморфизъм (припокриване на операции или различни реализации на една и съща функционалност).

Статичния модел описва архитектурата на системата, а динамичния – нейното поведение.

За представяне на тези модели обикновено се използват UML диаграми. В езика UML са дефинирани множество (14 на брой) диаграми, някои от които са:

- ✓ Диаграми на класовете – показват класовете и техните връзки;
- ✓ Диаграми на обектите – моделират обектите и техните връзки;
- ✓ Диаграми на компонентите - показват компонентите и техните връзки;
- ✓ Диаграми на състоянията – моделират динамичното поведение на обектите;
- ✓ Диаграми на действията – моделират паралелни процеси;
- ✓ Диаграми на сценарии за използване – представят функционалните възможности на системите и взаимодействието на потребителите с тях.

Диаграмата на класове (Class diagram) се използва за описване на статичната структура на дадена система. Чрез нея се описват класове, обекти, пакети и връзките между тях. В диаграмите класове се изобразяват с правоъгълник. Той може да е разделен на три хоризонтални части, в които влизат съответно име на класа, атрибути и операции. Примерна диаграма на класове за една компютърна игра е представена на фиг. 3.



Фиг. 3. UML диаграма на класове на компютърна игра

Диаграмата на компонентите (Component diagram) е структурна диаграма, която отразява архитектурата на компонентите на системата и зависимостите между тях (фиг.4).

Диаграмата на състоянията (State diagram) се използва като допълнение на диаграмата на класове. Те показват всички възможни състояния, които обектите могат да заемат и събитията, които предизвикват промяната на състоянието или преход в ново състояние. Обектите се представят като крайни автомати, които заемат състояния. Под състояние на обект се разбира съвкупност от стойностите на атрибутите му. Състоянията на обектите могат да бъдат начални, междинни и крайни. Когато даден обект се създава неговото състояние е начално, а когато не може повече да се променя – крайно. От началното състояние в следващото се преминава в под въздействието на събитие (действие). Възможно е в рамките на състоянието да се разграничат междинни състояния, които да не предизвикват излизане от текущото състояние. Състоянията се запазват, но във вътрешността им се оформят подсъстояния.

Диаграмата на дейностите (Activity diagram) се използва за моделиране на начина на действието на системата. Могат да се разглеждат като специализации на диаграмите на състоянията. Диаграмите на състоянията описват системите в случай на еднократна употреба, докато диаграмите на дейностите представят последователността от събития, които променят състоянието на системите. Състоянията са дейности. Дейност е изпълнението на задача, например изпълнение на код или някаква физическа дейност.

Диаграмите на дейностите позволяват да се документира логиката на операциите или бизнес процесите в системата. Могат да се асоциират и като обектен еквивалент на структурните диаграми т.е. на блок-схеми на алгоритмите.

Диаграмата на дейности е техника за описание на бизнес процеси, работни потоци и процедурна локика.

Диаграмата на сценарии за използване (Use case diagram) се използва за разясняване на работни изисквания към една система. Чрез тях се описват връзките между участниците и случаите на използване.

Диаграмата на последователност (Sequence diagram) е диаграма на взаимодействие, която представя особеностите на взаимодействие на елементите на моделираната система при определен случай на употреба. Акцент в тези диаграми е редът/последователността на обмяната на съобщения във времето.

Основа за създаване на обектноориентираните модели са класовете. Всеки клас се определя с име, атрибути и операции. Между класовете може да има различни връзки, като асоциации, наследяване и др. Връзките отразяват взаимодействието между обектите и имат собствени характеристики, като кардиналност, роли и др.

Асоциациите моделират отношенията между обекти на класове от един и същи ранг т.е. обекти от различни класове. Допускат се и асоциации между обекти от един клас. Изобразяват се като линии между свързаните обекти. Ако съществуват повече от една асоциация между два класа, тогава асоциациите се именуват. Друг начин на означение са ролите. С тях се описват функциите на обектите в една асоциация. Например между обекти от тип „Книга” и „Потребител” може да има следните роли – читател, доставчик, собственик. Имената на ролите се задават в края на асоциациите, към класа за който се отнасят.

Отношения между цялото и неговите части се наричат агрегации. Те са частен случай на асоциациите между обектите, които се интерпретират като „състои се от” или „част от”.

Броят на обектите, с които един обект от даден клас може да се свързва се начира кардиналност. Тя се интепретира като тип на връзката, например едно към много (1:N), много към много (M:N).

Връзките между класовете се отразяват в диаграмите на класовете. За по-голяма яснота в диаграмите може да се добавят данни, наречени речници от данни и псевдокод. С тази допълнителна информация се описва функционалността на класовете, структурата и типа на атрибутите, типа на параметрите на операциите.