Софийски университет “Св. Климент Охридски”

Факултет по математика и информатика

Магистърска програма “Софтуерни технологии”

Обектно-ориентиран анализ и проектиране на софтуерни системи

Зимен семестър, 2024/2025 год.

Есе

Тема № 33: Поддръжка на потокови диаграми (Data Flow Diagrams) във VP for UML

Автор: Стефан Велев, фак. № 0MI3400521,
магистърска програма “Технологии за големи данни”

декември, 2024 г.
гр. София

СЪДЪРЖАНИЕ

1. Въведение

Концепцията за **потокови диаграми** (Data Flow Diagrams – DFD) навлиза в областта на софтуерното инженерство, благодарение на понятията **структурен анализ** (Structured Analysis – SA) и **структурен дизайн** (Structured Design – SD). Първоначалната идея за възникването на тези две понятия е дошла от нуждата да се направи ясно разграничение между разбирането на проблема и разработването на решение на проблема. В наши дни, обикновено системният анализ (SA) е отговорност на бизнес анализатора, докато фазата на дизайна и проектиране на архитектурата на решението се поема от софтуерния архитект. Ролята на програмиста остава да трансформира този дизайн в код.

Структурният анализ и техниките за дизайн са основни инструменти на системния анализ (Systems Analysis). Структурният анализ става популярен през 80-те години на 20 век и все още се използва в наши дни. Състои се от интерпретиране на системни концепции (или ситуации от реалния свят) в терминологията на данни и контрол, представени чрез потоковите диаграми (още наречени диаграми на потока от данни). Проследяването на потока от данни и контрол, представени с „балончета“ (малки кръгчета), към хранилище на данни и после обратно до „балончето“ може да бъде трудна задача. „Балончетата“ обикновено се групират в по-големи „балончета“ от по-високо ниво, за да се намали сложността.

Диаграмите на потока от данни (DFD) представляват графична репрезентация на потока на данните през информационната система. Те се различават от системните потокови диаграми, тъй като показват този поток на данни от гледна точка на процесите, а не на компютърния хардуер. Самите диаграми са предложени от Larry Constantine – един от основателите на структурния дизайн като методология. Диаграмите са основани на изчислителния модел за „граф на потока от данни“. Тяхното предназначение е да покаже как системата е разделена на по-малки части и да подчертае потока от данни между тези части. Контекстната диаграма представлява потокова диаграма от най-високо ниво на абстракция. Веднъж построена, нейната роля е да се „разгръща“, за да може да покаже повече подробности за системата, която се моделира.

Целта на есето е именно да разгледа **потоковите диаграми** (Data Flow Diagrams) в контекста на средата за моделиране **Visual Paradigm**. Структурата на документа може да бъде обобщена накратко в следните основни точки:

В секция 2. Характеристики и използване на потокови диаграми ще разгледаме в детайли същността на потоковите диаграми. Това се състои в посочване на основните дефиниции и нотация, която се използва в потоковите диаграми. Ще разгледаме диаграмите от гледна точка на техните нива на декомпозиция. Ще акцентираме върху поддръжката на потоковите диаграми във VP и какви са възможните начини за работа с инструмента. Ще споменем за недостатъците и типичните грешки при работа с потокови диаграми (DFD).

В секция 3. Сравнителен анализ ще сравним потоковите диаграми с две други диаграми, типични за Unified Modelling Language (UML) – диаграмата на потребителските случаи (Use Case Diagram) и диаграмата на дейностите (Activity Diagram).

В секция 4. Примери на използване на потокови диаграми ще разгледаме диаграмите на потока от данни от гледна точка на различни домейни, които ще ни помогнат да разберем важността им и тяхната роля.

В секция 5. Добри практики и методи за използване ще разгледаме полезни препоръки и съвети за работа с потокови диаграми в средата на Visual Paradigm и като цяло.

В секция 6. Заключение и очаквано бъдещо развитие ще обобщим значимостта на предмета, който разглеждаме, доколко използваеми са диаграмите в практиката, както и ще дадем възможни насоки за бъдещо развитие.

В секция 7. Използвани литературни източници ще цитираме всички ресурси, които са били необходими на автора за написването на есето.

2. Характеристики и използване на потоковите диаграми

В тази секция ще бъдат разгледани **основни характеристики** на потоковите диаграми (DFD), като в самото начало ще бъдат засегнати някои от основните понятия, които биха помогнали на читателя за по-лесно въведение в темата, след което ще бъде преминато към основните елементи на нотацията.

2.1 Основни дефиниции и нотация

Моделът представлява абстракция на нещо за целите на разбирането му преди да се започне процесът по неговото конструиране. Той е опростена репрезентация на реалността. Ще разгледаме един от подходите за софтуерно моделиране и дизайн – **техниките за моделиране на обекти** (Object Modelling Techniques).

OMT е подход за моделиране на обекти, разработен през 1991 г. от Rumbaugh, Blaha, Premerlani, Eddy и Lorensen, като метод за разработване на обектно-ориентирани системи и за поддръжка на обектно-ориентирано програмиране. OMT е предшественик на Unified Modelling Language (UML), като много от елементите за моделиране на OMT се срещат и в UML.

OMT се състои от три свързани, но различни по своята същност гледни точки, всяка от които улавя важни аспекти на системата.

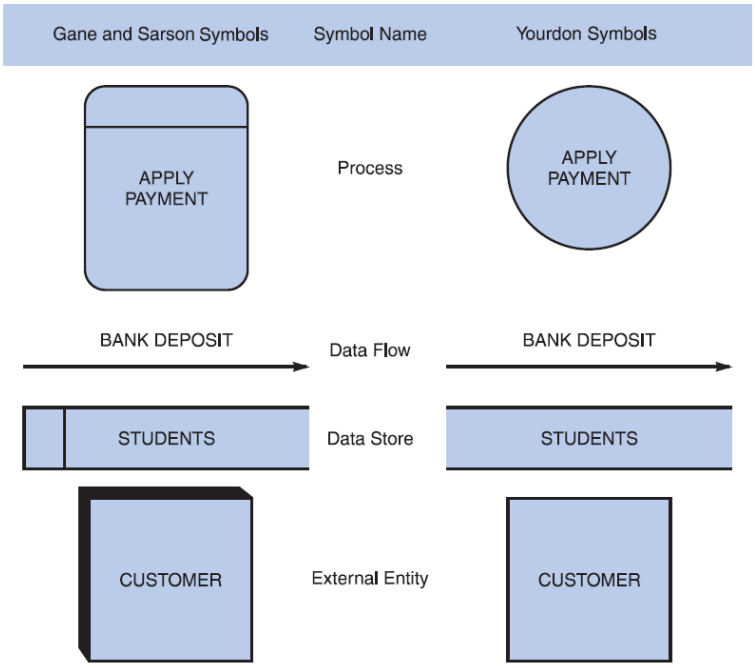
- **Обектният модел** (object model) описва основната структура на обектите и техните взаимовръзки. В него влизат обектните диаграми (object diagrams), които представляват граф, на който възлите са класовете на обектите, а дъгите представляват връзката между класовете.
- **Динамичният модел** (dynamic model) описва аспектите на системата, които се променят с времето. Специфицира и имплементира контролните аспекти на системата. Включва диаграмите на състоянието (state diagrams), които представляват граф, на който възлите са състояния, а дъгите представят преходите между тях.

- **Функционалният модел** (functional model) описва трансформацията на данните в системата. Той представя как входните данни се трансформират в необходимите изходни данни. Включва диаграмите на потока от данни (data flow diagrams – DFD), които представляват граф, на който възлите са процеси, а дъгите представят потока на данните (data flows).

Функционалният модел описва изчисленията и уточнява тези аспекти на системата, които отговарят за трансформации на стойности – функции, преобразувания, ограничения и функционални зависимости. Функционалният модел улавя това, което системата прави, без значение на това как или кога го прави. Представя се графично с множество диаграми на потока от данни, които показват потока от стойности от външни входове, през операции и вътрешни хранилища на данни, към външни изходи. Диаграмите на потока от данни показват зависимостите между стойностите и изчислението на изхода и входните стойности и функции. Функциите се извикват като действия в динамичния модел и се илюстрират като операции на обектите в обектния модел.

Потоковите диаграми (DFD) представят потока на информация в рамките на системата, както и между системата и средата. Те не показват последователността на поведение и на управляваща информация. Потоковите диаграми (DFD) не са част от спецификацията на UML.

Диаграмите на потока от данни (Data Flow Diagrams) използват четири основни символа, които служат, за да означат процеси (processes), потоци от данни (data flows), хранилища на данни (data stores) и външни същности (external entities). Съществуват различни версии на нотацията, но всички те изпълняват една и съща цел. Основните две версии на нотации, на Gane & Sarson и на DeMarco & Yourdon, са показани на долната фигура:



Фигура 1: Символи от потокова диаграма и техните имена за нотации на Gane & Sarson и DeMarco & Yourdon (източник: [7])

Ще разгледаме в детайли всеки един от елементите на нотацията.

- **Процес (Process)**

Всеки един процес получава входни данни и произвежда изход, който има различно съдържание, форма или и двете. Например, процесът на изчисляване на заплата използва два входни елемента (дадена сума на час и брой отработени часове) и произвежда един изход (обща заплата). Процесите могат да бъдат много прости, но и твърде сложни. В една типична компания процесите може да включват изчисляване на заплати, попълване на застрахователни искове, поръчка на оборудване, верифициране на имейл адреси и много други. Процесите съдържат бизнес логиката и бизнес правилата, които служат да трансформират данните и да произведат желания резултат.

Името на процеса идентифицира специфична функция и се състои от глагол, следван от прилагателно и/или съществително име, ако е необходимо. Примери за имена на процеси са: ПРИЛОЖИ ПЛАЩАНЕ ЗА НАЕМ, ИЗЧИСЛИ КОМИСИОННА, НАНЕСИ ФИНАЛНА ОЦЕНКА, ПОТВЪРДИ ПОРЪЧКА, ПОПЪЛНИ ПОРЪЧКА и др. Детайлите по самия процес не се отразяват в потоковите диаграми. Например, ако имаме процес на име ДЕПОЗИРАЙ ПЛАЩАНЕ, това не ни разкрива по никакъв начин бизнес логиката на самия процес. Ако искаме да документираме логиката, то ние трябва да създадем отделно описание на процеса, което само по себе си не е задължителен елемент на диаграмата.

В диаграмата на потока от данни (DFD) за означенията на процеси можем да си мислим като черни кутии, защото входните данни, изходните данни и общите функции на процесите са ясни, но вътрешните детайли и логиката на процеса остават скрити за нас. Това прави диаграмите подходящи за бизнес анализаторите, които искат да покажат системата на определено ниво на детайл, но без да изпадат в структурни подробности. Представянето на диаграмите на различно ниво на детайл позволява информационната система да бъде моделирана като серия от нарастващи по детайл картини.

Пример за устройства от тип „черна кутия“ са мрежовите маршрутизатори (рутери), за които виждаме, че кабелите са носители на входна и изходна информация, но нямаме представа от вътрешните процеси на устройството. Това, което за нас остава очевидно, са резултатите.

- **Поток на данни (Data Flow)**

Потоците на данни представляват пътища, по които данните да могат да се преместят от една част на информационната система до друга. Изобразяваме ги със стрелки. Името на потока на данни може да бъде над, под или до стрелката. Те могат да представят един или повече елемента с данни. Например, един поток на данни може да се състои от един елемент като факултетен номер на студента, или пък да включва множество от данни като факултетен номер, имена, имейл, телефонен номер и др. Самата диаграма не показва в подробности какво включва самият поток, но тази информация се включва в речника на данни (data dictionary).

Имената на потоците от данни обикновено включват съществително име в единствено число и прилагателно име, ако се налага. Примери за подходящи имена са ДЕПОЗИТ, ОЦЕНКА НА СТУДЕНТ, ПОРЪЧКА и др. Тъй като процесите променят съдържанието или формата на данните, то поне една стрелка трябва да влиза и поне една стрелка да излиза от всеки процес.

Възможни са също така повече от един вход и повече от един изход. Възможно е изходните данни от един процес да са вход за друг процес. Задължително условие е поне единия край на поток от данни да бъде процес. Типични грешки при работа с процеси и потоци на данни ще бъдат коментирани в секция 2.3 Ограничения и типични грешки при работа с потокови диаграми.

- **Хранилище на данни (Data Store)**

Хранилищата на данни се използват в потоковите диаграми, за да представят данни, които системата съхранява, защото един или повече процеси се нуждаят от това да ги използват на по-късен етап. Например, компания съхранява данните за заплатите на служителите, за да може да принтира отчети в края на годината. Диаграмата на потока от данни не показва в детайли съдържанието на хранилищата на данни – специфичната структура и елементите се дефинират в речника на данни (data dictionary).

Физическите характеристики на хранилищата на данни не са от значение, защото това, от което се интересуваме, е логическият модел. Също така не се интересуваме от времето, за което ще се съхраняват данните – може да става въпрос за секунди докато се обработва транзакция или за период от месеци, по време на който данните ще се обработват. Това, което остава важно, е че процесите се нуждаят от достъп до данните в определен период от време.

Името на хранилището на данни се намира между двете стени на плоския правоъгълник, с който то се изобразява според нотацията. То се състои от съществително име в множествено число и прилагателно, ако е необходимо. Примерни имена на хранилища на данни са СТУДЕНТИ, АКАУНТИ, ПРОДУКТИ, РАБОТНИЦИ и др. Изключение от правилото за множествено число са общите понятия за структури от подобни елементи, като например КНИГА С ОЦЕНКИ, която съдържа данни за отделните студенти, заедно с техните оценки.

Едно хранилище на данни трябва да бъде свързано с процес чрез поток на данни. Във всеки случай, хранилищата на данни трябва да имат поне един входящ и поне един изходящ поток от данни и да са свързани с процес чрез поток на данни. Две хранилища от данни не могат да бъдат свързани директно едно с друго чрез поток от данни.

Съществува изключение на правилото, че хранилището винаги трябва да има поне един входен поток от данни и поне един изходен такъв. В някои ситуации хранилище на данни няма насочен входен поток от данни към себе си, защото то съдържа фиксирана област от данни, които не се актуализират от системата. Например, хранилището на данни ТАБЛИЦА НА ДАНЪЦИТЕ може да съдържа информация за данъци, които дадена компания само да изтегля от друга система за приходите. В такъв случай, процесът ИЗЧИСЛИ ДАНЪЦИ може само да използва данните от това хранилище, като тогава стрелката на потока от данни ще бъде от ТАБЛИЦА НА ДАНЪЦИТЕ към ИЗЧИСЛИ ДАНЪЦИ.

Хранилищата на данни могат да съдържат буква, към която да се добави номерът, с който се идентифицират. Тези букви характеризират сами по себе си типа хранилище и могат да бъдат:

- „M“ – използва се, където се изобразява ръчно хранилище на данни
- „D“ – използва се, където се изобразява компютъризирано хранилище на данни
- „T“ – използва се, където се изобразява временно хранилище на данни

- **Външни същности (единици) (External Entities)**

Името на същностите се намира вътре в правоъгълника, с който те се изобразяват. В потоковата диаграма (DFD) само външните същности, които предоставят данни на системата или получават изход от системата, се изобразяват. Диаграмата на потока от данни (DFD) показва границите на системата и начините за комуникация с външния свят. Например, клиент, представен като външна единица, предоставя поръчка на система за обработване на поръчки. Други примери за същности включват пациент, който предоставя данни на система за обработване на медицинските записи, или пък домоуправител, който получава сметка от градската данъчна система, или система за плащане на акаунти, която получава данни от друга система за плащане на дадена компания.

Външните единици на потоковата диаграма се наричат още терминатори (terminators), защото те представляват или източници на данните, или финални дестинации. Системните анализатори наричат същност, която предоставя данни към системата, източник (source), докато същност, която получава данни от системата – получател (sink). Името на външната единица е в единствено число и може да бъде елемент от департамент, външна организация, друга информационна система, човек и др. Външните единици могат да бъдат източник, получател или и двете, но всяка единица трябва да бъде свързана с процес чрез поток от данни. Диаграмите на потока от данни (DFD) на по-високо ниво на детайл не могат да въвеждат нови външни същности. Правилни и грешни употреби на външни единици ще бъдат представени в секция 2.3 Ограничения и типични грешки при работа с потокови диаграми.

- **Речник на данни (Data Dictionary)**

Множество потокови диаграми представляват логическия модел на системата, но детайлите по отделните диаграми се документират отделно в речник на данните, който е следващият елемент от структурния анализ. Той се използва от анализатора за събиране, документиране и организиране на специфични факти за системата, включително четирите съставни елемента на потоковите диаграми. Всички те трябва да бъдат подробно документирани, заедно с всякакви допълнителни взаимовръзки. За целта се използват подходящи CASE инструменти за улесняване на задачите, които предлагат определени начини на работа.

2.2 Основни характеристики на потоковите диаграми

2.2.1 Видове потокови диаграми по отношение на нивото на детайлност

Въпреки, че всички потокови диаграми са съставени от едни и същи видове символи и правилата за валидация са еднакви за всички, има три основни типа потокови диаграми:

- **Контекстна диаграма** – диаграми, които представят общ поглед на системата и връзките ѝ с останалата част от света

Контекстните диаграми се използват, за да установят контекста и границите на системата, която ще се моделира – кои неща са част от системата и кои не са, както и каква е връзката на системата с външните същности (единици). Понякога контекстната диаграма се нарича диаграма на потока от данни на ниво 0. Основната ѝ цел е да дефинира и очертае границите на

софтуерната система. Тя идентифицира потока на информация между системата и външните същности. Цялата софтуерна система се представя като един процес.

Първата стъпка за построяването на множество потокови диаграми е направата именно на контекстна диаграма. За целта се започва от поставянето на единичен символ за процес в средата на диаграмата. Този процес представлява цялата информационна система. След това следва поставянето на външни същности (единици) около периметъра на диаграмата, както и потоци на данни, които да свързват същностите с централния процес. Хранилищата на данни не се изобразяват на контекстните диаграми, защото те се съдържат в системата и остават скрити на това ниво на детайл.

Как можем да разберем кои външни същности и потоци от данни трябва да изобразим в контекстната диаграма? Системните изисквания могат да ни помогнат за идентифициране на всички външни източници и получатели. По време на този преглед можем да придобием представа за данните и информацията, която се предава, както и за посоката на предаване. Накрая границите на системата и всички детайли, изобразени на контекстната диаграма, трябва да бъдат дискутирани (и модифицирани, ако се налага) в консултация с нашите клиенти, преди да бъде достигнато до споразумение за продължаване на работата в следващо ниво на детайл.

- **Диаграми на потока от данни от първо ниво** – диаграми, които представят по-детайлен изглед на системата, отколкото контекстните диаграми, като показват основните подпроцеси и хранилища на данни, които образуват системата като цяло

Диаграмите на потока от данни на първо ниво представят всеки от основните подпроцеси, които заедно формират цялата система. Можем да си мислим за тази диаграма като разширена версия на контекстната диаграма. Външните същности не са част от системата и те могат да бъдат отделени чрез пунктиран правоъгълник, който очертава границите на системата. На това ниво на детайл вече разполагаме с хранилища на данни и потоци от данни между процесите и тези хранилища. Всички потоци на данни от външните същности към системата, представени на контекстната диаграма, трябва да бъдат запазени и на това ниво на детайл. Всеки път, когато даден процес се разшири до по-високо ниво, диаграмата на по-високото ниво трябва да показва всички потоци от данни откъм и извън процеса на по-ниското ниво, който се разширява. Преди представяне на системата на това ниво на детайл, трябва да се уверим, че вече съществува контекстна диаграма. За основните подпроцеси в системата отново се обръщаме към системните изисквания.

- **Диаграми на потока от данни на второ (и още по-високо ниво на детайл) ниво** – огромно предимство на техниката за моделиране с потоци от данни е т.нар. представяне на отделни нива – сложността на системите в реалния свят може да бъде обхванатата в детайли и моделирана чрез йерархия от абстракции; определени елементи на потоковата диаграма могат да бъдат декомпозирани в по-детайлен модел в отделно по-високо ниво на диаграмата

С нарастването на разбирането на софтуерните инженери за системата, става необходимо повечето от процесите на първо ниво да бъдат разширени до второ ниво или дори до трето ниво, за да може да се представят на необходимото ниво на детайл. Декомпозицията се прилага

върху всеки процес от потоковата диаграма на първо ниво, за който има достатъчно скрит детайл вътре в него. Всеки процес от второ ниво също трябва да бъде проверен за декомпозиция и т.н. Процес, който не може да се декомпозира повече, може да се маркира със звездичка в долния десен ъгъл. Кратко описание на всеки процес трябва да бъде представено допълнително с диаграмата на потока от данни. Обикновено процесите, които се нуждаят от декомпозиране, са такива със сложни наименования или пък такива, които имат повече от определен брой (обикновено шест) връзки за поток от данни от и към себе си.

Стъпките за декомпозиция са подобни на тези при декомпозицията на първо ниво. Важно е да се отбележи, че избраният процес за декомпозиране от първо ниво става новата граница на системата (изобразена с пунктиран правоъгълник) на потоковата диаграма от следващото ниво.

2.2.2 Декомпозиране на нива

За всички системи е препоръчително изобразяването на поне две нива на абстракция – контекстната диаграма и диаграма на потока от данни от първо ниво. Независимо от броя им обаче, те трябва да бъдат синхронизирани, т.е. корекции на едната диаграма водят до потенциални такива и в другата.

Какви могат да бъдат причините, за да правим диаграми на следващо по-високо ниво на детайл? Първата причина е по-очевидната – не сме описали напълно даден процес, така както искаме и следователно се налага неговото разпадане на подпроцеси. Новата диаграма е изградена точно по същия начин, по който първо ниво се изгражда от контекстната диаграма само че новите входове и изходи са точно за потоците от данни към и от процеса, който разширяваме.

Втората причина е, ако осъзнаваме, че диаграмата, върху която работим, става претрупана и неясна. За да опростим диаграмата, събираме заедно няколко от процесите. В идеалния случай тези процеси трябва да са свързани по някакъв начин. Заменяме ги с един процес и разглеждаме оригиналната колекция от процеси като на по-високо ниво, разширяващи новия процес. Важно е да бъдем внимателни с входовете и изходите на новия процес. Трябва да не забравяме да преномерираме старите процеси, за да покажем подходящо, че те са преместени едно ниво нагоре. Не трябва да групираме случайни процеси заедно само за да направим диаграма от друго ниво. Това само по себе си би довело до множество стрелки и несвързани процеси. Добър ориентир за това дали сме избрали разумна колекция е да опитаме да измислим ново име за заместващия процес. Ако се затрудняваме да направим това, то вероятно сме направили твърде общо групиране. Винаги трябва да имаме предвид, че групирането по нива има за цел да опрости и изясни диаграмите, като ако това не може да бъде направено по никакъв начин, то по-добре да оставим потоковата диаграма на нивото на абстракция, на което вече е.

2.2.3 Синхронни и асинхронни операции

Съществуват начини в диаграмите на потока от данни (DFD), по които да покажем, че два процеса си комуникират синхронно (с изчакване) или асинхронно (без изчакване). Ако два процеса са директно свързани чрез стрелка, обозначаваща потока на данни, то тогава те са

синхронни. Това означава, че те работят с еднаква скорост. Ако два процеса са свързани чрез хранилище на данни, то тогава тяхната скорост на оперирање остава независима. В такъв случай, казваме, че двата процеса са асинхронни.

2.2.4 Множество копия на външни същности и хранилища на данни на една диаграма

Понякога една потокова диаграма може да бъде по-ясна, ако поставим повече от едно копие на определена външна единица или хранилище на данни на различни места. Това може да доведе до решаване на проблема с преплитането на множество стрелки. Когато се нуждаем от повече от едно копие на външна същност, ние го представяме с допълнителна наклонена черта (подобна на тази при бизнес актьорите и случаите на употреба) в горния ляв ъгъл. Ако искаме да изобразим повече от едно копие на хранилище на данни, то поставяме вертикална черта в левия ъгъл на съответния символ (в зависимост от избраната нотация).

2.2.5 Балансиране

Ключът към успешното групиране по нива е балансирането на диаграмите. Например, ако диаграма от второ ниво разширява процес от първо ниво, тогава всички входове към процеса трябва да бъдат входове на диаграмата от второ ниво и всички изходи от процеса трябва да бъдат изходи на диаграмата от второ ниво. Освен това не трябва да има други входове и изходи. За да бъдем по-конкретни, всички входове и изходи на системата, които се появяват на контекстната диаграма, трябва да се изобразяват на диаграмата от първо ниво и на нея не трябва да има други входове и изходи.

Това не означава, че не може да има промени в по-ниските нива на дадени диаграми. Когато създаваме диаграма от по-високо ниво, може да разберем, че е необходим нов вход, за да може процесът да изпълни задачата си. В този случай трябва да добавим този поток от данни като вход и веднага след това да добавим входа като поток от данни към оригиналния процес. Ако е необходимо, този вход може да се представи на няколко нива по-високо. По същия начин могат да се добавят и нови изходи.

2.2.6 Номериране (индексиране)

Номерирането в набор от диаграми на различни нива е важно, тъй като номерацията ни помага по-лесно да се ориентираме. Нека да представим идеята с пример. Да предположим, че ПОЛУЧИ ПОРЪЧКА е процес, номериран с 3 на диаграмата от ниво едно (номерата на процесите не показват определен ред, а са само етикети) и се разширява до диаграма от ниво две. Номерата на процесите на диаграмата от ниво две ще бъдат 3.1, 3.2, 3.3 и т.н. Да предположим сега, че процес 3.4 на диаграмата от ниво две е РЕГИСТРИРАЙ НОВ КЛИЕНТ и се нуждае от допълнително разширение до диаграма от ниво три. Номерата на процесите на следващата диаграма ще бъдат 3.4.1, 3.4.2, 3.4.3 и т.н. Използваното тук правило е следното: ако X е номерът на процеса, който искаме да разширим, то тогава номерата на следващото ниво са X.1, X.2, X.3 и т.н.

Същото важи и за хранилищата на данни. Хранилищата на данни, които се появяват в диаграма от ниво две, разширяваща процес, обозначен с 4 в диаграмата от ниво едно, ще бъдат номерирани D4.1. D4.2, D4.3 и така нататък. По-дълбоките нива ще бъдат D4.1.1, D4.1.2, като

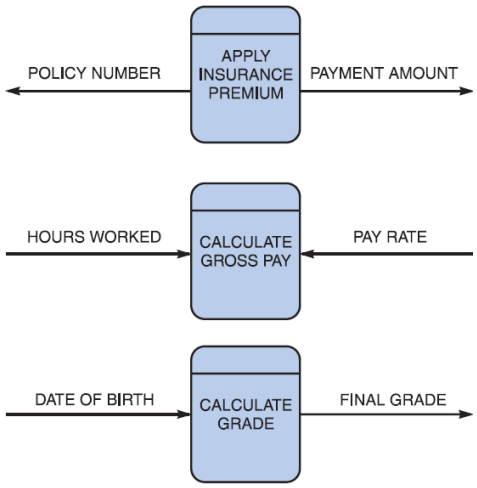
схемата за номериране е същата като за процесите. Важно е да се посочи обаче, че хранилищата на данните сами по себе си не се разпадат, а само се появяват в разширяването на процеси.

2.3 Ограничения и типични грешки при работа с потокови диаграми

Сред основните недостатъци на потоковите диаграми са, че те все пак оставят достатъчно възможности за неточност. Контролните аспекти (control flows), които представляват условие или събитие (не включва данни), задвижващо даден процес, не могат да бъдат представени с тяхна помощ. Съществуват нотации, които го позволяват чрез пунктирана стрелка, но това е извън стандартното разбиране за диаграма на потока от данни (DFD). Друга тяхна минусова черта е това, че те не са еднозначни по отношение на репрезентация на елементите и съществуват няколко алтернативи.

Съществуват редица типични грешки по отношение на отделните елементи от потоковите диаграми. Ще ги илюстрираме чрез примери.

- **Типични грешки по отношение на процесите:**

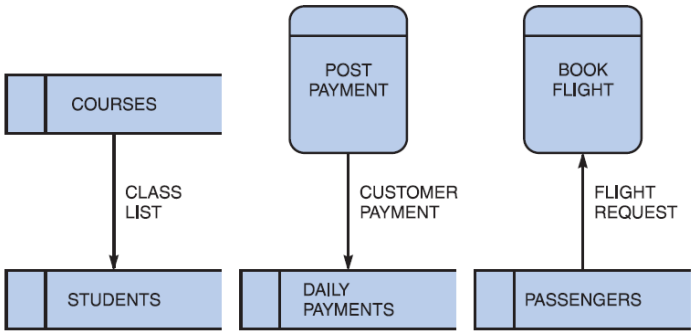


Фигура 2: Типични грешки по отношение на процесите и връзката им с потоците от данни (източник: [7])

Горните случаи са примери за следните категории грешки:

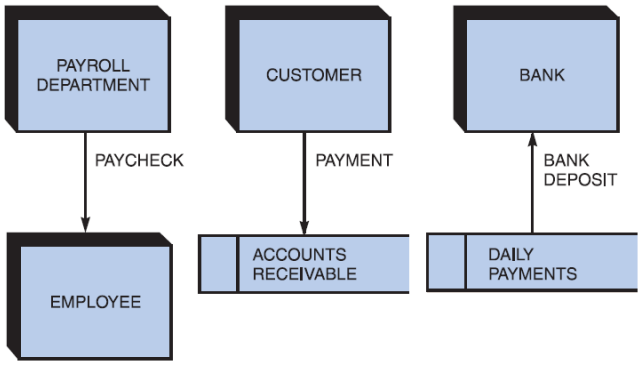
- **Процес на спонтанно генериране** – произвежда изход, но няма зададен входен поток от данни
  Пример: APPLY INSURANCE PREMIUN
- **Черна дупка** – процес, който има вход, но няма изход
  Пример: CALCULATE GROSS PAY
- **Сива дупка** – процес, който има поне един вход и един изход, но входът очевидно е недостатъчен, за да генерира дадения изход
  Пример: CALCULATE GRADE

- **Типични грешки по отношение на хранилищата на данни:**



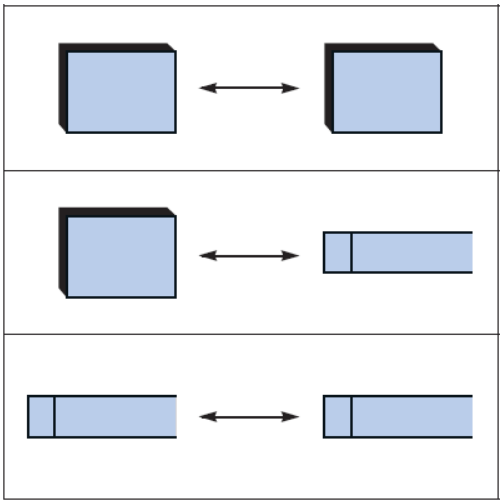
Фигура 3: Типични грешки по отношение на хранилищата на данни (източник: [7])

- **Типични грешки по отношение на външните същности (единици):**



Фигура 4: Типични грешки по отношение на външните същности (източник: [7])

- **Типични грешки по отношение на потоците от данни:**



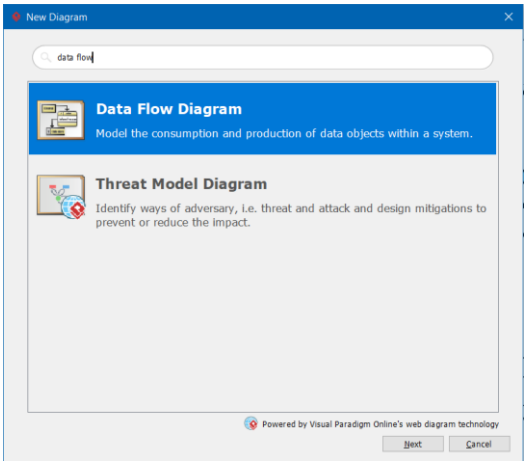
Фигура 5: Типични грешки по отношение на потоците от данни (източник: [7])

2.4 Потоковите диаграми и тяхната поддръжка във Visual Paradigm

Колкото по-сложна е системата, толкова по-трудно е да се поддържа пълна и точна документация. За щастие, съществуват съвременни CASE инструменти, които да опростят тази задача. Един от тях е софтуерната платформа **Visual Paradigm**.

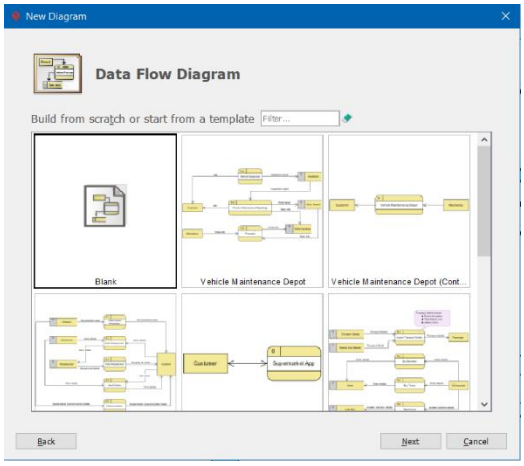
Поддръжката на потоковите диаграми (Data Flow Diagrams) съществува в следните продукти на **Visual Paradigm**: VP Enterprise, VP Professional, VP Standard, VP Modeler. Всички те са платени, като предлагат безплатен 30-дневен пробен период за оценяване. В безплатния продукт за некомерсиална употреба **Visual Paradigm Community** потоковите диаграми не се поддържат като функционалност.

Създаването на диаграма на потока от данни (DFD) става от менюто ***Diagram → New***. Оттам се избира опцията ***Data Flow Diagram***.



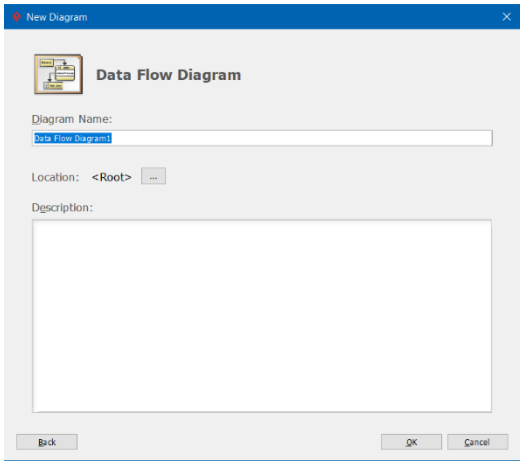
Фигура 6: Създаване на потокова диаграма (стъпка 1)

Следващата стъпка е да изберем празен шаблон за нашата диаграма или да се спрем на някой от предложените такива на готови потокови диаграми.



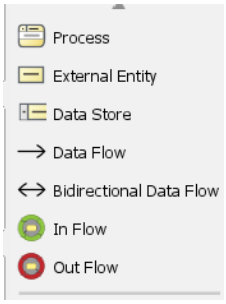
Фигура 7: Създаване на потокова диаграма (стъпка 2)

Последната стъпка е да зададем име на диаграмата, да я позиционираме спрямо работната директория и по желание да ѝ зададем описание.



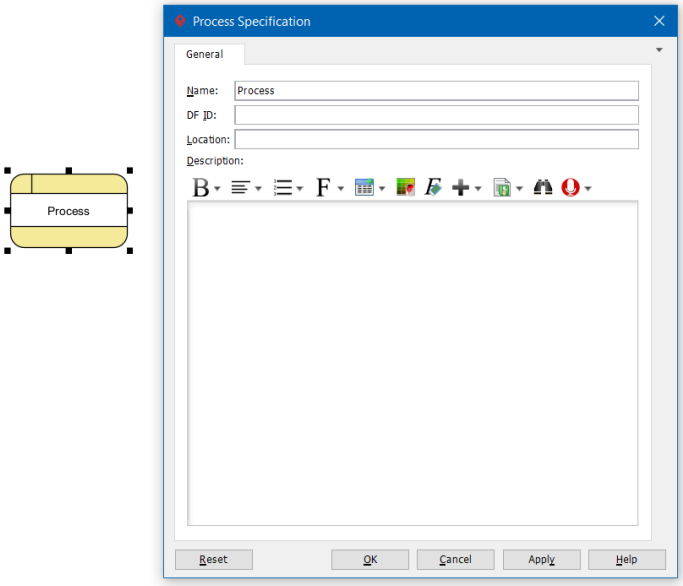
Фигура 8: Създаване на потокова диаграма (стъпка 3)

Самата диаграма разполага със следните графични елементи, които съответстват на основната нотация на диаграмите на потока от данни (забелязваме, че предпочетената от създателите на ***Visual Paradigm*** нотация в потоковите диаграми е тази на Gane & Sarson):



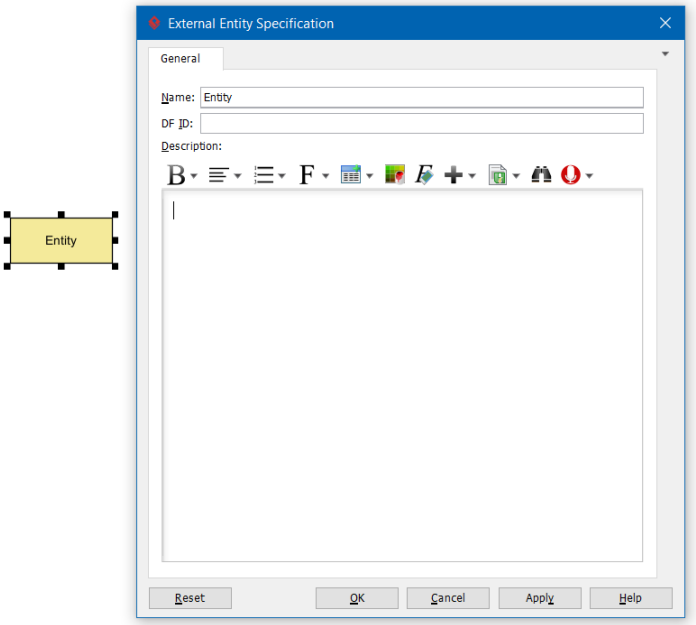
Фигура 9: Нотация на диаграма на потока от данни във Visual Paradigm

Всеки един процес може да бъде специфициран чрез няколко характеристики – ***име***, ***уникален номер***, ***местоположение*** и ***описание***:



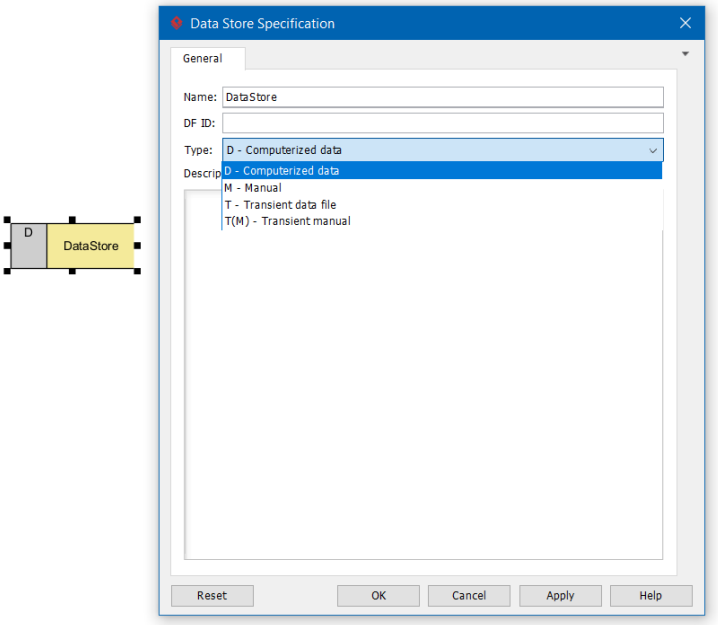
Фигура 10: Спецификация на процес във Visual Paradigm

Външните същности (единици) могат да бъдат специфицирани чрез полетата ***име***, ***уникален номер*** и ***описание***.



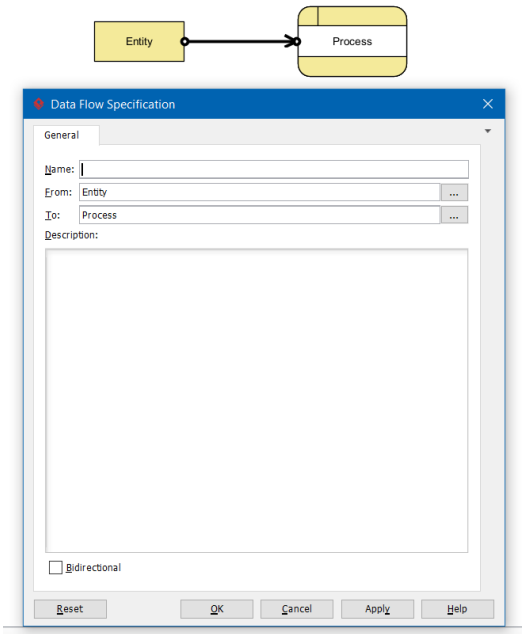
Фигура 11: Спецификация на външна същност (единица) във Visual Paradigm

Хранилищата на данни могат да бъдат специфицирани чрез полетата ***име***, ***уникален номер***, ***тип (Computerized data, Manual, Transient data file, Transient manual)*** и ***описание***.



Фигура 12: Спецификация на хранилище на данни във Visual Paradigm

Потокът от данни може да бъде еднопосочен или двупосочен. И в двата случая той може да бъде специфициран чрез неговите полета ***име***, ***начало***, ***край***, ***описание*** и ***опция за двупосочност на връзката***.



Фигура 13: Спецификация на поток на данни във Visual Paradigm

Елементите ***In Flow*** и ***Out Flow*** служат за връзка между родителска диаграма към съдържанието в диаграмите наследници и обратно.

В секция 4. Примери на използване на потокови диаграми ще демонстрираме цялостно завършени примери в средата на ***Visual Paradigm***.

3. Сравнителен анализ

Диаграмите на потока от данни (DFD) са мощен инструмент за визуализация на процесите в системата и как данните преминават през тях. Те предоставят яснота относно взаимодействията между различните компоненти на системата, което ги прави изключително полезни за анализ и проектиране. Основното предимство на DFD е способността им да показват потока от информация без излишна сложност, като се фокусират върху логиката на процесите. Въпреки това, в рамките на UML (Unified Modelling Language) съществуват други диаграми, като **диаграмите на случаите на употреба** (Use Case Diagrams) и **диаграмите на дейностите** (Activity Diagrams), които могат да изпълняват сходна функция, предлагайки някои допълнителни перспективи.

3.1 Критерии за сравнение

Критериите за сравнение между диаграмите на случаите на употреба и диаграмите на дейностите в UML могат да бъдат разгледани през следните аспекти: използваната нотация, която влияе на яснотата и стандартизацията; предимствата, които всяка диаграма предлага спрямо описването на различни аспекти на системата; недостатъците, свързани с тяхната

приложимост; ограниченията, които могат да възникнат при използването им в различни етапи от разработката; приликите, които осигуряват съвместимост и лесно интегриране. Тези критерии позволяват ефективен анализ и информиран избор на подходящ инструмент според нуждите на проекта.

3.2 Сравнение с диаграмите на случаите на употреба (UML Use Case Diagrams)

Диаграмите на случаите на употреба са част от спецификацията на UML. Те представят реалните софтуерни изисквания на дадена система. Подходящи са за моделиране на изискванията на системата, като показват взаимодействието между външни актьори и самата система. Те са полезни за улавяне на функционалността от гледната точка на потребителя, което ги прави лесноразбираеми за бизнес потребители и клиенти. Чрез визуализиране на връзките между актьорите и случаите на употреба, те предоставят ясна представа за основните функции и зависимости, което подпомага комуникацията между техническите екипи и заинтересованите страни.

Таблица: Сравнение на диаграмите на потока на данни с диаграмите на случаите на употреба

Критерий	**Диаграма на потока от данни (Data Flow Diagrams)**	**Диаграми на случаите на употреба (Use Case Diagrams)**
История	предложени от Larry Constantine (1970)	предложени от Ivar Jacobson (1986)
Изглед	функционален изглед от гледна точка на системата	функционален изглед от гледна точка на актьори
Нотация (компоненти)	процеси, външни същности, потоци от данни, хранилища от данни	актьори, потребителски случаи, асоциации, граница на системата
Предимства	+ лесна за разбиране графична техника + ясно очертава границите на системата + подходяща за представяне пред заинтересованите лица + ясно обяснява логиката, свързана с потока на данни в системата	+ достатъчно ясно обхваща функционалностите на цялата софтуерна система в една диаграма + съдържа обяснения на естествен език + предлага по-богат инструментариум за представяне на връзките
Недостатъци	– претърпява доста промени – не разглежда физическите ограничения – объркваща на пръв поглед за програмистите – различни нотации	– не могат по лесен начин да бъдат зададени нефункционални изисквания – остава по-трудна за четене за нетехнически лица

Ограничения	• не задава последователност на действията • липсват времеви интервали на процесите • не показва колко пъти даден процес се повтаря	• не задава последователност на действията • липсват времеви интервали на процесите • не изобразява хранилища на данни
Сходства	външна същност (единица)	актьор
	процеси	потребителски случаи
	поток на данни	асоциация

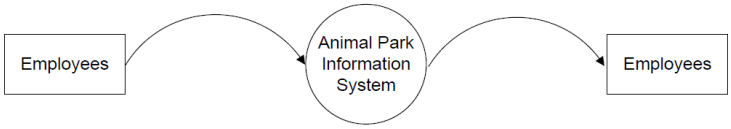
3.3 Сравнение с диаграмите на активностите (UML Activity Diagrams)

Диаграмите на дейностите в UML са средство за моделиране на последователността от действия и потоци в процеси или случаи на употреба. Те визуализират както контролния поток, така и потока от данни, което ги прави подходящи за представяне на динамичното поведение на системата. Благодарение на лесно разпознаваемата си нотация с действия, преходи и условия, те са ефективни за анализиране на сложни процеси и изясняване на сценарии за изпълнение.

Диаграмите на дейностите и диаграмите на потока от данни (DFD) имат различни приложения и подходи за визуализация на процеси в системата. Диаграмите на дейностите използват стандартизирана UML нотация, която акцентира върху последователността на дейностите, условията и началните/крайните точки, докато DFD разчитат на символи за процеси, потоци от данни, хранилища на данни и външни същности. Едно от основните предимства на диаграмите на дейностите е тяхната способност да представят динамичното поведение на системата, докато диаграмите на потока от данни са изключително ефективни за илюстриране на трансформациите и движението на данни. Въпреки това, потоковите диаграми не показват времеви зависимости за разлика от диаграмите на дейностите. Те пък от своя страна могат да станат твърде сложни за големи системи. И двата типа диаграми обаче споделят силата да визуализират процеси, което ги прави достъпни както за технически, така и за нетехнически заинтересовани страни.

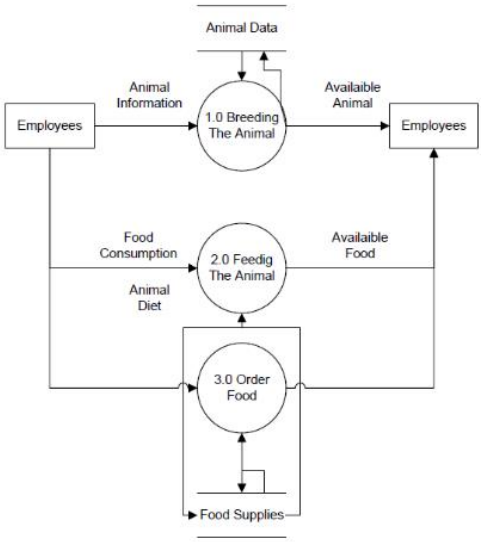
Ще разгледаме този сравнителен анализ под формата на казус (източник: [8]). Ще построим различни диаграми, свързани с процеса на размножаване в зоологическата градина в Somerleyton, England, като така ще илюстрираме различните типове означения. Сценарият на процеса на размножаване се отнася до процеса на развъждане на животни. Системният администратор проверява в системата кое животно е готово за размножаване и проверява дали в парка има партньор, с когото може да се размножава. Когато животното и неговият партньор са на разположение, тогава извършваме размножаването с помощта на животновъд. Ако няма подходящ партньор в парка, тогава системният администратор трябва да се свърже с друга зоологическа градина, която може да има подходящо животно. В такъв случай, той трябва да уреди график кога и къде ще се извършва размножаването.

Да разгледаме контекстната диаграма на системата. Първата външна единица представя работника като системен администратор, който работи директно с информационната система на зоологическата градина. Втората външна единица представя работник в ролята си на животновъд, който се грижи за животните.



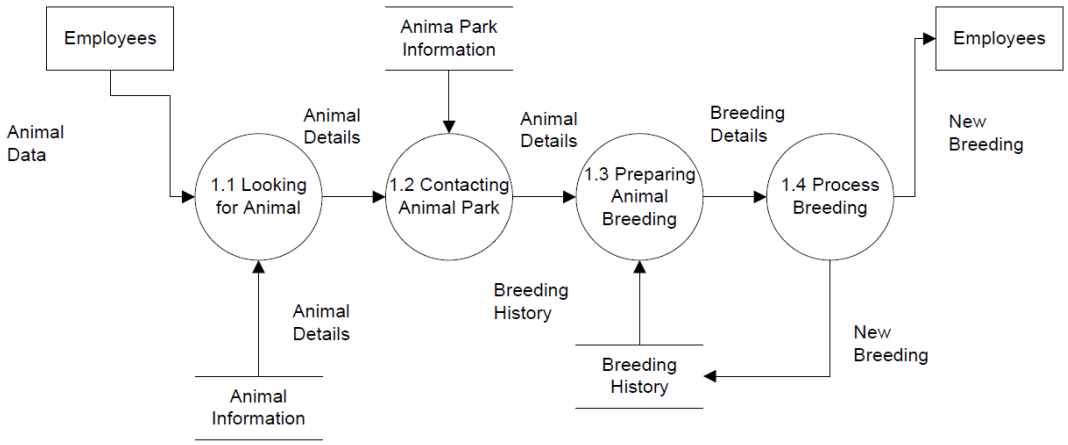
Фигура 14: Контекстна диаграма на информационна система на зоологическа градина (източник: [8])

В диаграмата на потока на данните от следващото ниво можем да видим как системата е разделена на три подпроцеса: (1.0) РАЗМНОЖИ ЖИВОТНИТЕ, (2.0) НАХРАНИ ЖИВОТНИТЕ и (3.0) ПОРЪЧАЙ ХРАНА.



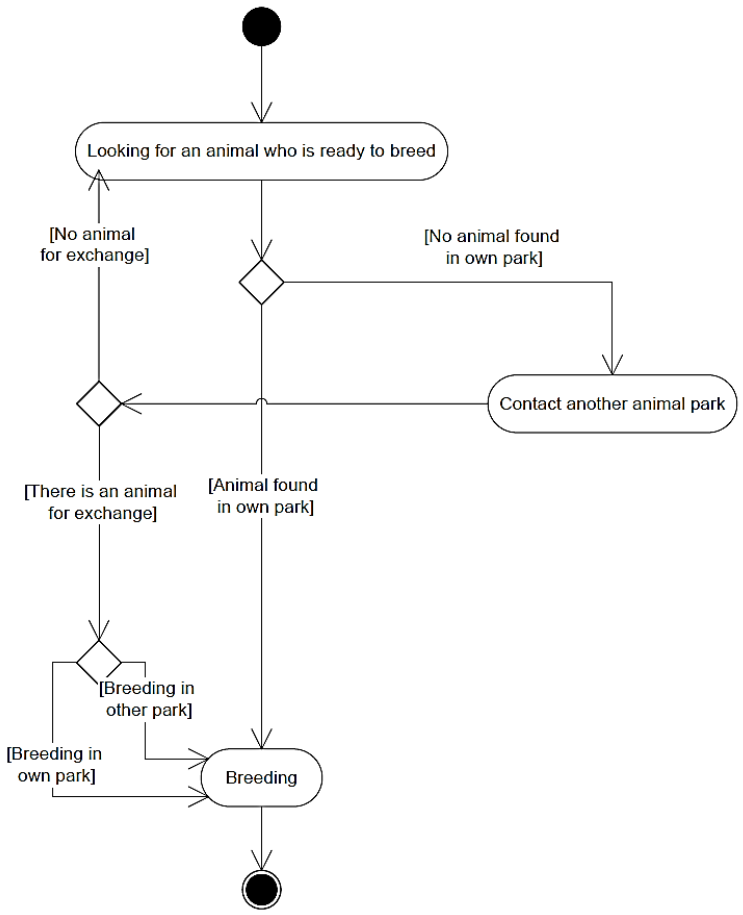
Фигура 15: Диаграма на потока от данни от първо ниво на информационна система на зоологическа градина (източник: [8])

Процесът, който искаме да декомпозираме на следващо ниво, е (1.0) РАЗМНОЖИ ЖИВОТНИТЕ. Виждаме, че той се състои от четири подпроцеса. Забелязваме, че входните и изходните потоци на данни се запазват дори и след декомпозицията. Това беше едно от правилата за балансиране на потоковите диаграми.



Фигура 16: Диаграма на потока от данни от второ ниво на информационна система на зоологическа градина (източник: [8])

Нека сега да разгледаме диаграма на дейностите за същия процес по размножаване на животните. Тя се състои от три дейности – търсене на животно, което е готово за размножаване, самото размножаване и установяването на контакт с друга зоологическа градина.



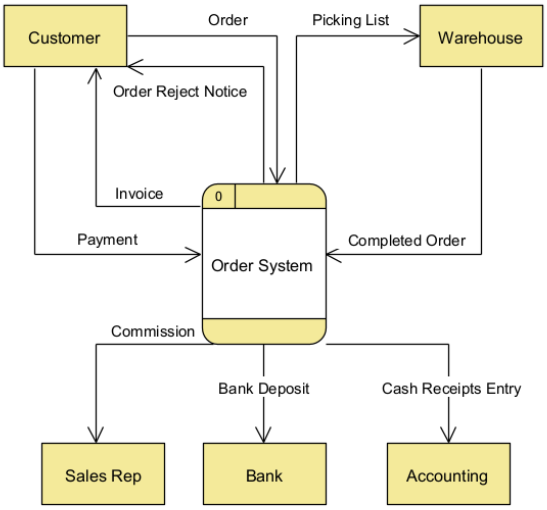
Фигура 17: Диаграма на дейностите за процеса по размножаване на животните (източник: [8])

4. Примери на използване на потокови диаграми

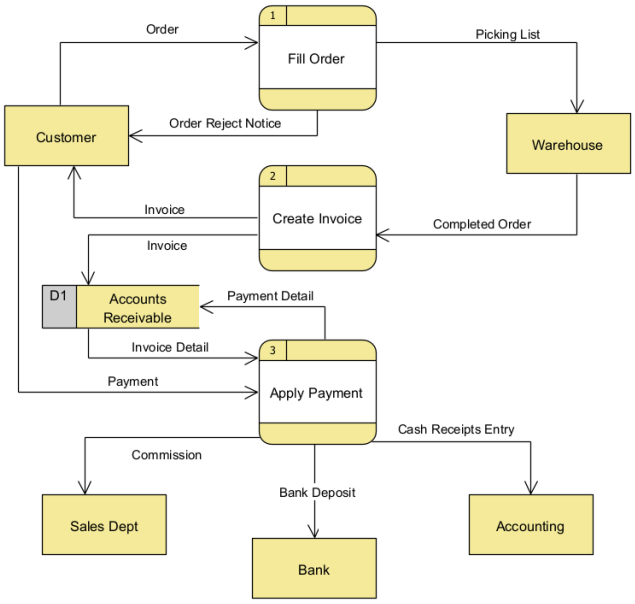
Ще разгледаме два примера на диаграмите на потока от данни в средата на ***Visual Paradigm***.

4.1 Пример 1 – Система за поръчки (източник: [7])

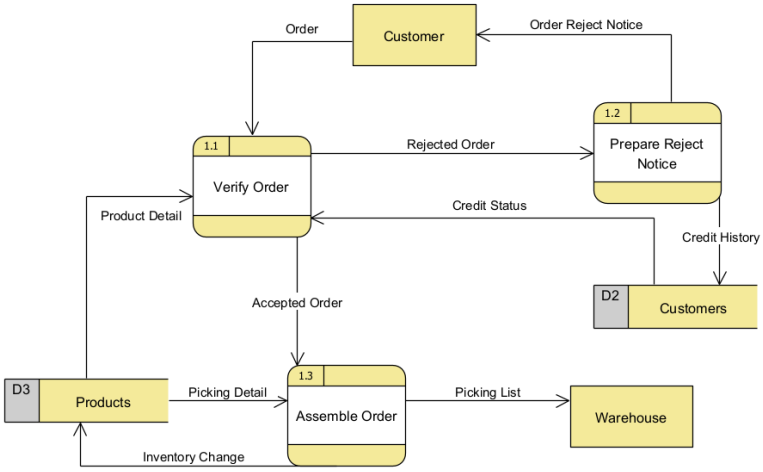
Разглежданият пример представя няколко потокови диаграми на различни нива на детайлност. Забелязва се, че принципите за декомпозиция по нива и принципите за балансиране са спазени.



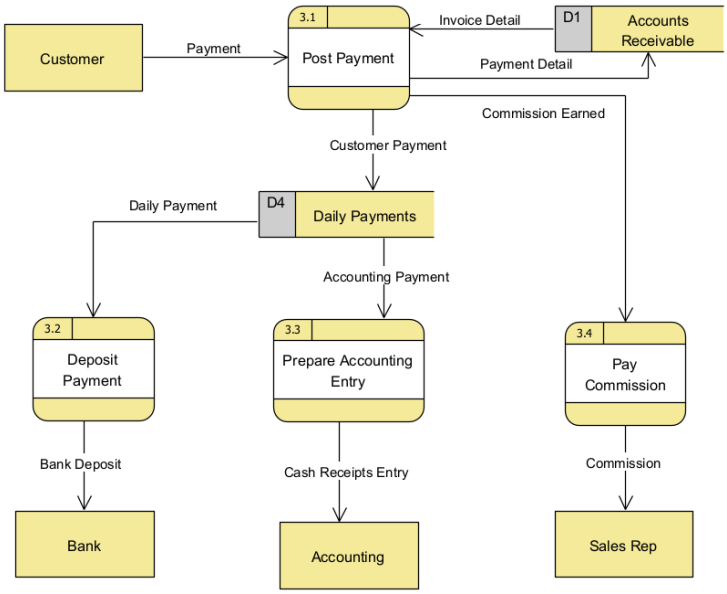
Фигура 18: Контекстна диаграма за система за поръчки



Фигура 19: Диаграма на потока от данни от първо ниво за система за поръчки



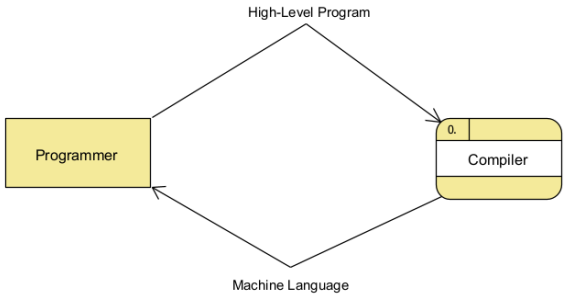
Фигура 20: Диаграма на потока от данни от второ ниво на процеса по попълване на поръчка от системата за поръчки



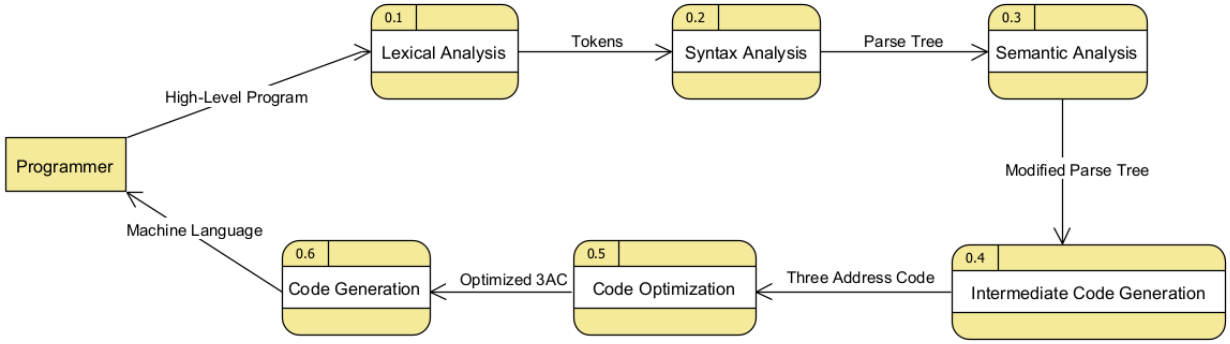
Фигура 21: Диаграма на потока от данни от второ ниво на процеса по прилагане на плащане от системата за поръчки

4.2 Пример 2 – Компилатор (източник: [6])

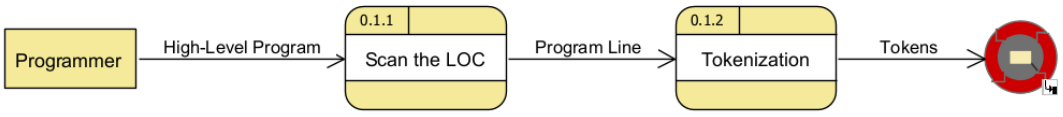
Разглежданият пример представя няколко потокови диаграми на различни нива на детайлност. Забелязва се, че принципите за декомпозиция по нива и принципите за балансиране са спазени.



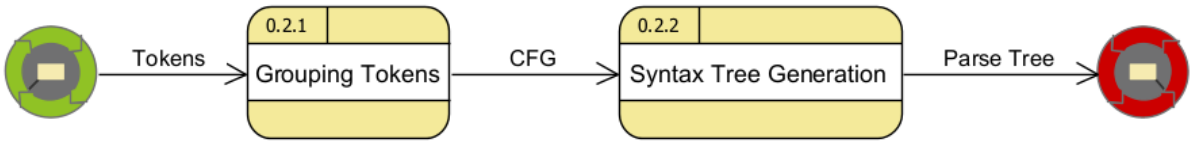
Фигура 22: Контекстна диаграма за компилатор



Фигура 23: Диаграма на потока от данни от първо ниво за компилатор



Фигура 24: Диаграма на потока от данни от второ ниво за процеса по лексикален анализ на компилатор

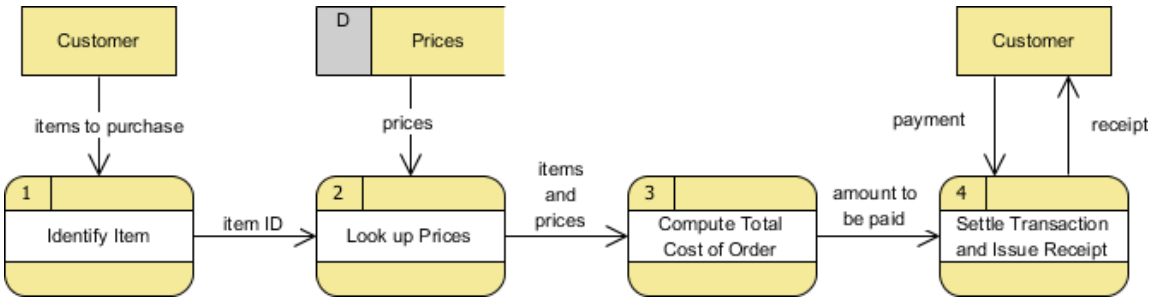


Фигура 25: Диаграма на потока от данни от второ ниво за процеса по синтактичен анализ на компилатор

5. Добри практики и методи за използване

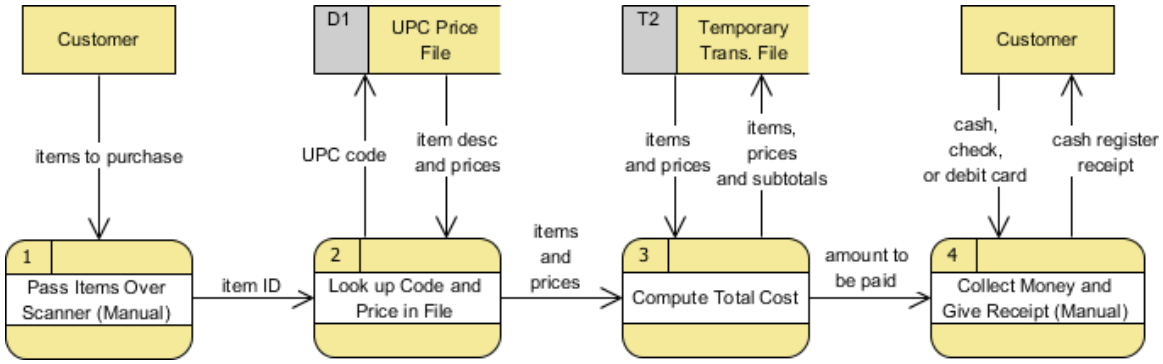
Сред основните предимства на потоковите диаграми са, че те са прости за разбиране и използване. Позволяват ни да разберем функциите и границите на системата, като ни предоставят възможност за преминаване към по-голяма степен на детайл. Те използват много малко примитивни елементи за представяне на функционалностите на системата и потокът на данни сред тези функции. Тога ги прави лесни за разбиране дори и от заинтересовани лица, които не познават нотацията или нямат техническо образование. Като всеки друг основен вид диаграми, те могат да бъдат част от официалната документация.

В зависимост от това какво искаме да представим с потоковите диаграми, съществуват два отделни вида – логически (Logical DFD) и физически (Physical DFD) потокови диаграми. **Логическите потокови диаграми** се фокусират основно върху системния процес, който илюстрира как данните протичат в системата. Диаграмата на логическия поток от данни се фокусира главно върху процесите на високо ниво и потока от данни, без да се гмурка надълбоко в детайлите на техническото изпълнение. Използват се в различни организации за нормалното функциониране на системата, за да опишат как данните се преместват от един обект към друг.



Фигура 26: Логическа диаграма на потока от данни от първо ниво за система за касово плащане в супермаркет (източник: [9])

Физическите потокови диаграми показват как потокът от данни всъщност е имплементиран в системата. В диаграмата на физическия поток от данни ние включваме допълнителни детайли като съхранение на данни, предаване на данни и специфични технологии или системни компоненти. Те са по-специфични и по-близки до етапа на имплементация.



Фигура 27: Физическа диаграма на потока от данни от първо ниво за система с касово плащане в супермаркет (източник: [9])

6. Заключение и очаквано бъдещо развитие

В заключение, **потоковите диаграми** (Data Flow Diagrams) представляват един от най-ефективните инструменти за визуално представяне на процеси и потоци от данни в различни области на дейност. Те осигуряват яснота, структурираност и лесна интерпретация на сложни системи, като спомагат за подобряване на комуникацията между екипите, оптимизиране на работните процеси и вземане на информирани решения. В практиката тези диаграми намират широко приложение в отделните фази на софтуерното инженерство, управлението на проекти, бизнес анализа и много други сфери, като подпомагат както начинаещи, така и опитни специалисти.

Развитието на потоковите диаграми може да се свърже най-вече с тяхната интеграция с облачни технологии и колаборативни платформи. Това би позволило на екипи, работещи дистанционно, да създават, редактират и споделят диаграми в реално време, като същевременно се осигурява проследимост на промените и сравнение с други диаграми и варианти на процесите.

7. Използвани литературни източници

[1] **Aleryani**, Arwa. *Comparative Study between Data Flow Diagram and Use Case Diagram.* Saba University, Yemen, 2016. ISSN 2250-3153. Available from: https://www.researchgate.net/publication/313808834_Comparative_Study_between_Data_Flow_Diagram_and_Use_Case_Diagram

[2] **Bentley**, Lonnie, **Whitten**, Jeffrey. *Systems Analysis and Design Methods*. 7th Edition: McGraw-Hill/Irwin, 2007. ISBN 978-0-07-305233-5

[3] **Blaha**, Michael, **Rumbaugh**, James. *Object-Oriented Modeling and Design with UML™*. 2nd Edition: Pearson Prentice Hall, 2005. ISBN 0-13-015920-4

[4] **Constantine**, Larry, **Yourdon**, Edward. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. 2nd Edition: YOURDON Press, 1978. ISBN 0-917072-11-1

[5] **GeeksForGeeks:** *What is DFD (Data Flow Diagram)?* [online]. [Viewed 22.11.2024]. Available from: https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/

[6] **Jacob**, Pramod. *Object-Oriented Modelling and Design: Concepts, Notes & Solved Questions*. M. Tech SOFTWARE ENGINEERING [online]. [Viewed 22.11.2024]. Available from: https://kailash392.wordpress.com/wp-content/uploads/2018/11/oomd-notes-question-bank-by-pramod.pdf

[7] **Rosenblatt**, Harry, **Shelly**, Gary. *Systems Analysis and Design*. 9th Edition: Course Technology, Cengage Learning, 2012. ISBN 978-0-538-48161-8

[8] **Tangkawarow**, Irene, **Waworuntu**, J. *A Comparative of business process modelling techniques*. IOP Conference Series: Materials Science and Engineering, 128 012010, 2016. Available from: https://iopscience.iop.org/article/10.1088/1757-899X/128/1/012010

[9] **Visual Paradigm:** *What is Data Flow Diagram?* [online]. [Viewed 22.11.2024]. Available from: https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/