Структурен подход –
структурен анализ и структурно проектиране

гл. ас. д-р Цветелина Кънева

tskaneva@uni-ruse.bg

Структурен подход

Top-down refinement approach

Структурен подход

(на англ. *Structured Analysis and Structured Design, SASD*)

Структурният подход се концентрира върху вътрешната структура и организация на системата. Той разглежда системата като съвкупност от взаимосвързани компоненти, които трябва да бъдат проектирани по определен начин, за да функционират ефективно заедно.

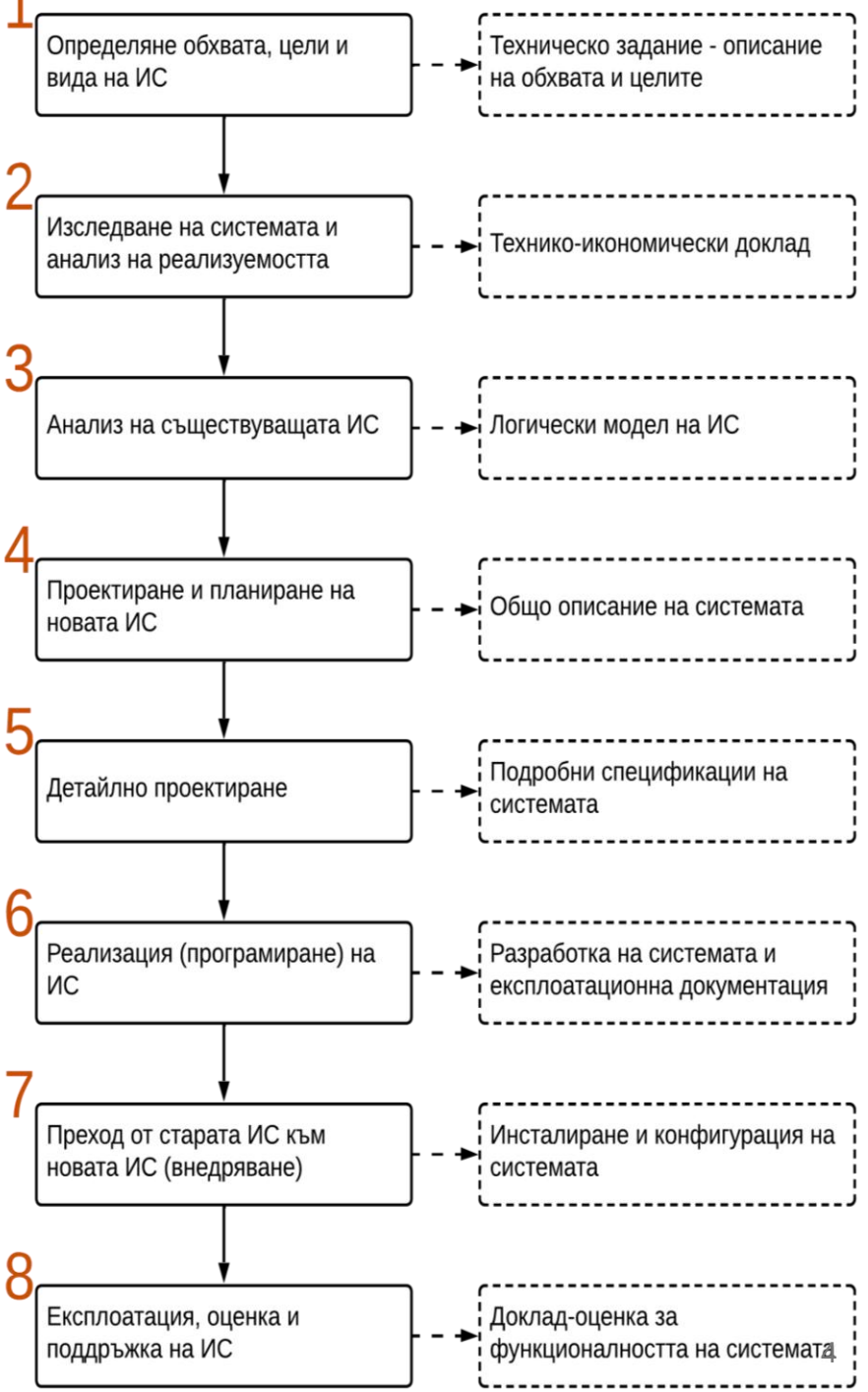
Целта на структурния подход е да се осигури **логическа** и **физическа** структура на системата, която да бъде **стабилна**, **ефективна** и **лесна за поддържане** и **разширяване**.

Използват се аналитични и формални методи, като моделиране на данни, диаграми на потоци, и други техники за проектиране на структурата на системата.

Акцентът е върху създаването на модели като **диаграми на потоците от данни** (на англ. ***Data Flow Diagrams, DFD***), диаграми на класове (на англ. *class diagrams*) или релационни схеми (чрез ER диаграми).

Жизнен цикъл при структурния подход

Всеки етап приключва с резултати, които трябва да бъдат постигнати, за да се премине към следващия етап.



Жизнен цикъл – етапи

- Разделяне на сложен процес на подпроцеси – лесно управление
- По-лесна координация и независима разработка по време на етапа
- Добра документация – основа за всеки следващ етап и отчетност
- Етапите представляват „естествено“ разпределение на дейностите
- Етапите позволяват плавно нарастване на разходите по изпълнение на проекта

Жизнен цикъл – етап 1

Определяне на обхвата, целите и вида на ИС

1. Определяне на структурата на организацията – организационен модел
2. Определяне на обхвата и областта на действие на организацията
   - Характерните процеси на предметната област;
   - Характерните процеси за конкретната организация;
   - Ресурсите, с които разполага организацията;
   - Законови норми и ограничения на организацията;
3. Определяне на целите – KPI показатели
4. Определяне на проблемите
   - Процесите, които пораждат проблема;
   - Процесите, които са засегнати от проблема;
   - Входните данни на проблемните процеси;
   - Изходните данни от проблемните процеси;
   - Използваните ресурси;
   - Служителите, участващи в проблемните процеси.

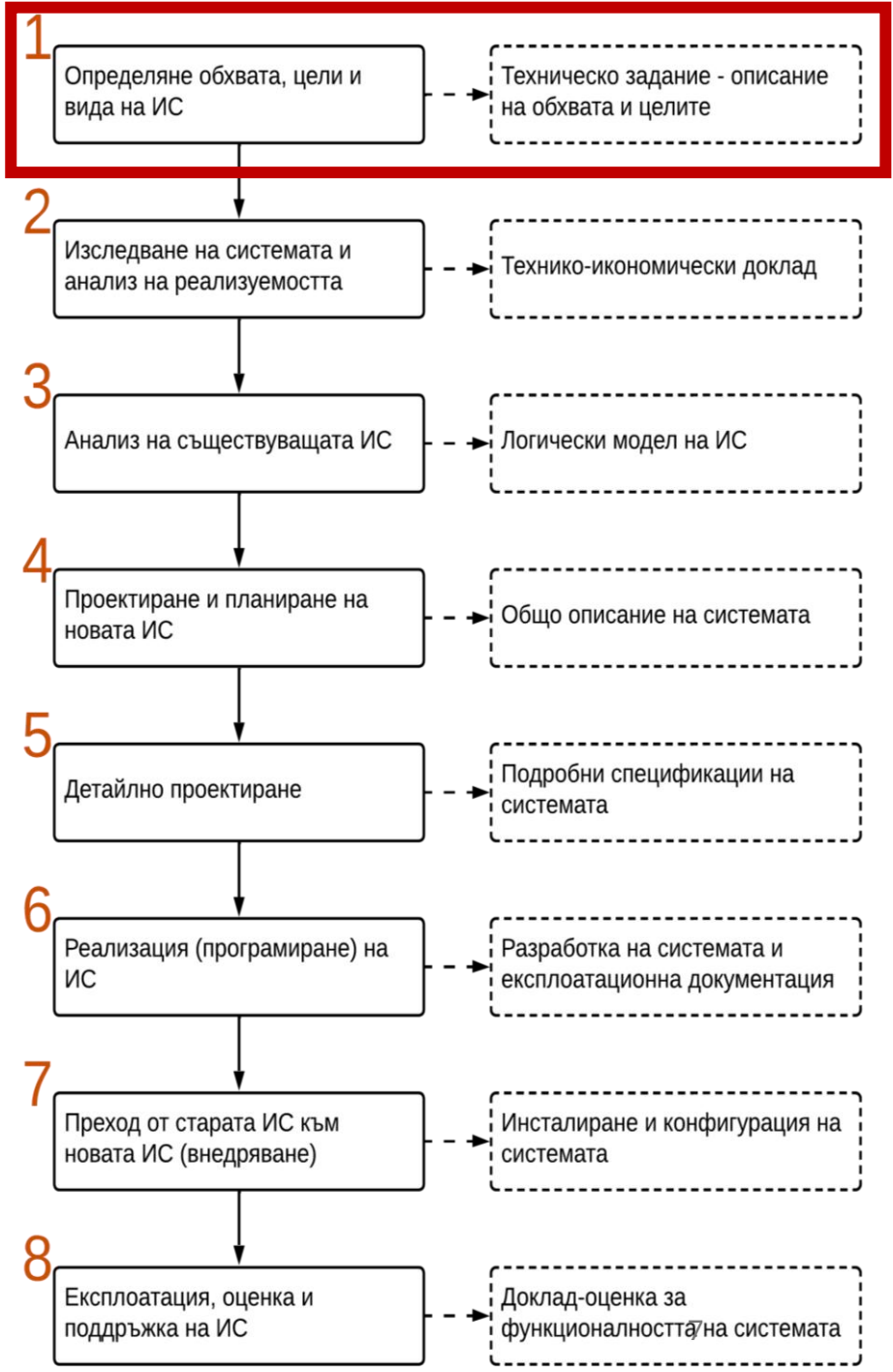
Структурен подход

Жизнен цикъл – етап 1

Определяне на обхвата, целите и вида на ИС

Резултат – техническо задание:

- Структурата на организацията;
- Обхвата и областта на действие на ИС;
- Целите, които трябва да бъдат постигнати с ИС;
- Проблемите, които ИС трябва да разреши;



Жизнен цикъл – етап 2

Изследване на системата и анализ на реализуемостта

На този етап системният аналитик трябва да:

- Определи целите на съществуващата система;
- Принципа на функциониране на съществуващата система;
- Законови норми и разпоредби, касаещи организацията и системата;

“Защо трябва да се замени съществуващата система?”

1. Интервюта със служителите
2. Прочуване на съществуващата документация
3. Наблюдения
4. Анкетни карти
5. Измервания

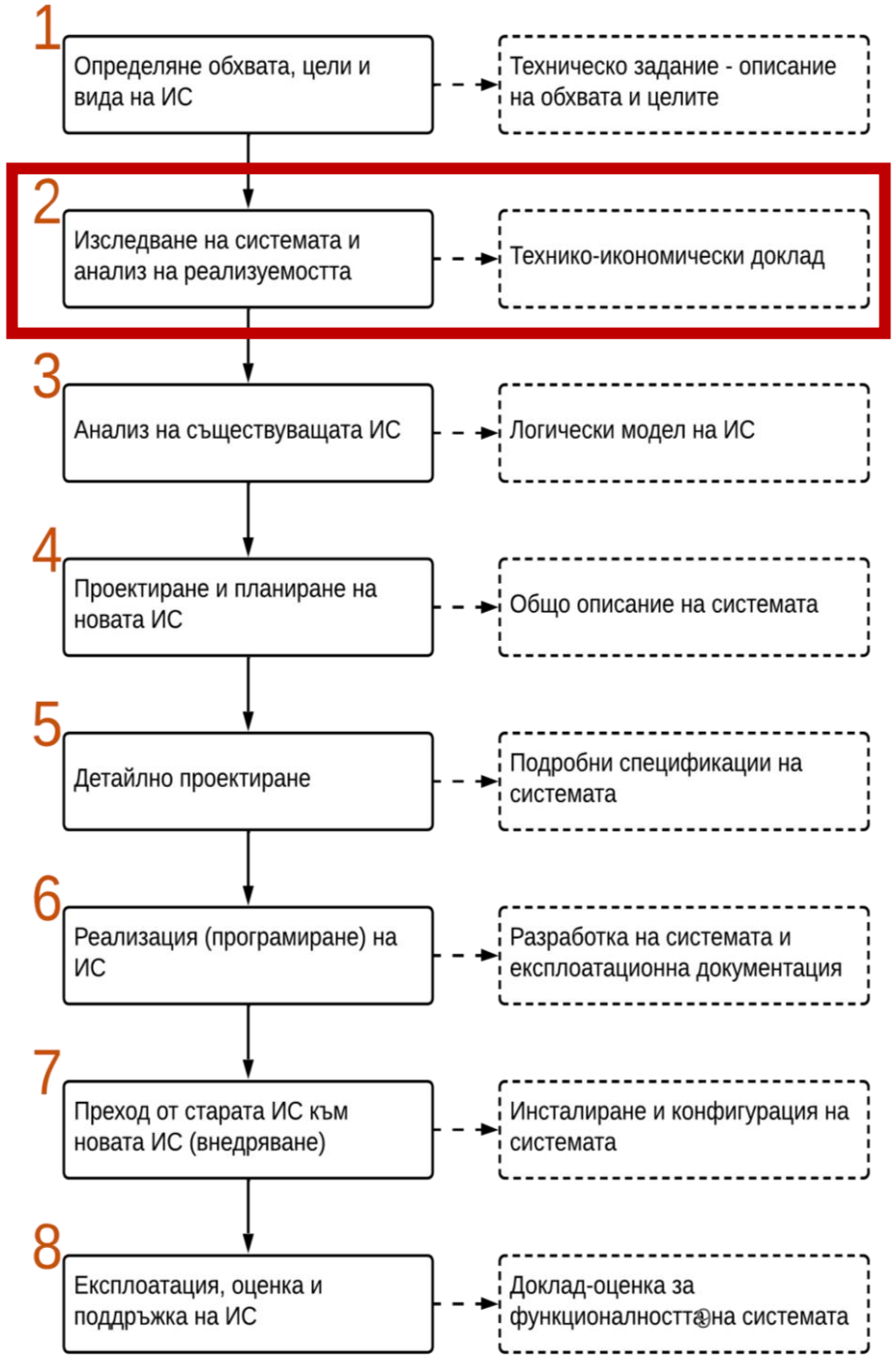
Структурен подход

Жизнен цикъл – етап 2

Изследване на системата и анализ на реализуемостта

Резултат – технико-икономически доклад:

- Оценка на реализуемостта;
- Предложени решения, на базата на етап 1 и етап 2 и описани с обобщени показатели:
  - Оценка на разходите;
  - Ползи;
  - Реализуемост.



Жизнен цикъл – етап 3

Анализ на съществуващата информационна система

Целта на анализа на съществуващата система е да се открият както проблемите, така и областите, в които могат да се въведат подобрения и нововъведения.

Декомпозиране на функционалностите на старата система е метод за изследване, който е приложим на този етап.

На този етап се създават логически модели на процесите и потоците от данни.

Добра практика е да се направи и анализ на потока от физически документи, които служителите обменят между процесите и отделите.

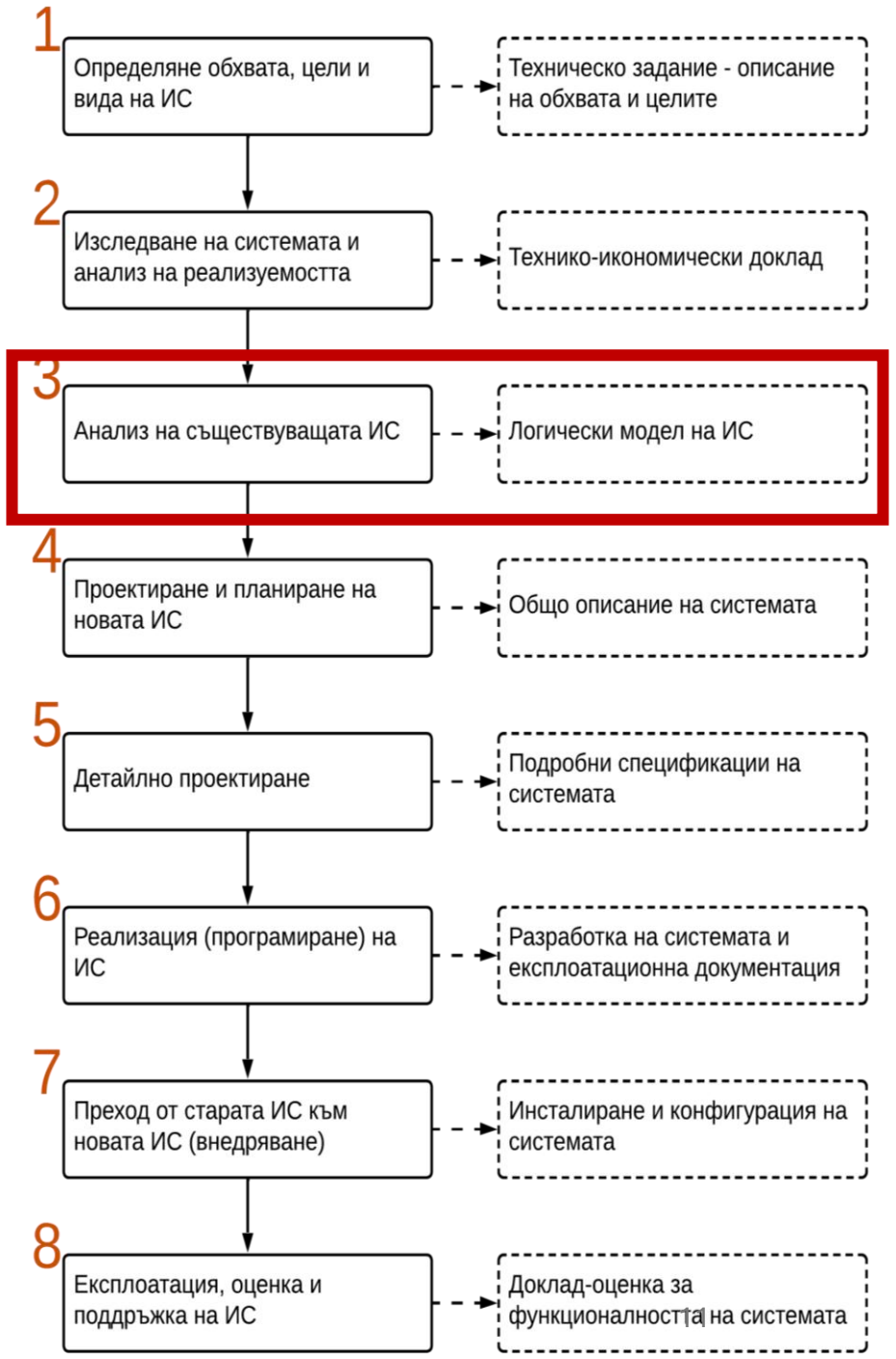
Структурен подход

Жизнен цикъл – етап 3

Анализ на съществуващата информационна система

Резултат – логически модел на старата ИС:

- Диаграми;
- Блок-схеми;
- ДПД;
- Речници на данни;



Жизнен цикъл – етап 4

Проектиране и планиране на новата система

Избор на архитектура на системата:

- Централизирана или разпределена архитектура
- Клиент-сървър архитектура

Проектиране на базата данни:

- Тип на базата данни
- Механизми за архивиране и възстановяване на данни

Сигурност на системата:

- Механизми за защита на данните
- Механизми за удостоверяване и авторизация

Проектиране на потребителския интерфейс (UI):

- Сценарии на потребителите

Планиране на разработката и внедряването:

- Определяне на ключови етапи (на англ. milestones)
- Управление на риска

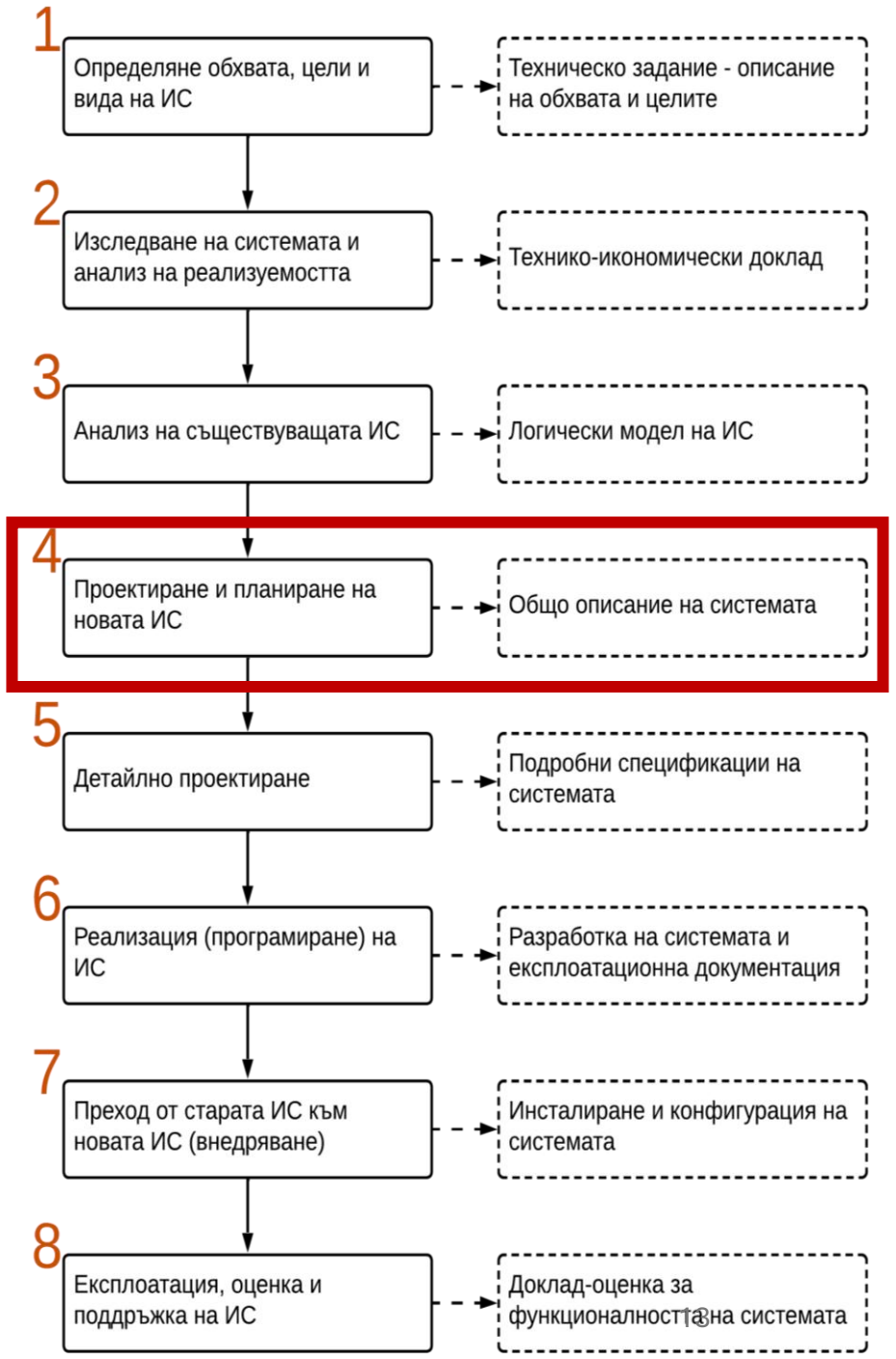
Структурен подход

Жизнен цикъл – етап 4

Проектиране и планиране на новата система

Резултат – общо описание на новата ИС:

- Няколко алтернативни решения за избор
- Диаграми – use-case, ER, др.;
- Блок-схеми;
- ДПД;
- Речници на данни;



Жизнен цикъл – етап 5

Детайлно проектиране

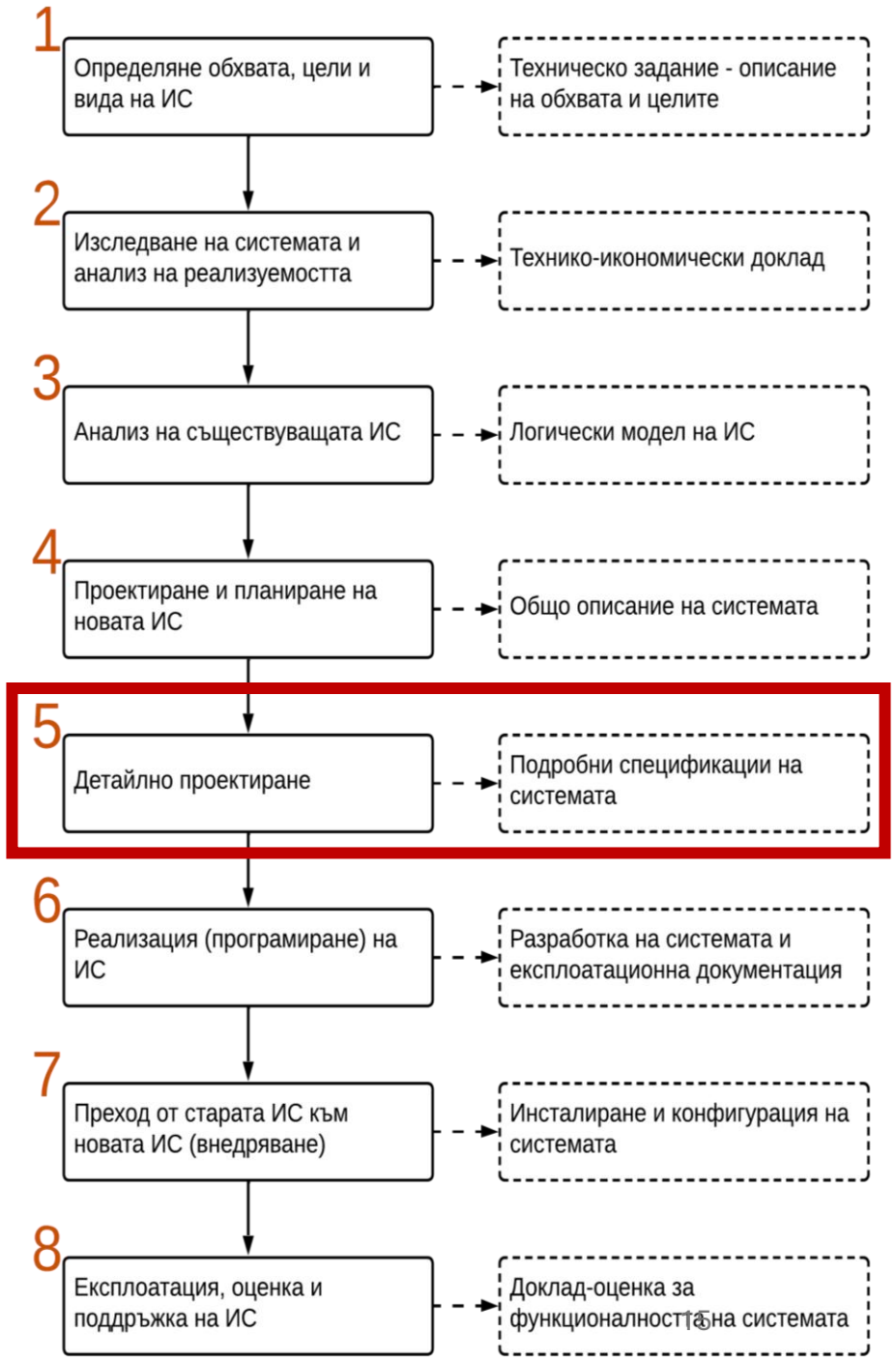
1. Определяне на базата данни или файловата система;
2. Спецификация на хардуерните изисквания;
3. Определяне на програмните модули, които да се разработят от програмистите;
4. Определяне на план-график за разработка и внедряване.
5. Проектиране на моделите на данни, посредством ER диаграми;
6. Определяне на механизми за имплементация на сигурност на системата;
7. Дефиниране на ключови моменти за потребителския интерфейс.

Структурен подход

Жизнен цикъл – етап 5

Детайлно проектиране

- Описание на предлаганата системата;
- Описание на функциите, изпълнявани в системата – функционални изисквания;
- Описание на програмното осигуряване – описания на модулите, структурни диаграми;
- Спецификация на входа – описание на формата на входните документи, вида на информационните екрани, процедури за контрол, др.
- Спецификация на изхода – описание на отчетите, съобщения от системата, изходни документи;
- Описание на съхраняваните данни – спецификация на файловата система и/или базата от данни;
- Спецификация на контролните средства;
- Спецификация на изискванията към хардуера;
- Спецификация на нефункционалните изисквания;
- Спецификация на служителите и техните отговорности;
- График за реализиране на системата;
- Оценка на разходите.



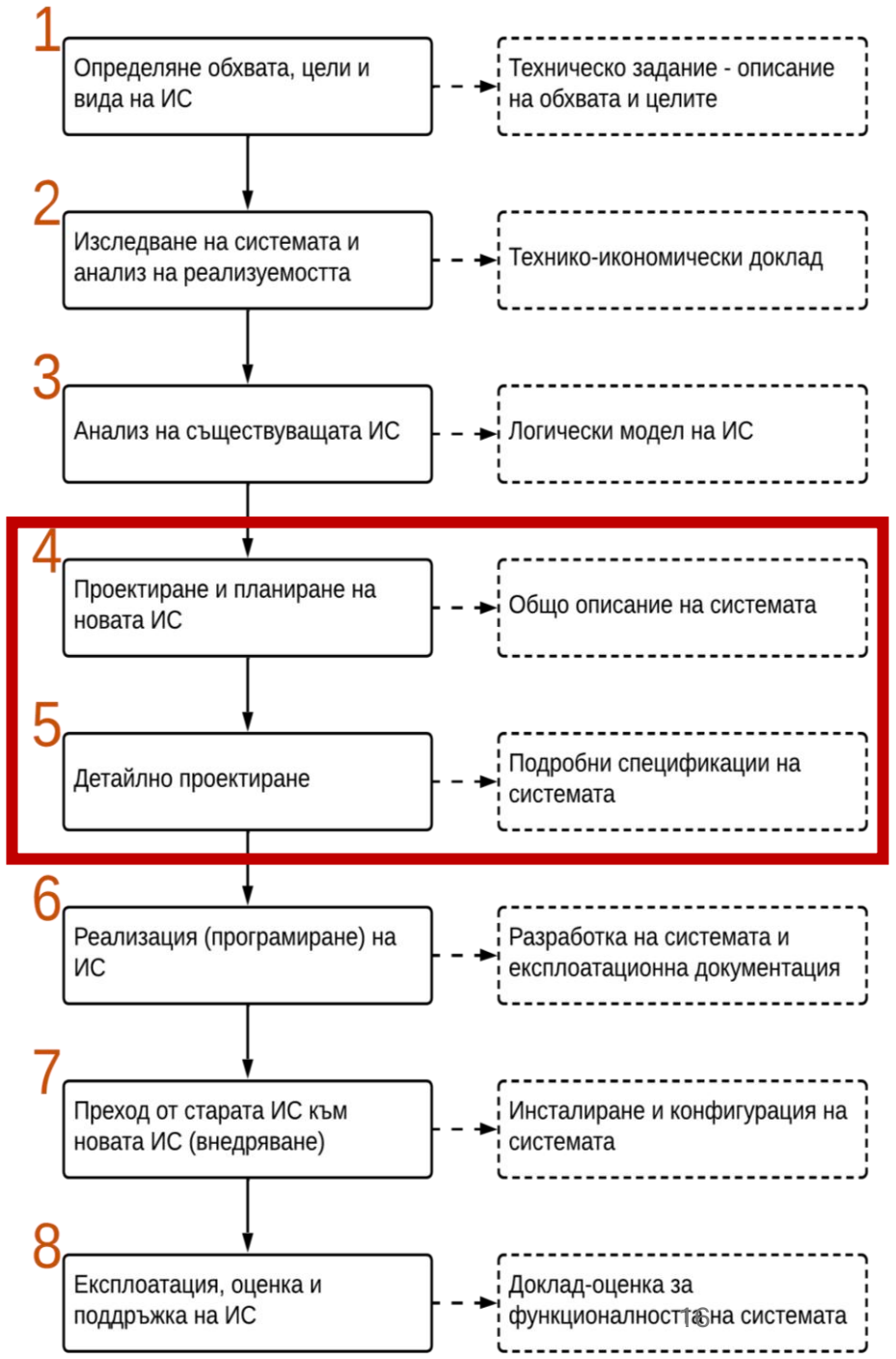
Структурен подход

Жизнен цикъл – етап 4 и етап 5

Разлики между етапите

Етап 4 се занимава с високото ниво на архитектурата и основните концептуални решения за системата, докато Етап 5 е свързан с конкретизацията и специфицирането на всички детайли, нужни за реалната разработка и изпълнение на системата.

Етап 4 е за стратегическо планиране, а Етап 5 е за оперативно изпълнение.



Жизнен цикъл – етап 6

Реализация на ИС

1. Най-скъп и най-дълъг;
2. Закупуване на необходимия хардуер и софтуер;
3. Програмиране на системата;
4. Тестване;
5. Корекции;
6. Обучение.

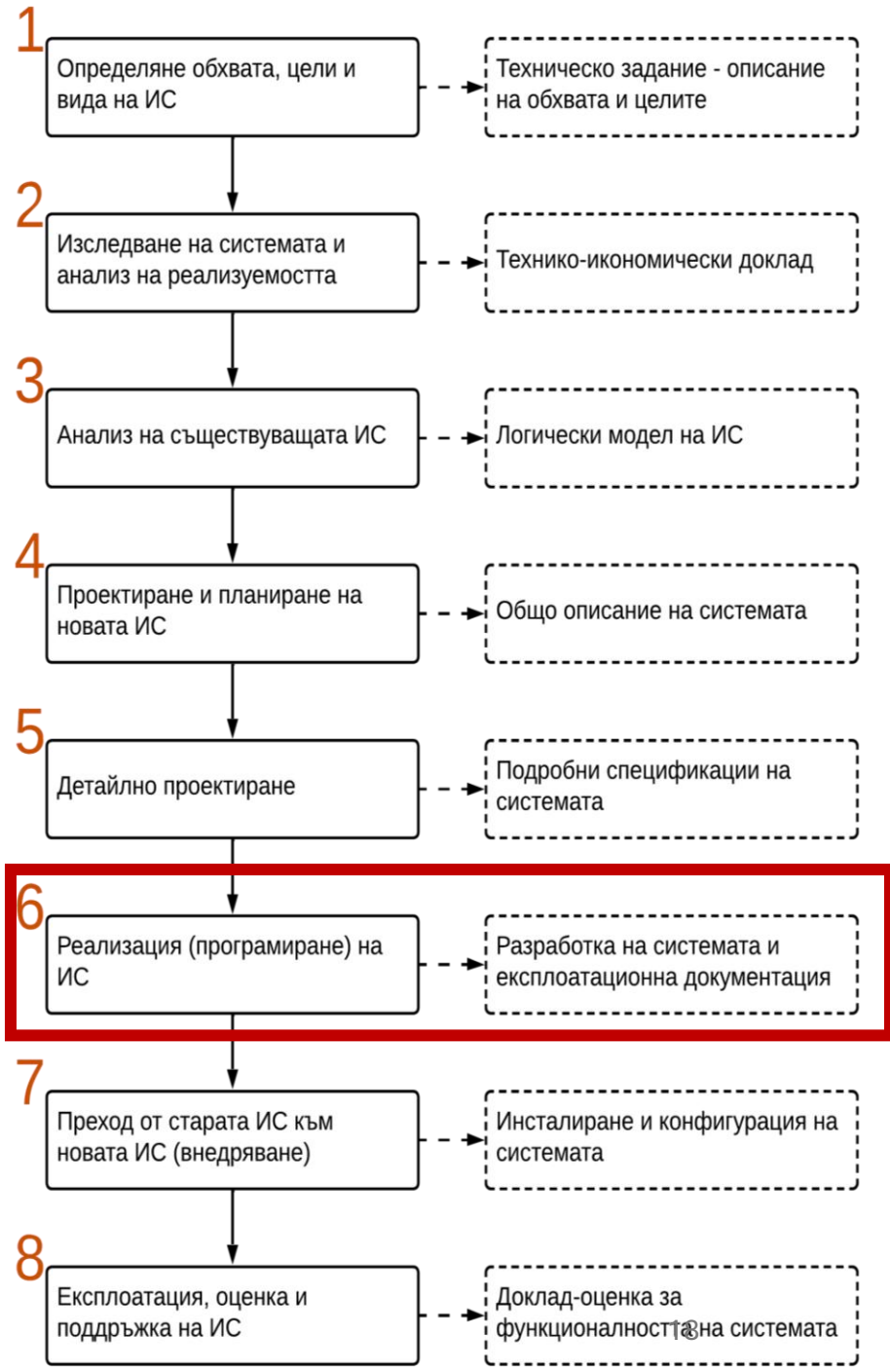
Структурен подход

Жизнен цикъл – етап 6

Реализация на ИС

Резултат – разработка, документация:

- Разработена ИС;
- Експлоатационна документация;
- Техническа документация.



Жизнен цикъл – етап 7

Преход от старата ИС към новата ИС

Преходът от старата към новата система може да бъде осъществен по един от четирите начина:

1. **Директна смяна** - директно спиране на съществуващата система и смяна с новата. За да бъде безпроблемна подобна смяна, етапите по анализ, проектиране, разработка и обучение трябва да са извършени безупречно, така че потребителите да нямат проблеми с новата система.
2. **Паралелен подход** - използва се там, където има подобие между старата и новата система. При този подход двете системи се използват паралелно, докато не се достигне на чувство на сигурност и безпроблемна работа с новата система. Този подход затруднява най-вече потребителите, тъй като означава, че те ще трябва да извършват работните си процеси по два пъти.
3. **Пилотен подход -** въвежда се в експлоатация, т.нар. бета версия на системата, която има за цел да „провери“ как се възприема от потребителите и да се открият грешки преди да бъде пусната официалната версия.
4. **Фазова смяна** - въвеждане на отделни модули на системата поетапно.

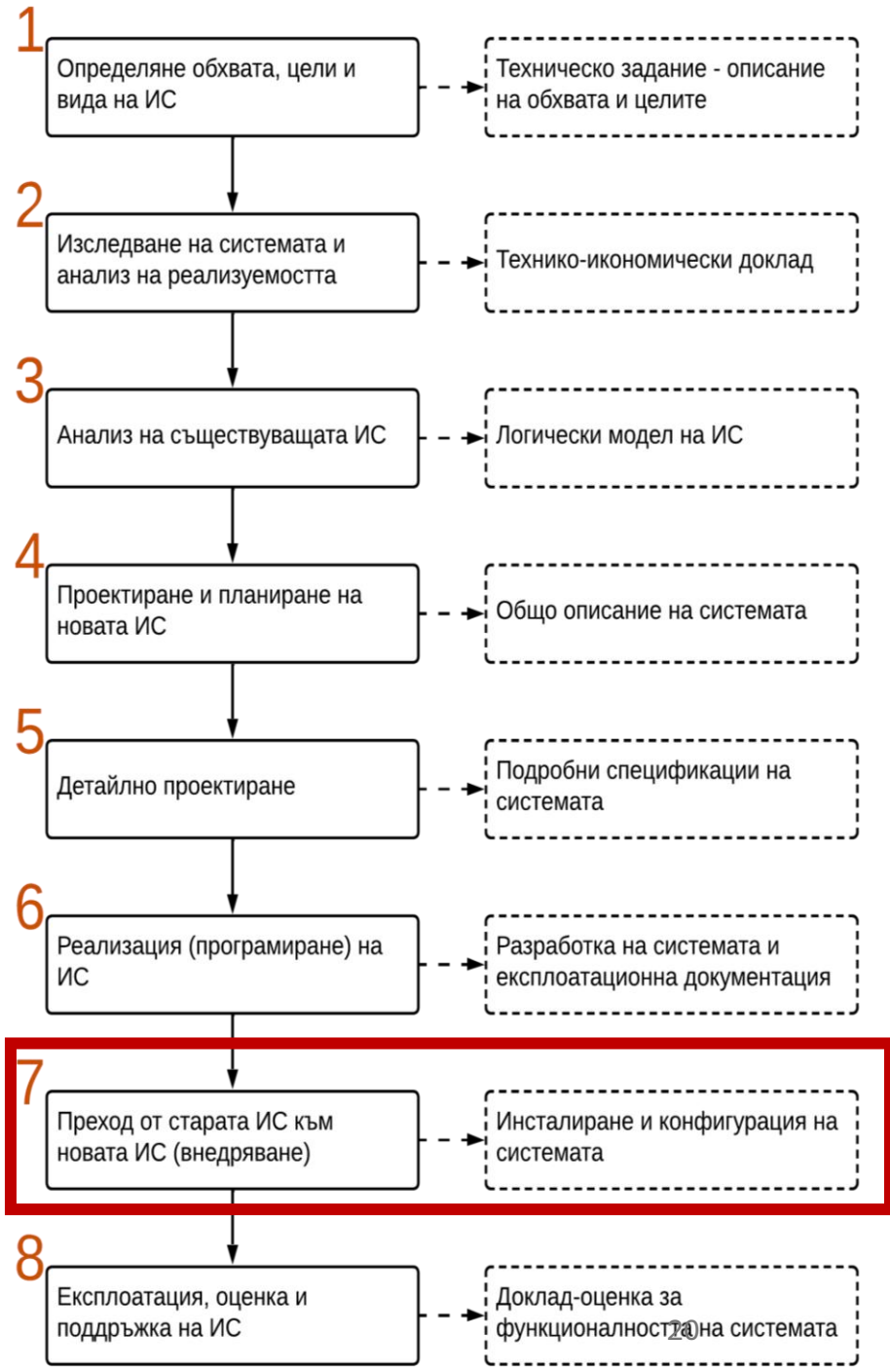
Структурен подход

Жизнен цикъл – етап 7

Преход от старата ИС към новата ИС

Резултат:

- Инсталиране;
- Конфигурация на новата ИС;
- Корекции при необходимост.



Жизнен цикъл – етап 8
Експлоатация, оценка, поддръжка

В етапа на поддръжка влизат освен постоянните обновления на хардуера и софтуера, но и „коригиране“ на програмните модули на системата с цел подобрение, откриване на неизправности, допълване, доразвиване.

След определен период на работа на системата, се създава доклад-оценка за работата на системата. В този доклад се изследват начините на работа на потребителите със системата, прави се сравнение на функционалностите на системата и целите, поставени в първия етап. В резултат на този доклад са възможни промени или повторно проектиране на части от системата.

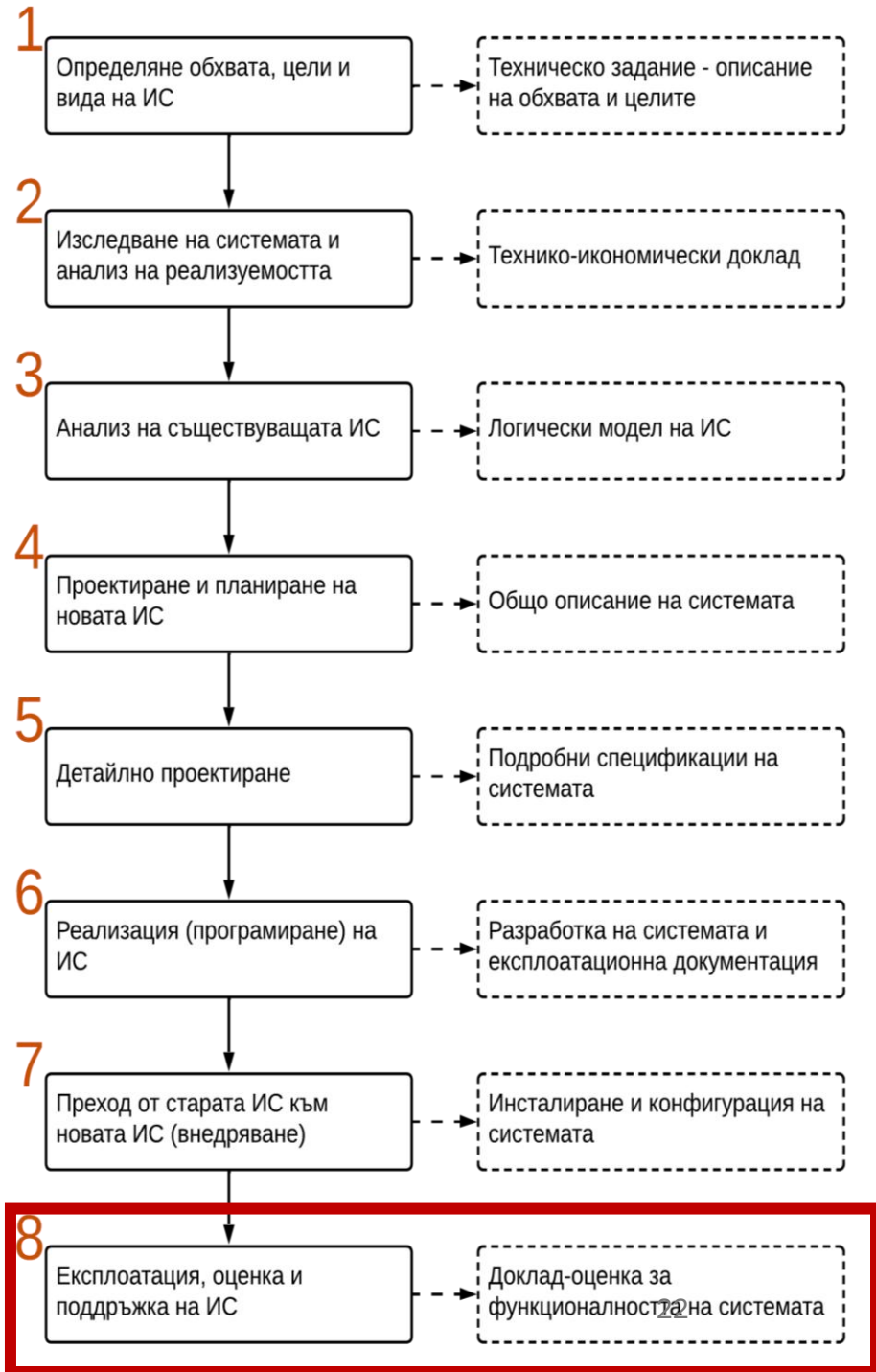
Структурен подход

Жизнен цикъл – етап 8

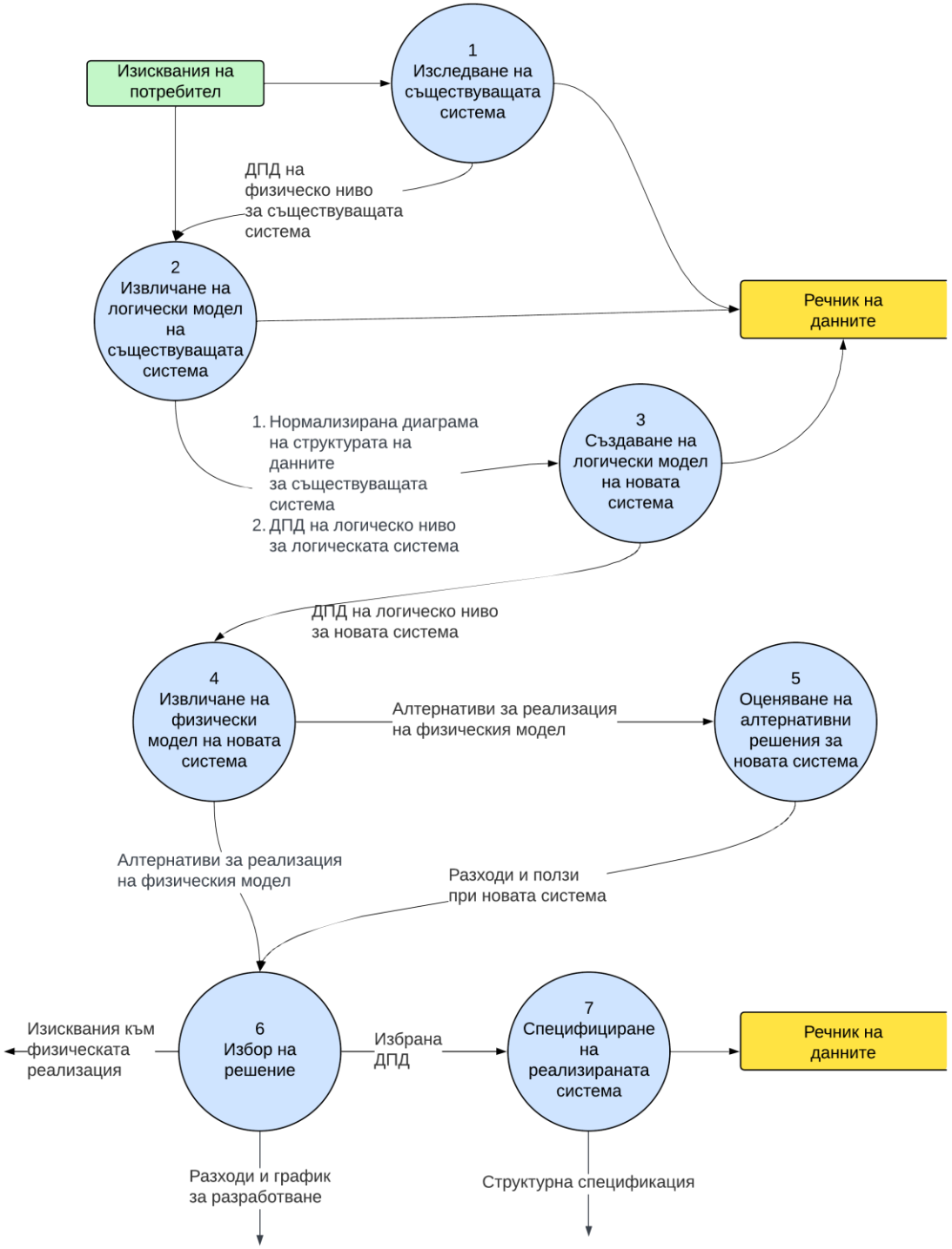
Експлоатация, оценка, поддръжка

Резултат – доклад-оценка за:

- Функционалността на системата;
- Експлоатацията от страна на потребителите;
- Проблеми при работа;
- Потенциални нови функционалности;
- Потенциални подобрения;
- Възможности за допълнителни автоматизации и оптимизации;
- др.



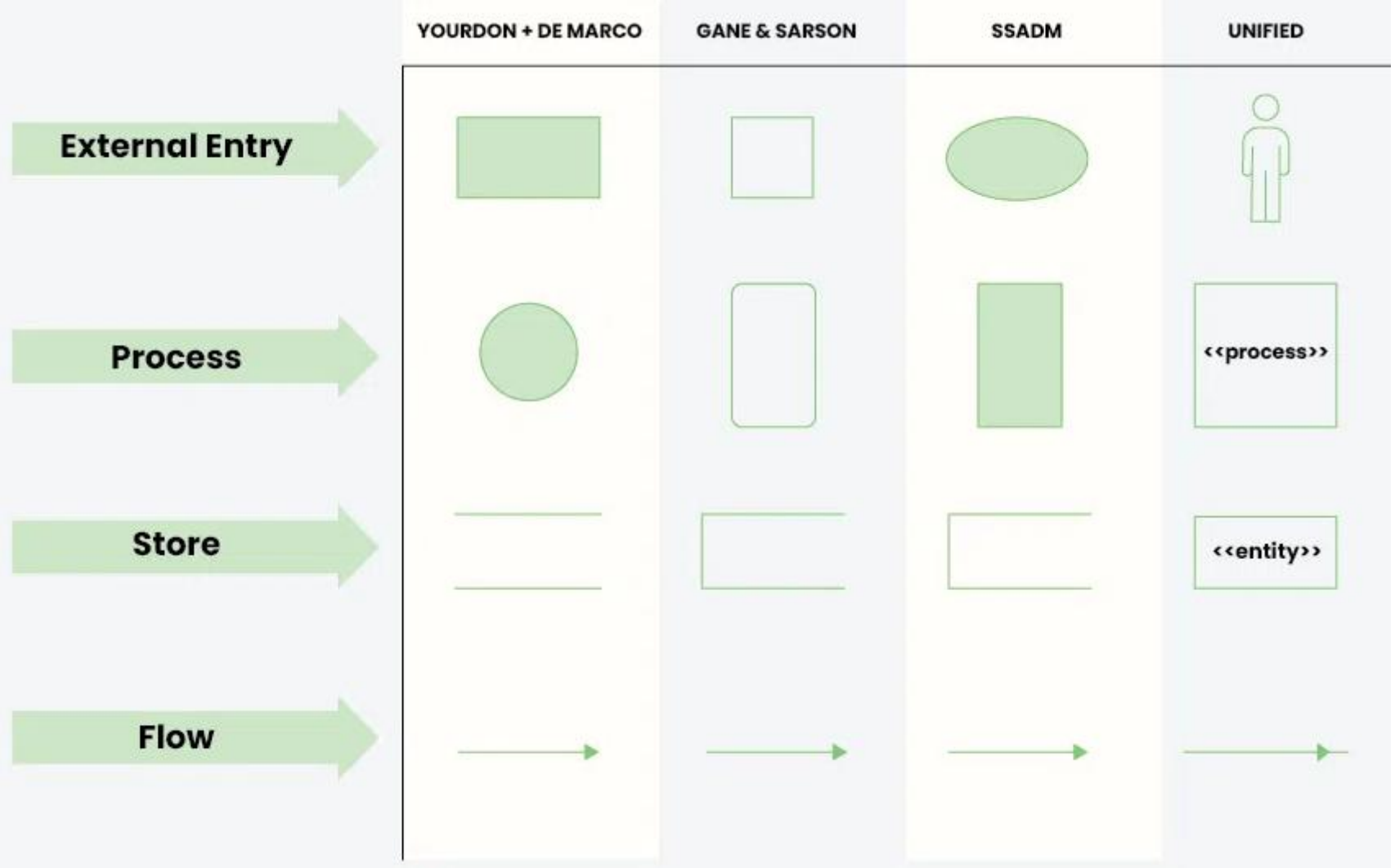
Стъпки при структурния анализ



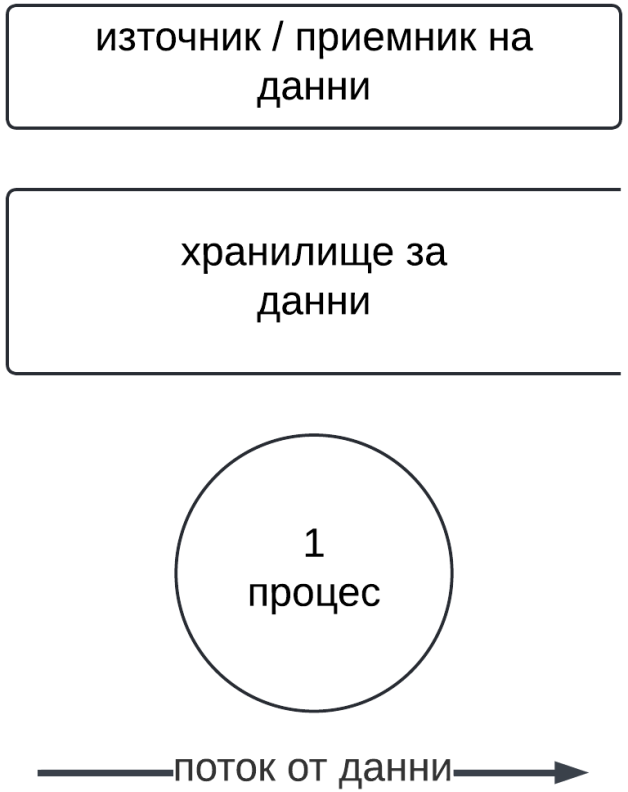
Средства за визаулизация при структурния анализ

- Контекстна диаграма
- Йерархична диаграма
- Диаграми на потоци от данни
- Речници на данни
- Таблици за вземане на решения
- Структуриран английски
- Диаграми на състояния
- E-R диаграми
- Структурни диаграми
- Псевдо-код

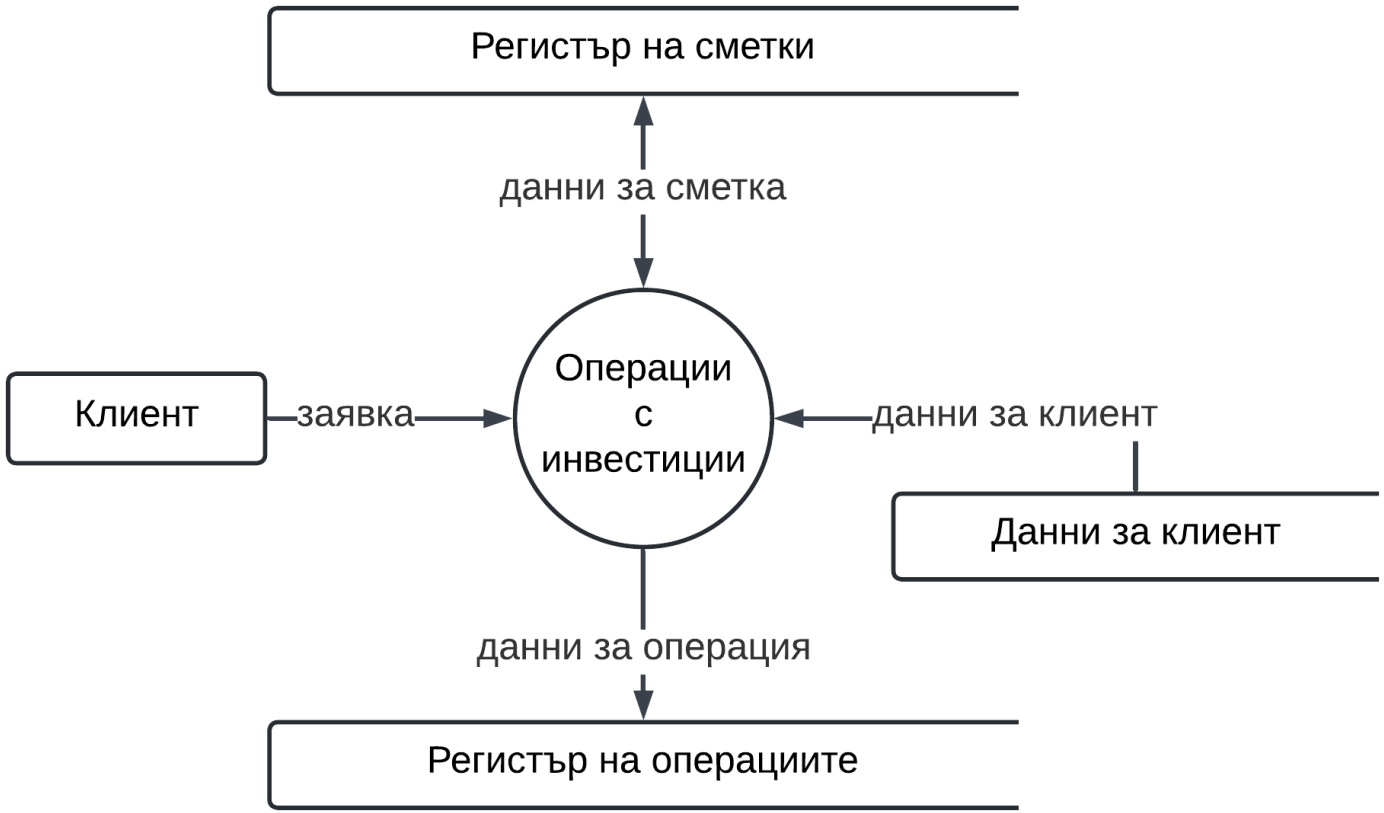
Средства за визуализация – основни елементи *(in the wild)*



Средства за визуализация – основни елементи *(които ще използваме)*

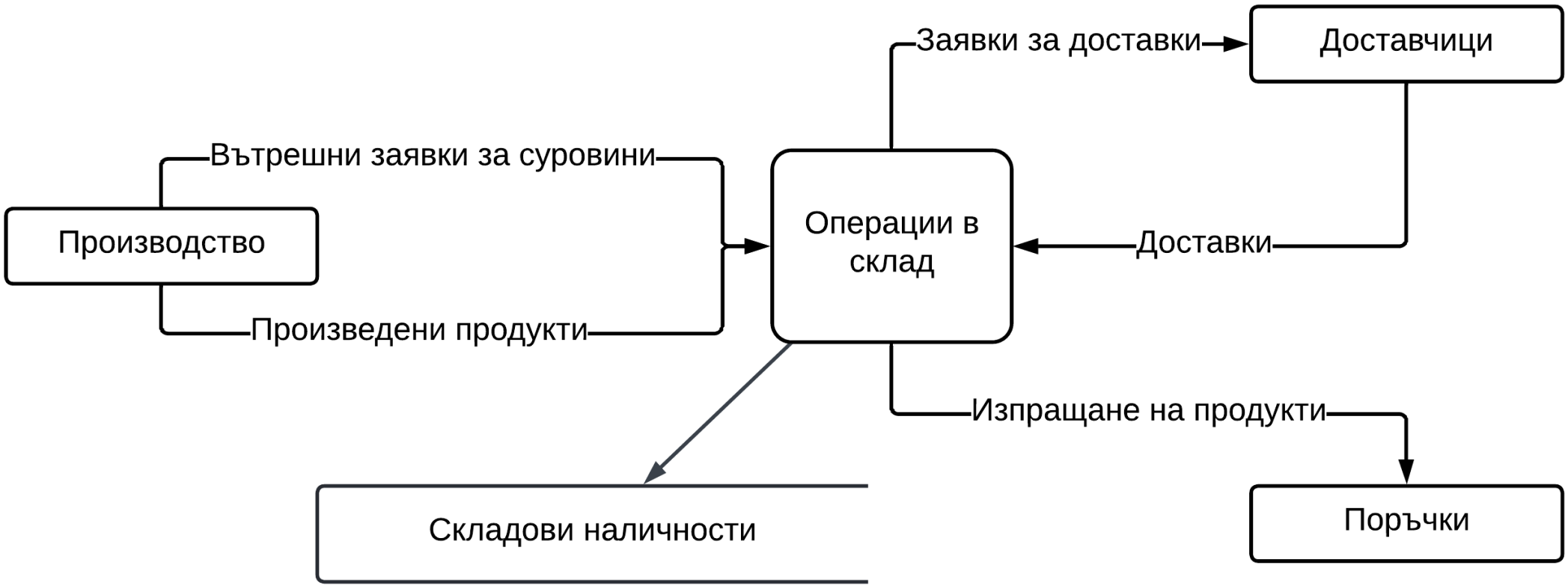


Контекстна диаграма

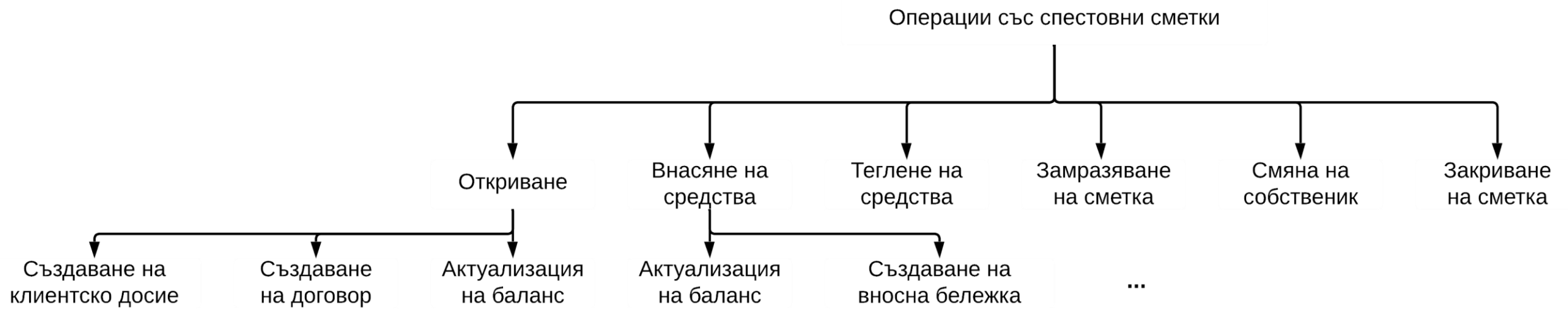


Цялата система се разглежда като единичен процес.
Идентифицират се всички входове, изходи, източници, хранилища и др.
Целта е потребителят да получи представа за функционирането на цялата системата.

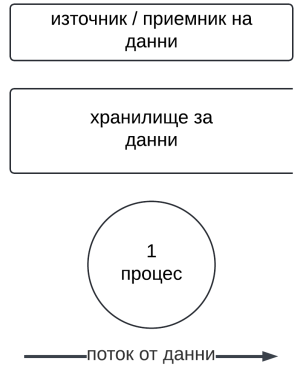
Контекстна диаграма – пример 2



Йерархична диаграма



Диаграми на потоците от данни



Диаграмите на потоците от данни (ДПД; на англ. *Data Flow Diagram, DFD*) са основен механизъм за визуализация на процеси в структурния анализ. С тяхна помощ системата може да бъде разделена на независими модули, така че да бъдат ясно отделени, разбрани и разработени.

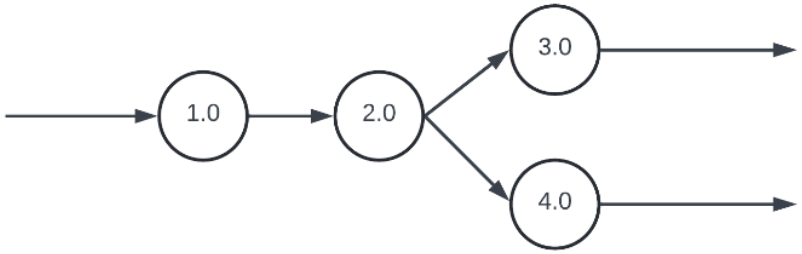
ДПД трябва да отговарят на следните критерии за коректност:

1) Всички компоненти трябва да са наименувани кратко и ясно.
2) Потоците от данни са от/към процеси, т.е. не може да има поток между две хранилища, два източника/получателя, между източник и хранилище и т.н.
3) За всеки процес и хранилище има поне един входен поток, т.е. не може да има процес, който не получава вход или хранилище, в което не се съхранява информация.
4) От всеки процес и хранилище има поне един изходен поток, т.е. не може да има процес, който не произвежда резултат и хранилище, от което не се използва информация.
5) Ако даден процес има много цели, т.е. много входни/изходни потоци, се препоръчва да бъде декомпозиран още на същото ниво на декомпозиция.

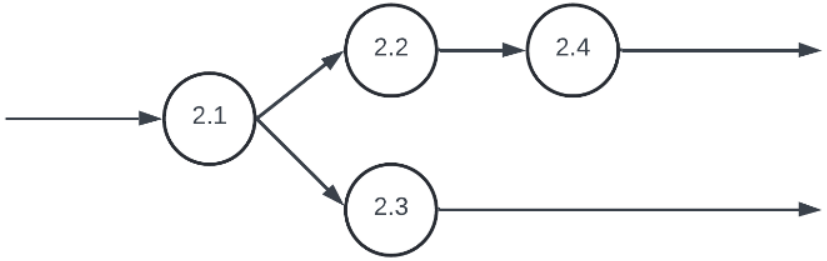
Диаграми на потоците от данни – нива

На **нулевото ниво** са изобразени основните процеси в системата. Всеки от тези процеси може да бъде декомпозиран и разделен на следващото **първо ниво**. Тези процеси могат да бъдат допълнително декомпозирани докато се достигне до най-ниското ниво, съдържащо **примитивни (неделими) процеси**.

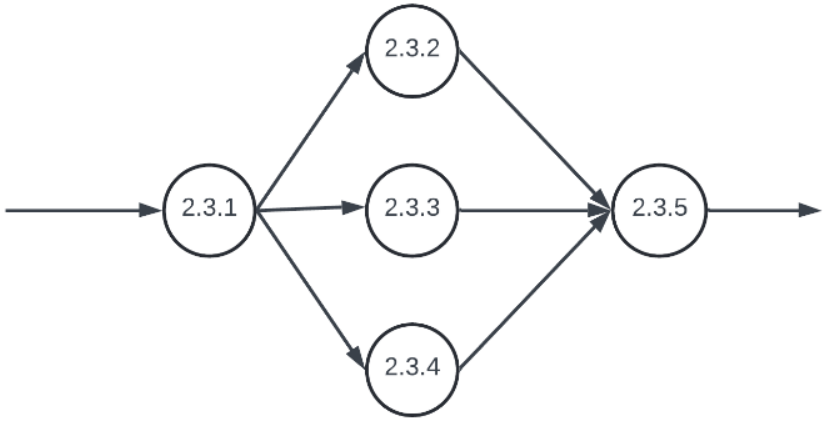
Диаграма на нулево ниво



Диаграма на първо ниво



Диаграма на второ ниво



Подходът на структурният анализ е **отгоре-надолу**.

Системният аналитик първо описва основната цел на организацията, след това основните процеси, а чак след това всеки процес се декомпозира и описва в детайли.

Етапи при построяване на ДПД (съкратени)

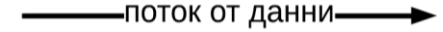
1. Уточняване и изследване на детайлите от текстово описания логически модел;
2. Идентифициране на „*статични компоненти*“ – източници/приемници, хранилища на данни;
3. Начертаване на първия процес (именуване);
4. Начертаване на потоците от данни (именуване);
5. Свързване на процесите с хранилища на данни (именуване);
6. Начертаване и свързване на следващия процес.

Итеративен процес!

Именуване на процеси

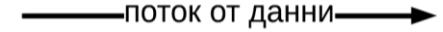
1. Имената трябва да покриват всички дейности, които участват в процеса.
2. Имена, за които не може да се спази условието за всички дейности, показват, че процесът трябва да бъде декомпозиран.
3. В имената трябва да има глагол.
4. Неопределени имена като „процес X“, „изход....“, “избор”, т.н., не трябва да се използват.
5. Ако в името на процеса има „**И**“, то това е признак, че процесът трябва да бъде декомпозиран.

Именуване на потоци от данни - 1



- Стрелката се именува, следвайки някои добри практики:
  - Възможно е името да не е цяло, тъй като потокът от данни или не съществува, или се състои от много елементи данни. В тези ситуации, потокът трябва да бъде разделен на няколко потока. Имена като „*поток-1*“, „*входен поток*“,, „*общи данни*“ ще покажат, че е необходим допълнителен анализ.
  - Името на потока трябва да съответства на целия поток на данни, а не само на част от него.
  - „Общи“ думи като „*форма*“, „*списък*“, „*данни*“, „*информация*“ трябва да се избягват, тъй като не дават никаква конкретика за потока от данни.
- Стрелката може да пренася данни от източник/приемник или хранилище на данни към определен процес, или от процес към друг процес, източник/приемник на данни или хранилище.
- Данни не могат да се предават директно:
  - между източници/приемници,
  - между хранилища,
  - от източник към хранилище
  - хранилище към приемник.

Именуване на потоци от данни - 2



- Потоци към/от хранилища **не са** именувани, тъй като се предполага, че съдържанието на потока от данни съвпада с името на и съдържанието на хранилището на данни.
- Потоци към процесите **трябва да са** именувани.
- Потоци от/към източници/приемници на данни **са** именувани.
- Стрелка, сочеща към хранилище, показва, че данните се *актуализират*. Стрелка от хранилището за данни, показва, че данните се *използват, но не се променят*. Ако данните едновременно се използват и променят, то тогава стрелката е двупосочна.

ДПД - пример

Процеса по покупка на продукти от онлайн магазин.

Disclaimer: за целите на примера, диаграмата е опростена и не представя всички процеси по покупка от онлайн магазин!

В процеса присъстват клиенти, които разглеждат продуктите в магазина, добавят ги в количка, попълват данни за доставка, която може да им бъде изчислена на момента, потвърждават количката и данните, и получават фактура.

ДПД - пример

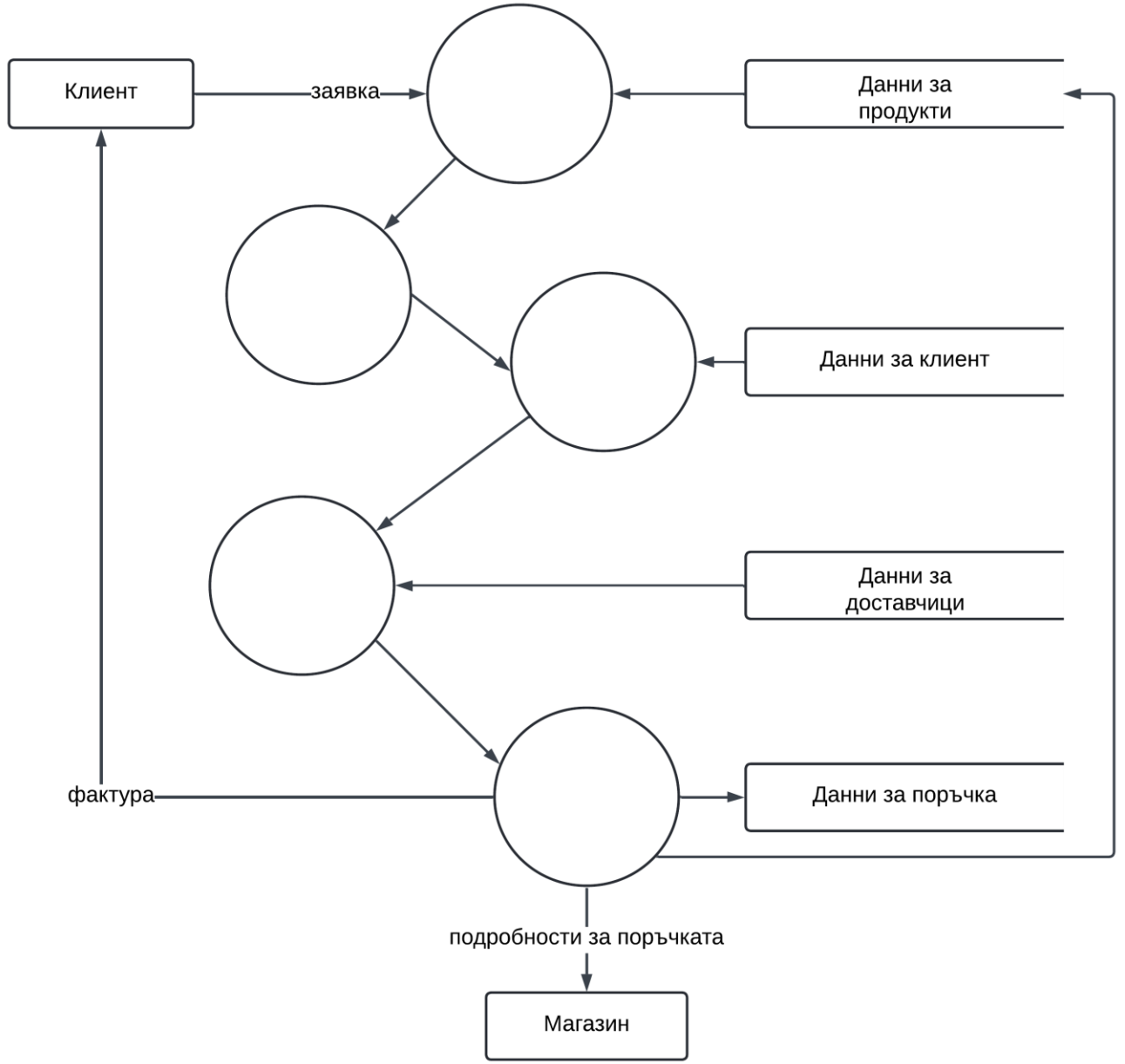
1. Анализиране на текстовото описание на логическия модел

2. Определяне на статичните компоненти

ДПД - пример

Стъпки 3 → 6

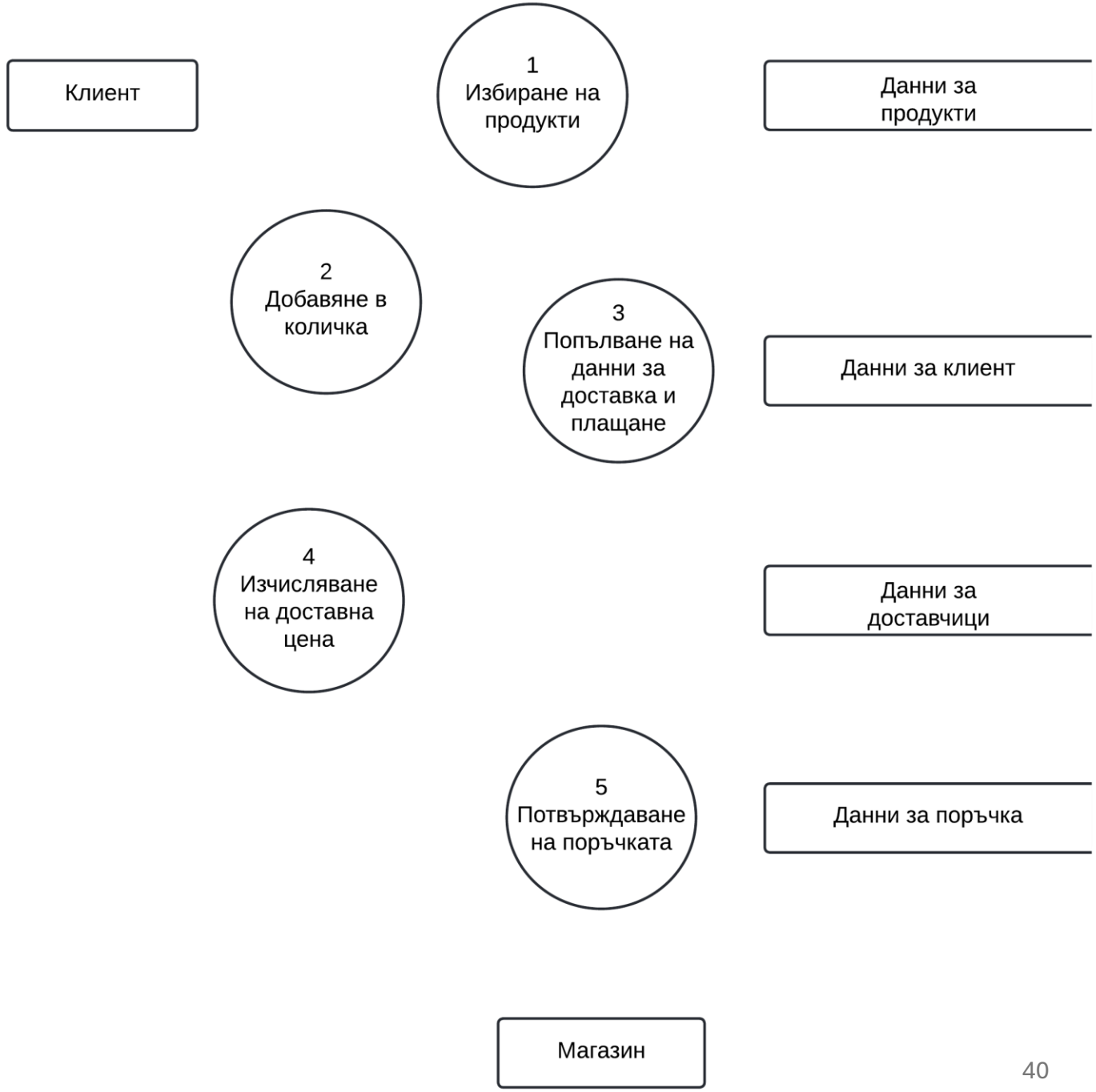
1 вариант →



ДПД - пример

Стъпки 3 → 6

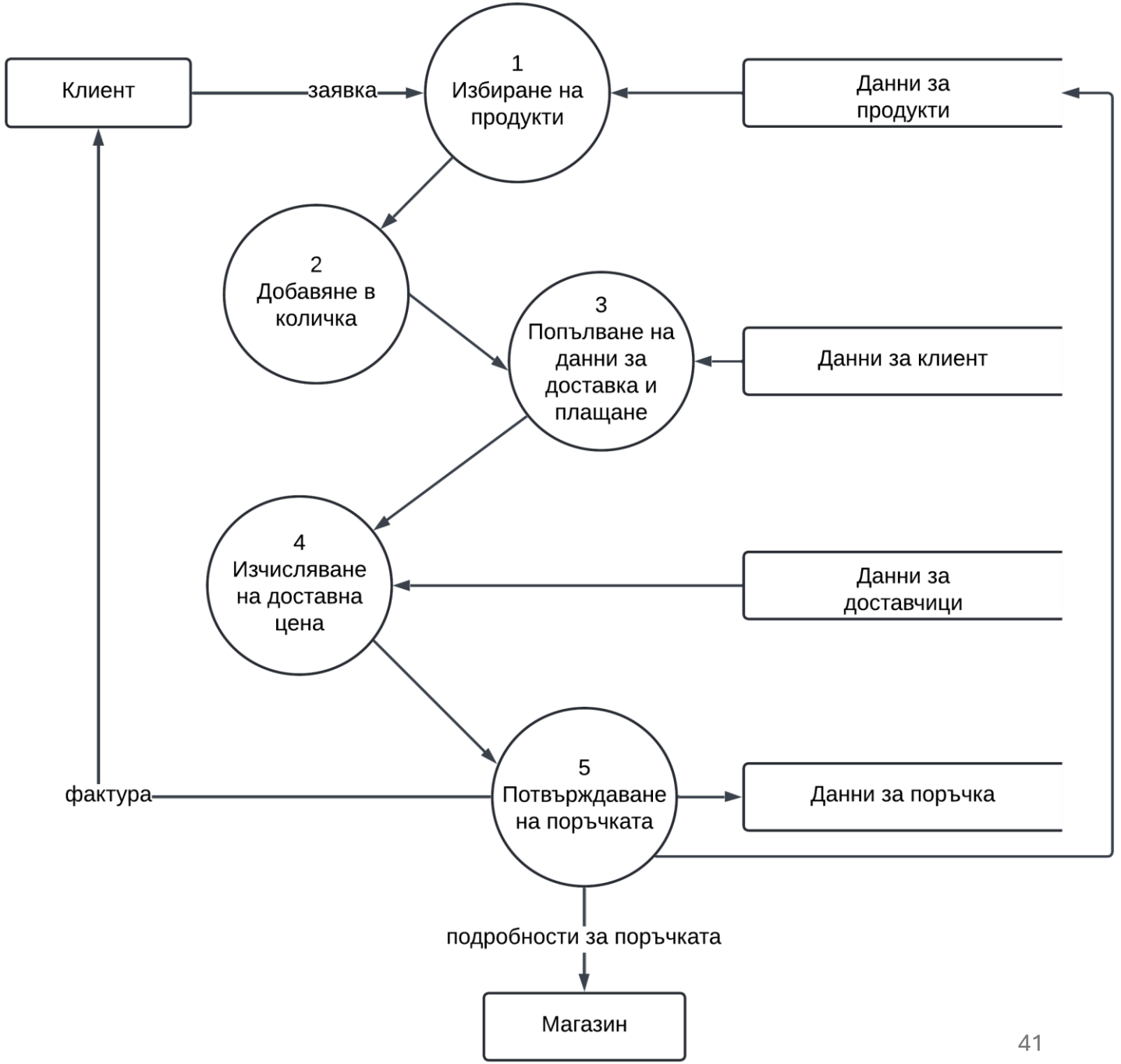
2 вариант →



ДПД - пример

Почти краен вариант:

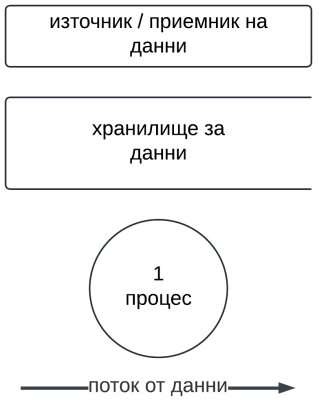
- Има източници/приемници на данни
- Хранилища на данни
- Потоци на данни
- Процеси ~~– декомпозирани?~~
- Няма елементи, които не са свързани
- Всички елементи са именувани, ~~според правилата и добрите практики~~



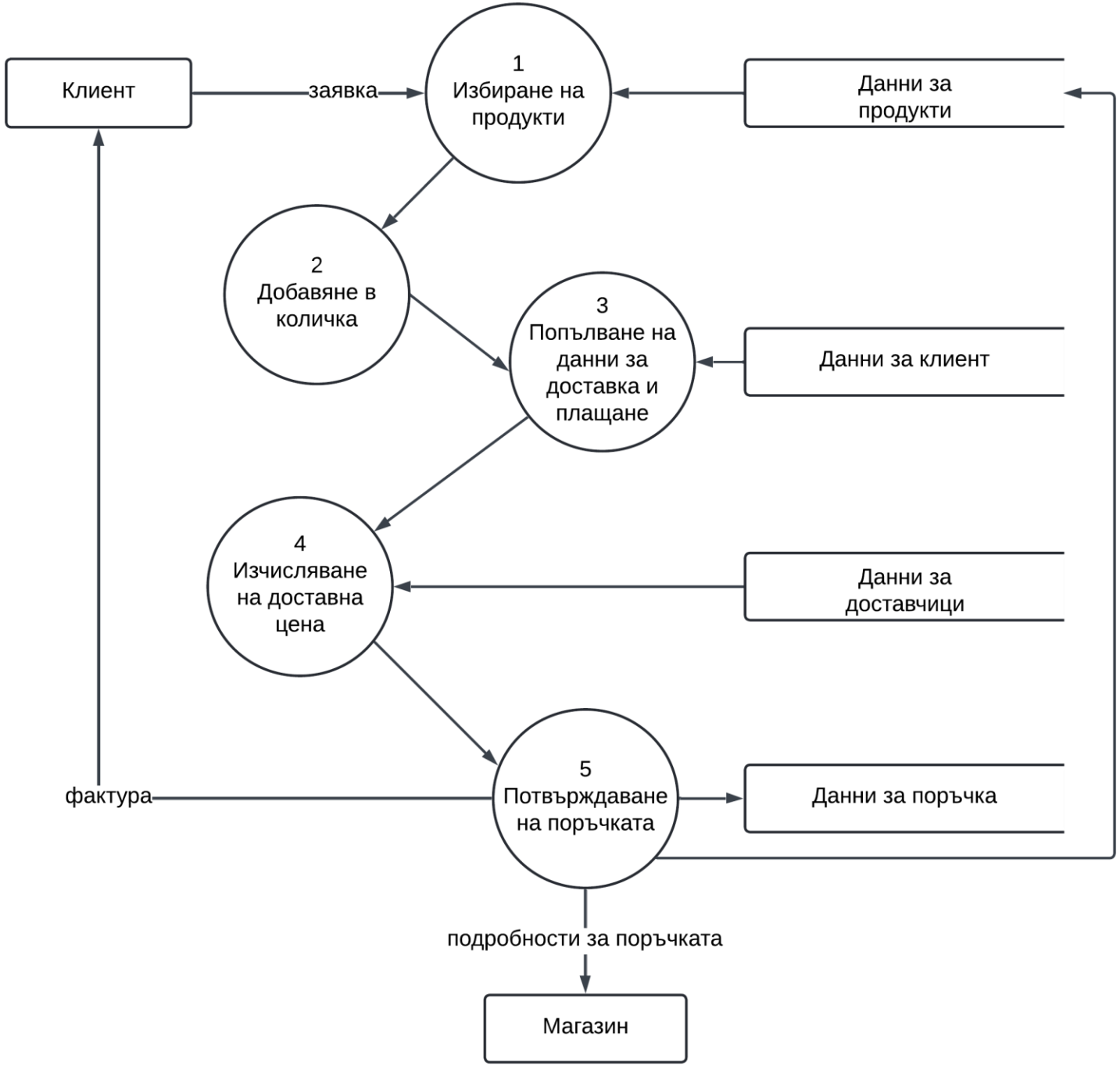
ДПД – Разширяване на диаграмата

На този етап на валидация и разширение на диаграмата се преглеждат няколко неща:

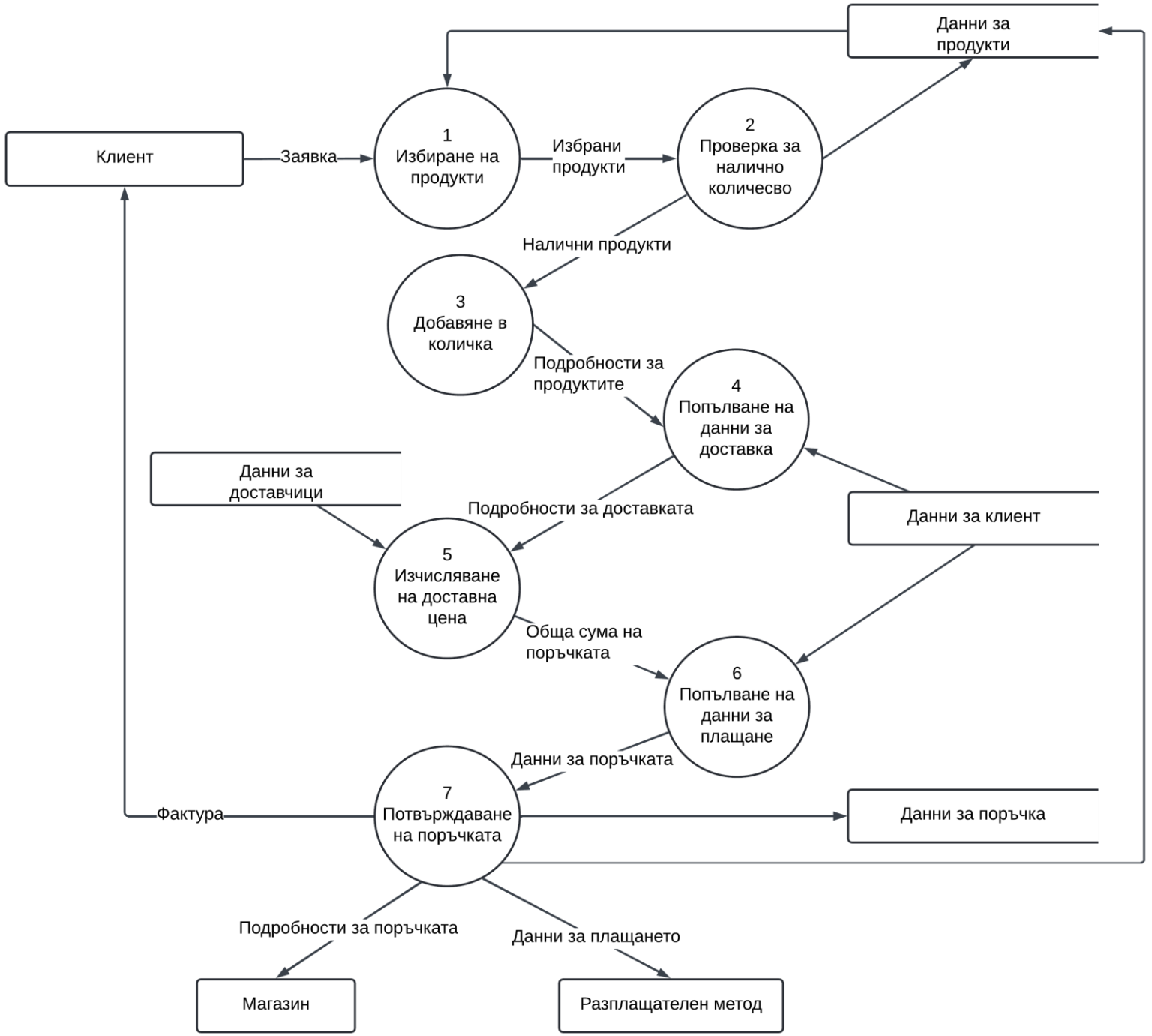
- Дали всички процеси са добавени;
- Дали всички потоци от данни са добавени;
- Дали всички източници и приемници на данни са добавени;
- Дали всички елементи са именувани коректно и ясно;
- Дали има процеси, които могат да бъдат допълнително разбити.

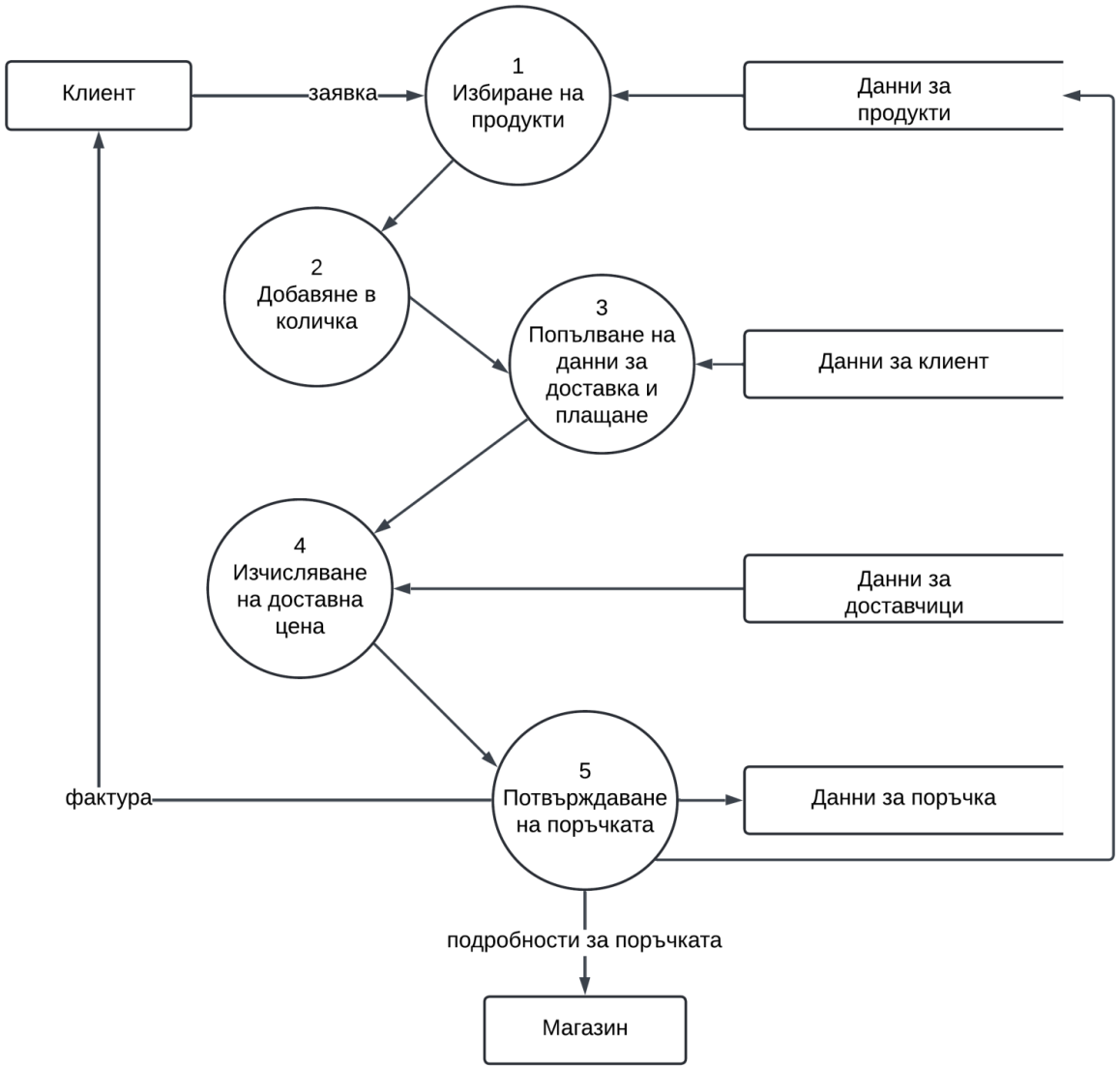


преди

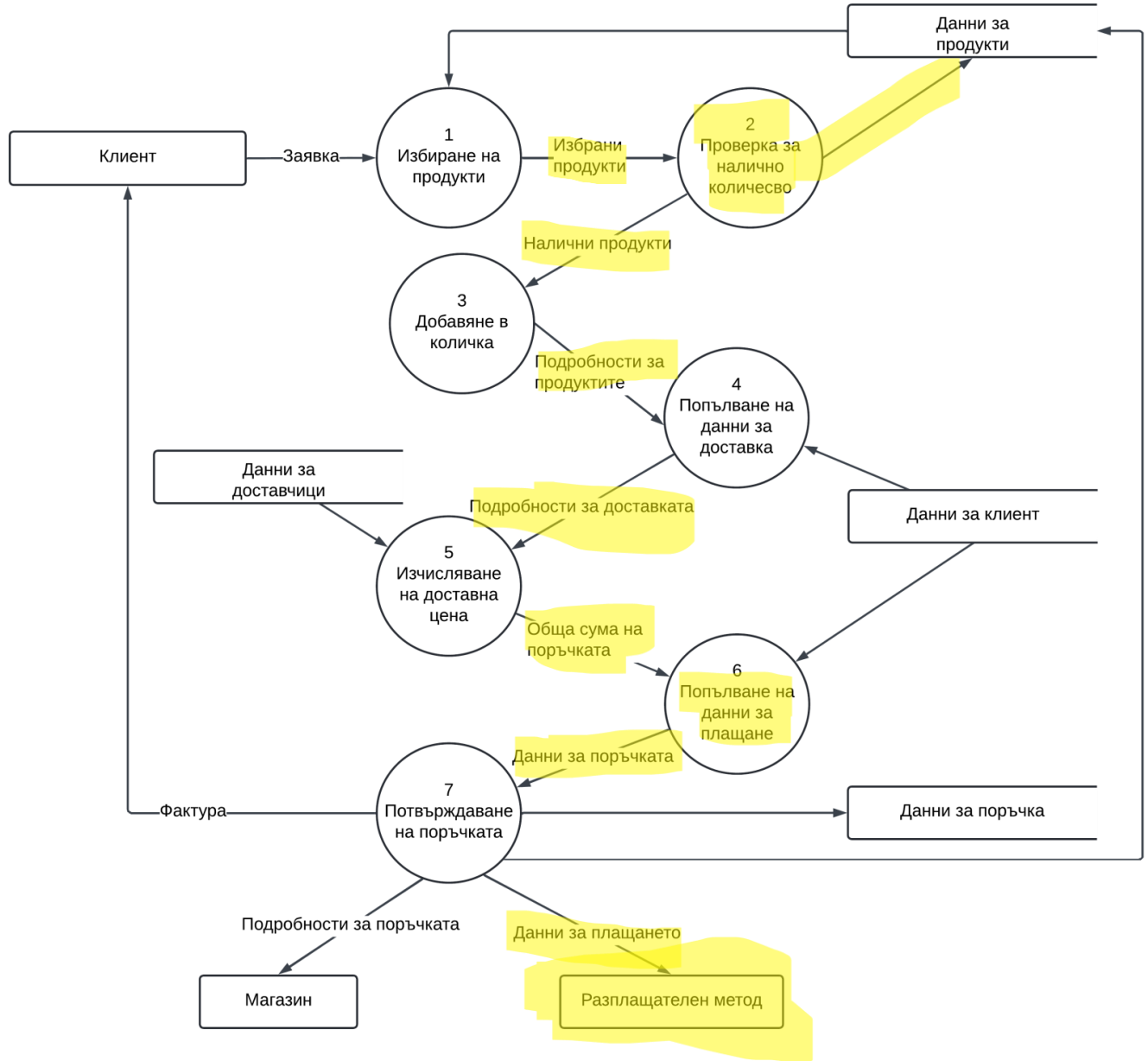

Клиент
заявка
1
Избиране на продукти
Данни за продукти
2
Добавяне в количка
3
Попълване на данни за доставка и плащане
Данни за клиент
4
Изчисляване на доставна цена
Данни за доставчици
5
Потвърждаване на поръчката
Данни за поръчка
фактура
подробности за поръчката
Магазин

след


Данни за продукти
Клиент
Заявка
1
Избиране на продукти
Избрани продукти
2
Проверка за налично количество
Налични продукти
3
Добавяне в количка
Подробности за продуктите
4
Попълване на данни за доставка
Данни за доставчици
Подробности за доставката
Данни за клиент
5
Изчисляване на доставна цена
Обща сума на поръчката
6
Попълване на данни за плащане
Данни за поръчката
7
Потвърждаване на поръчката
Фактура
Данни за поръчка
Подробности за поръчката
Данни за плащането
Магазин
Разплащателен метод

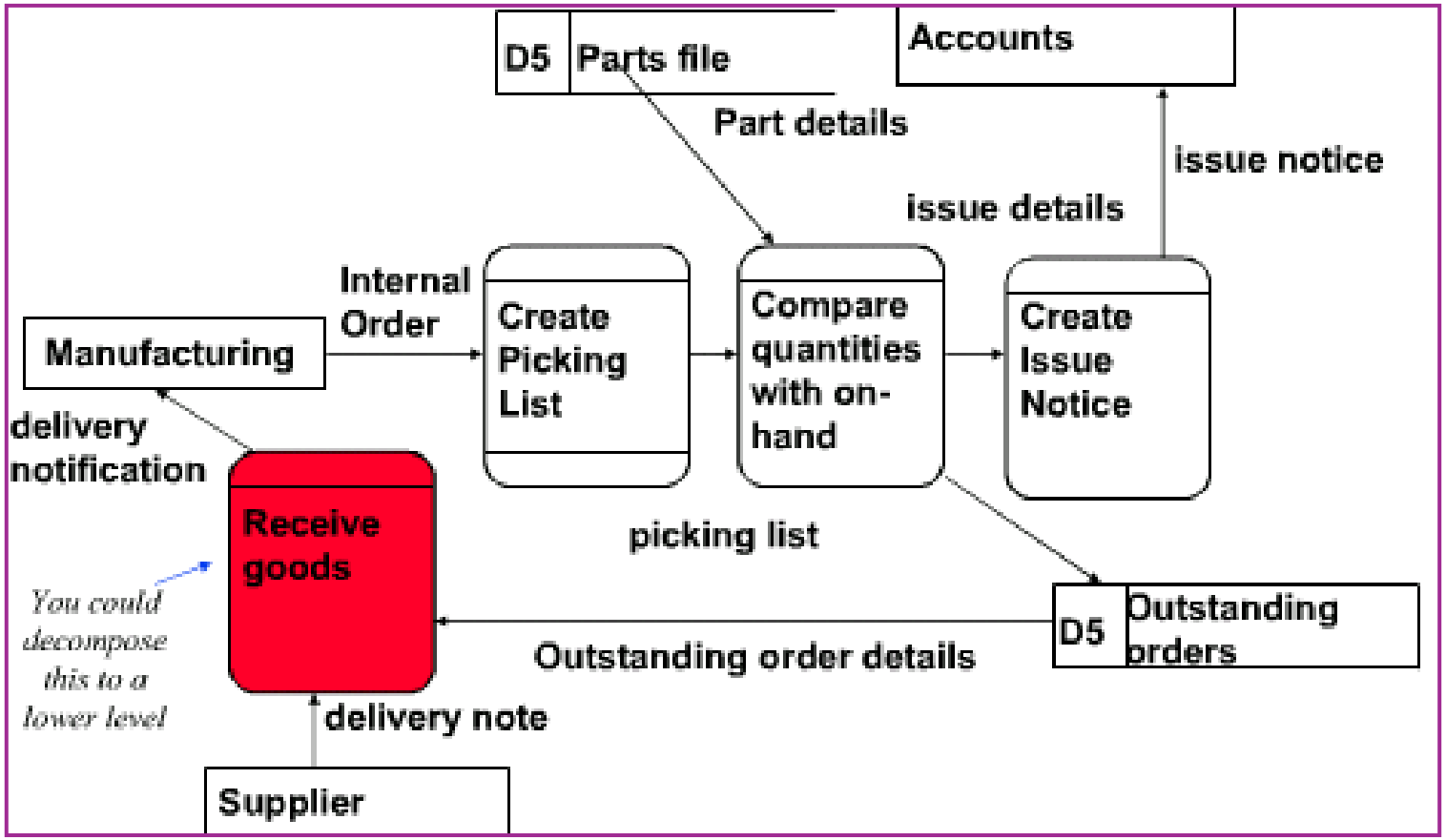

Клиент
заявка
1
Избиране на продукти
Данни за продукти
2
Добавяне в количка
3
Попълване на данни за доставка и плащане
Данни за клиент
4
Изчисляване на доставна цена
Данни за доставчици
фактура
5
Потвърждаване на поръчката
Данни за поръчка
подробности за поръчката
Магазин

преди


Данни за продукти
Клиент
Заявка
1
Избиране на продукти
Избрани продукти
2
Проверка за налично количесво
Налични продукти
3
Добавяне в количка
Подробности за продуктите
4
Попълване на данни за доставка
Данни за доставчици
Подробности за доставката
Данни за клиент
5
Изчисляване на доставна цена
Обща сума на поръчката
6
Попълване на данни за плащане
Данни за поръчката
Фактура
7
Потвърждаване на поръчката
Данни за поръчка
Подробности за поръчката
Данни за плащането
Магазин
Разплащателен метод

след

ДПД – Декомпозиране на процеси – пример



ДПД – Изолиране на процеси

Важни характеристики на процесите, на които трябва да се обърне внимание:

- Физическите процеси не са част от информационната система.
- Правилно дефиниран процес има краен резултат или съвкупност от резултати.
- Правилно дефиниран процес има точно определено начало и край.
- Участниците в процесите.
- Автоматизиране на процеси или подпроцеси.

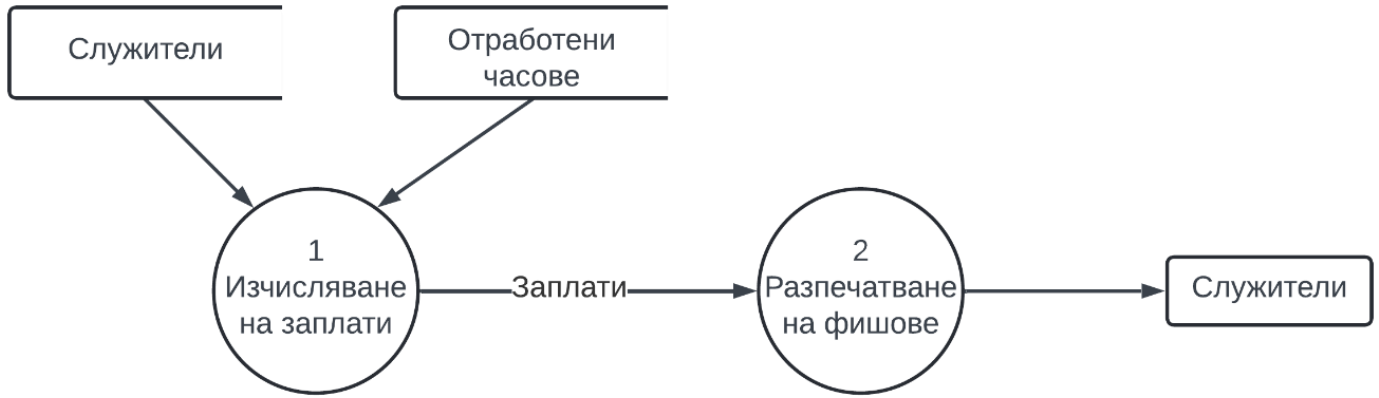
ДПД – Изолиране на процеси

Изолираните процеси могат да бъдат проверени в края на всеки поток от данни, като се зададат следните три въпроса:

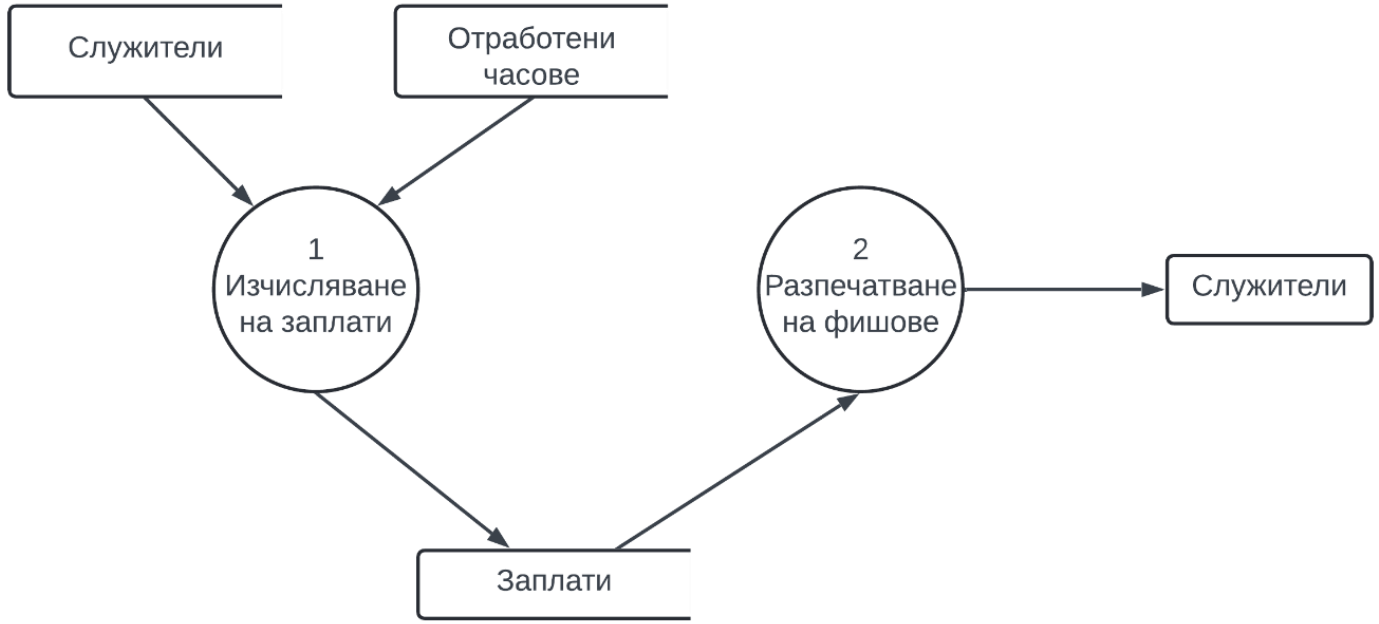
1) Следващият процес може ли да бъде изпълнен **в друго време**?
2) Следващият процес може ли да бъде изпълнен **на друго място**?
3) Следващият процес може ли да бъде изпълнен **от друг човек**?

Изолирането става като данните се записват в хранилище след като предходният процес е приключил и се четат от хранилището, когато следващият процес започне.

ДПД – Изолиране на процеси



а) неизолиран процес



б) изолиран процес

ДПД – коригиране на диаграмата

Речници на данни

В речниците на данни (на англ. *Data dictionaries*) се съхранява информация за самите данни – тяхната структура и използване. Целта му е да дава точна информация за използваните данни, за да може да бъде проектирането възможно най-коректно.

В речниците на данни се съхранява допълнителна информация, която няма място в ДПД, но е важна част за изпълнението на целия процес, описан с ДПД.

Важно е описанието на данните в речника да не бъде дублирано, за да могат данните да бъдат актуализирани само на едно място.

Речници на данни - примери

Означения:

= означава „еквивалентно на“

+ означава „и“

[] означава „или - или“

{} означава *итерация*

() означава, че изброените компоненти не са задължителни

1. **Актуализиране_на_документ=**
   номер_на_документ+
   идентификатор_на_потребител+
   дата_на_актуализация+
   информация_за_актуализацията
2. **Отчет_за_история_на_актуализации=**
   номер_на_документ+{актуализации_на_документа}
3. **Идентифициране_на_служител=**
   [идентификатор_на_служител|име_на_служител]
4. **Идентифициране_на_служител=**
   идентификатор_на_служител+име_на_служител
   +(e-mail_на_служител)

Таблици за вземане на решения

С таблиците за вземане на решения се представят сложни логически връзки. Таблиците са особено полезни, когато резултатните действа зависят от наличието на едно или няколко независими условия. В същината си, таблиците са матрици от редове и колони, дефиниращи проблем и възможни действия,

- **Пълен текст на условие** – намира се в горния ляв квадрант, който изброява всички условия, които трябва да бъдат проверени.
- **Действие** – в долния ляв квадрант и показва всички действия, които трябва да бъдат извършени, за да се изпълни това условие.
- **Въвеждане на условие** – намира се в горния десен квадрант и дава отговори на въпроси, зададени в квадранта на условието.
- **Въвеждане на действие** – в долния десен квадрант и показва подходящото действие, произтичащо от отговорите на условията в квадранта за въвеждане на условие.

Таблици за вземане на решения

1. **Правило 1**: Ако продуктът е наличен (Y), плащането е потвърдено (Y), и е избрана експресна доставка (Y), тогава потвърди поръчката и изпрати продукта (X).

2. **Правило 2**: Ако продуктът е наличен (Y) и е избрана експресна доставка (Y), но плащането не е потвърдено (N), тогава изпрати уведомление за очакване на плащане (X).

3. **Правило 3**: Ако продуктът не е наличен (N), но плащането е потвърдено (Y) и е избрана експресна доставка (Y), предложи алтернативен продукт (X).

4. **Правило 4**: Ако продуктът е наличен (Y) и плащането е потвърдено (Y), но не е избрана експресна доставка (N), предложи стандартна доставка (X).

5. **Правило 5**: Ако продуктът не е наличен (N), плащането не е потвърдено (N), и не е избрана експресна доставка (N), тогава откажи поръчката (X).

Таблици за вземане на решения

Условия/Действия	Правило 1	Правило 2	Правило 3	Правило 4	Правило 5
Условия					
Продуктът е наличен на склад	Y	Y	N	Y	N
Плащането е потвърдено	Y	N	Y	Y	N
Избрана е експресна доставка	Y	Y	Y	N	N
Действия					
Потвърди поръчката и изпрати	X				
Изпрати уведомление за плащане		X			
Предложи алтернативен продукт			X		X
Предложи стандартна доставка				X	
Откажи поръчката					X

Структуриран английски

Структурираният английски произлиза от структурирания език за програмиране, който дава по-разбираемо и точно описание на процеса.

Използва се най-добре, когато трябва да се вземат предвид последователности и цикли в програма, и проблемът се нуждае от последователности от действия с решения. Няма строго правило за синтаксиса. Той изразява цялата логика по отношение на последователни структури за вземане на решения и итерации. Прилича на псевдо-код, но не е!

Псевдо-код

1) Яснота и простота – псевдо-кодът трябва да бъде написан на обикновен език (най-често на английски) с ясни инструкции. Колкото по-прост, толкова по-лесен за разбиране;
2) Структура и последователност – използването на отстъпи е задължително, за да се покаже йерархията от цикли и условия. Оператори (условни оператори, цикли и т.н.) трябва да бъдат използвани по един и същ начин в кода;
3) Контрол на потока:
    a. Условни изрази – структурата if-then-else трябва да се използва за представяне на логически разклонения;
    b. Цикли – for, while, do-while, do-until;
    c. Край на блокове – всяка структура трябва да бъде затворена с някои от операторите – end for, end if;
4) Деклариране на променливи – имената на променливите трябва да е ясно, спрямо съдържанието на променливата. Ако е важно да се означи типа на променливата, това трябва да бъде направено. Ако типът не е от съществено значение, то може да бъде пропуснат;
5) Функции и методи – дефинирането на функция става със служебната дума function последвана от списък с входни аргументи, ако има такива;
6) Коментари – добавянето на коментари е добра практика за по-лесното разчитане и разбиране;
7) Специфики на езика – в псевдо-кода не трябва да има елементи и конструкции, специфични за даден програмен език.

Структуриран английски:

```
if customer pays advance
  then
    Give 5% Discount
  else
    if purchase amount >=10,000
      then
        if  the customer is a regular
          then Give 5% Discount
        else  No Discount
      end if
    else No Discount
  end if
end if
```

Псевдо-код:

```
FUNCTION FindMaxNumber(numbers)
    max = numbers[0]

    FOR each number IN numbers
        IF number > max THEN
            max = number
        END IF
    END FOR

    RETURN max
END FUNCTION
```

Структурно проектиране