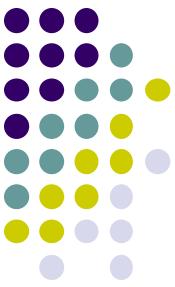


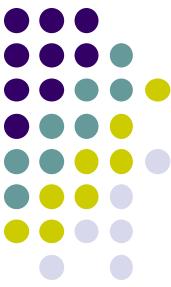


Анализ на софтуерните изисквания



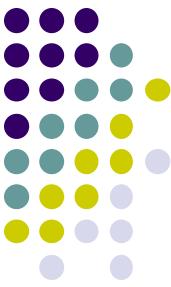
Преподавателски екип

- проф. Олга Георгиева, ФМИ, катедра СТ
o.georgieva@fmi.uni-sofia.bg
- ас. Иrena Pavlova, ФМИ, катедра СТ
irena_pavlova@fmi.uni-sofia.bg
- Светимир Игнатов
svetimir.ignatov@gmail.com



Цели на курса

- Да представи съвременното състояние на научните изследвания и практиката в областта на дисциплината **Анализ на софтуерните изисквания (Инженеринг на изискванията, Requirements Engineering)**
- Да разгледа аспектите *на процеса* на обработка на софтуерните изисквания:
 - извлечане и анализ на изискванията
 - специфициране на изискванията
 - валидиране на изискванията и
 - управление на изискванията
- Да представи *методи и техники*, прилагани при анализа на софтуерните изисквания:
 - структурни методи
 - методи в зависимост от гледната точка
 - техники за специфициране на функционални и нефункционални изисквания



Повече за курса

- Курсът обхваща материала, свързана с инженеринг на изискванията:
 - Анализ на проблема и ситуацията за бъдещата система
 - Формулиране на изискванията
 - Документиране на изискванията
 - Как и какъв процес да се изпълнява, с какви участници
 - Използване на подходящи софтуерни инструменти
- Курсът е различен от останалите курсове от КН:
 - Той **не засяга** въпросите как да се решат проблемите с използване на компютрите
 - Той решава въпросите на **идентифицирането на проблемите**

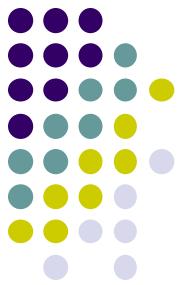


Умения

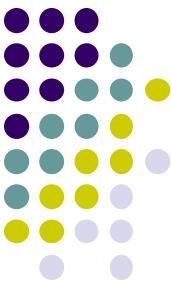
След завършване на курса студентите ще могат да:

- *откриват и извличат изискванията*, използвайки различни **техники**;
- прилагат различни *техники за изследване на изискванията* като: анализ на нуждите на потребителите, на целите на създавания софтуер, анализ на случаите на употреба;
- *организират и приоритизират* изискванията;
- *валидират изискванията* според различни критерии като приемливост, яснота, липса на двусмисленост и др.
- *представят функционални и нефункционални изисквания* за различни типове системи, използвайки *формални и неформални техники* на описание
- *водят преговори* за постигане на съгласие относно изискванията към разработвана софтуерната система.

Изисквания към слушателите на курса

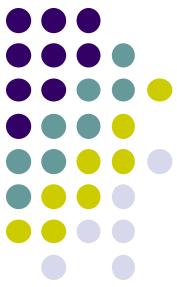


- Присъствие на лекциите и упражненията – сильно препоръчително
- Запознаване с посочена литература
- Работа в екип за курсовите задачи
- Предаване на домашните работи в указания срок



Оценяване

- 45% от писмен изпит (**задължителен компонент**, минимум среден 3.00)
- 55% от изработка и защита на:
 - Контролни работи 2бр. x 10%
 - Домашни работи 5бр. (общо 35%) , оформени в цялостен проект (**задължителен компонент**)



Страница на курса в MOODLE

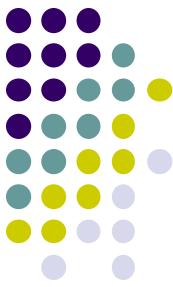
<https://learn.fmi.uni-sofia.bg>

парола: **RE2020**



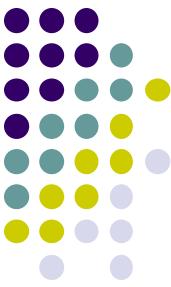
Структура на курса

- Тема 1: Инженеринг на изискванията като част от разработването на софтуерни системи
 - Мотивация
 - Дефиниране на основни понятия
 - Участници и роли при инженеринга на изискванията.
- Тема 2: Въведение в основните дейности при инженеринга на изискванията
 - идентифициране на изискванията,
 - анализ и специфициране,
 - валидиране и управление.
- Тема 3: Таксономия на изискванията - функционални, нефункционални, потребителски, бизнес и системни изисквания. Основни характеристики на изискванията според IEEE критериите. Нефункционални изисквания. Класификация на нефункционалните изисквания. Изисквания към критични системи.



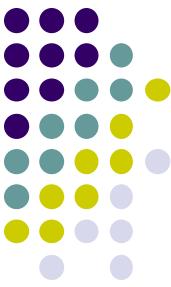
Структура на курса (2)

- **Тема 4: Граница на системата**, контекст на системата. Цели на инженеринга на изискванията. Източници за идентифициране на изискванията. Потребителски случаи – основни принципи и понятия. **Техники за извлечане на изискванията**. Проблеми при идентифициране на изискванията. SMARTT изисквания.
- **Тема 5: Прототипиране. Анализ на изискванията**. Процес на анализ на изискванията. Техники за анализ на изискванията. Договаряне на изискванията.
- **Тема 6: Документиране на изискванията** на естествен език. Специфициране на изискванията чрез аналитични модели: текстови и графични означения и езици. Формални методи за описание.
- **Тема 7: Валидиране на изискванията**. Принципи и техники за валидиране. Рецензиране и прототипиране. Валидиране на моделите. Тестване на изискванията.



Структура на курса (3)

- **Тема 8:** Подход на различните гледни точки при инженеринга на изискванията. Техника за структуриран анализ и дизайн. Метод CORE.
- **Тема 9:** Управление на изискванията. Приоретизиране на изискванията. – MoSCoW, RFC 2119. Характеристики на „добрите” изисквания. Дефиниране на постоянни и променливи изисквания. Управление на промените. Проследяване на изискванията в целия процес на разработване.
- **Тема 10:** Организация и стандарти при определяне на изискванията. ПВА ВOK, ITIL, BCS. Инженеринг на изискванията в модела SEI СММ/СММІ. Анализиране на бизнес изисквания – опит от фирми в България - гост лекция.



Структура на курса (4)

- Теми за упражнения
 - 1. Представяне и запознаване със съвременни софтуерни инструменти, подпомагащи процеса на инженеринг на изискванията
 - 2. Техники за извлечане на изискванията
 - 3. Документиране на изискванията на софтуерна система
 - 4. Специфициране и анализиране на изискванията чрез прилагане на аналитични модели
 - 5. Валидиране и тестване на изискванията
 - 6. Представяне и защита на курсови работи



Литература - 1

Основна

- Ian Sommerville, Software Engineering, Adisson Wesley, 8 edition, 2007, ISBN 0 321 31379 8
- Gerald Kotonya and Ian Sommerville, Requirements Engineering: Processes and techniques, John Wiley&Sons, 2003, ISBN 0 471 97208 8

Допълнителна

- Klaus Pohl, Requirements Engineering, Springer-Verlag Berlin Heidelberg, 2010, ISBN 978 3 642 12577 5.
- Karl E. Wiegers, Software Requirements, 2E, ISBN: 0735618798.



Разпис - лекции

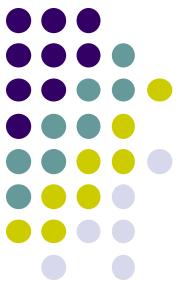
| | | | | |
|-------------|----------------|----------------------|---------------|------------------------|
| Л 1 | 20 февруари | 12:00–14:00 ч | з. 200 | Тема 1 |
| Л 2 | 27 февруари | 12:00–14:00 ч | з. 200 | Тема 3 |
| Л 3 | 12 март | 12:00–14:00 ч | з. 200 | Тема 4 |
| Л 4 | 19 март | 12:00–14:00 ч | з. 200 | Тема 5 |
| Л 5 | 26 март | 12:00–14:00 ч | з. 200 | Тема 6 |
| Л 6 | 2 април | 12:00–14:00 ч | з. 200 | Контр. работа 1 |
| Л 7 | 9 април | 12:00–14:00 ч | з. 200 | Тема 7 |
| Л 8 | 23 април | 12:00–14:00 ч | з. 200 | Тема 8 |
| Л 9 | 30 април | 12:00–14:00 ч | з. 200 | Тема 2 |
| Л 10 | 7 май | 12:00–14:00 ч | з. 200 | Тема 9 |
| Л 11 | 14 май | 12:00–14:00 ч | з. 200 | Контр. работа 2 |
| Л 1 | 21 май | 12:00–14:00 ч | з. 200 | Тема 10 |



График на упражненията

| | | |
|----------------------------------|---|-------------------|
| • Упр. 1: Въвеждащо | 17 / 20 февруари | 5 гр x 2 ч |
| • Упр. 2: Софт. инструменти | 24 / 27 февруари | 5 гр x 2 ч |
| • Упр. 3: Извличане на из-я | 2 / 5 март 16 / 19 март 23 / 26 март | 5 гр x 6 ч |
| • Упр. 4: Анализ и специфициране | 30 март / 2 април | 5 гр x 2 ч |
| • Контролна работа 1 | 2 април | 5 гр x 2 ч |
| • Упр. 5: Моделиране | 6 / 9 април 13 / 23 април 27 / 30 април | 5 гр x 6 ч |
| • Упр. 6: Валидиране | 4 / 7 май | 5 гр x 2 ч |
| • Контролна работа 2 | 14 май | 5 гр x 2 ч |
| • Упр. 7: Защита на проекти | 28 май / 1 юни | 5 гр x 6 ч |

График домашни работи



I Домашно (Инструменти) 24 февруари

II Домашно (Изисквания) 9 април

III Домашно (Модели) 7 май

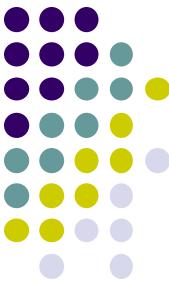
IV Валидиране 18 май

V Домашно (Презентация) 26 май / 1 юни

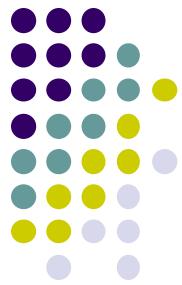
Инженеринг на изискванията

Лекция 1

Съдържание



- Основни концепции и дефиниции
- Роля на Инженеринга на Изискванията (ИИ) като част от системния и софтуерен инженеринг
- Какво инженерите по изискванията правят?
- Връзки на ИИ с
 - организационния процес
 - процеса на разработката



Какво е изискване на софтуерна система?

What is a software requirement?

Изискване на софтуерна система - 1

IEEE Stud. 610.12-1990



Дефиниция (софтуерно изискване):

- 1)Условие или способност, необходимо на потребителя да реши проблем или да постигне определена цел.
- 2)Условие или способност, което трябва да има или да осъществи системата или компонент на системата, за да удовлетвори договор, стандарт, спецификация или друг формален наложен документ.
- 3)Документално представяне на условието или способността от т.1) и т.2) .

/Клаус Пол/

Изискване на софтуерна система - 2



- **Описанието на услугите/функционалностите и на ограниченията на системата са изисквания на системата.**
- Документирано представяне на нужда, способност или качество на софтуерна система.
- **Изискванията се генерират по време на процеса на инженеринга на изискванията.**

„Прост“ пример:

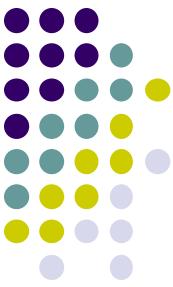
A ski resort operates several chairlifts. Skiers buy RFID-equipped day access cards. Access to the lifts is controlled by RFID-enabled turnstiles. Whenever a turnstile senses a valid access card, it unlocks the turnstile for one turn, so that the skier can pass.





Когато изграждаме софтуерна система:

- How do we determine the requirements?
- How can we analyse and document these requirements?
- How do we make sure that we've got the right requirements?
- How do we manage and evolve the requirements?

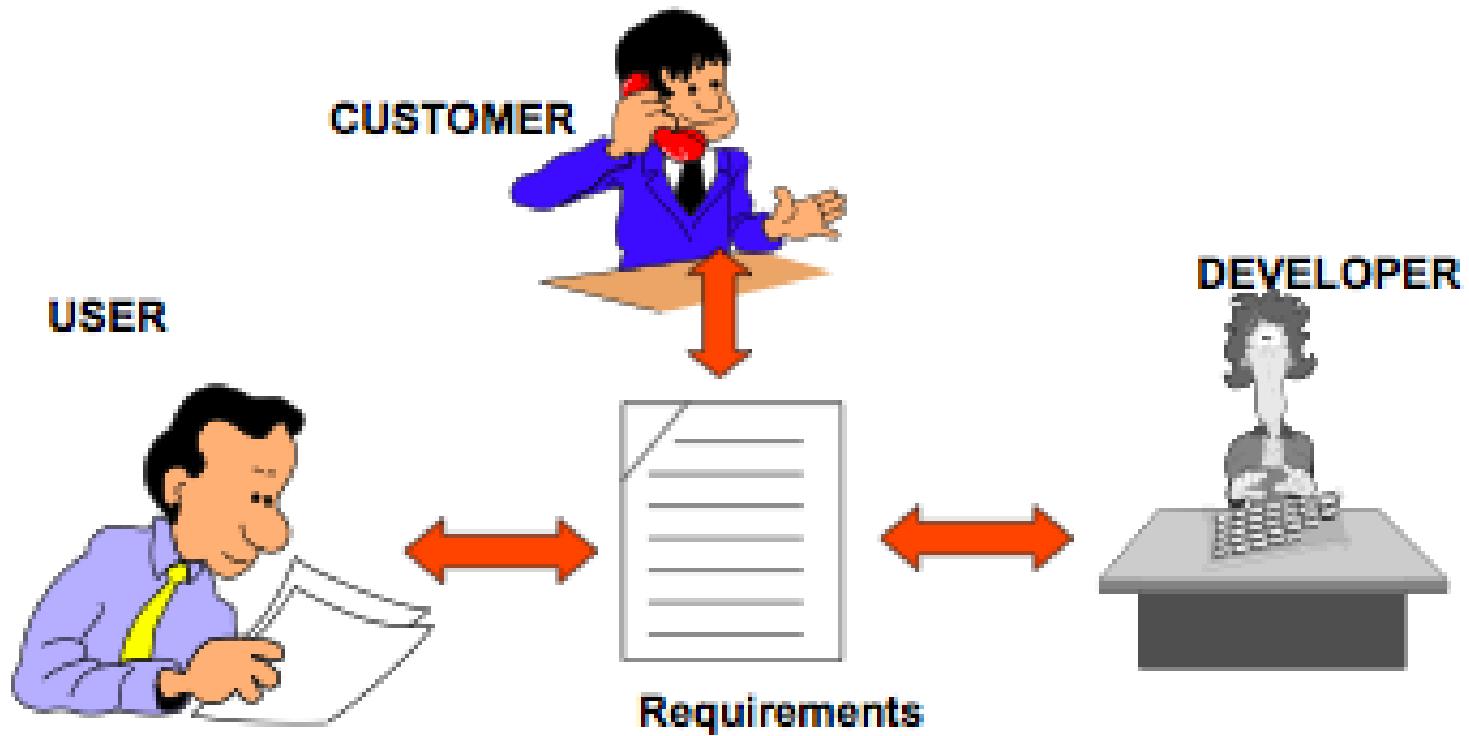


Зашо са важни изискванията?

Изискванията са как общуваме.

Те са единствената част, която всички разбират.

Основа за бъдещата разработка.



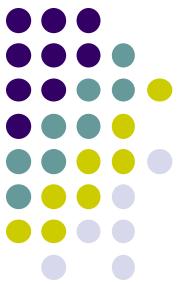


Процес на инженеринг на изискванията (Requirements Engineering process) -1

Дефиниция: Систематичен процес на идентифициране, анализиране, документиране и проверка/валидиране на функционалностите (услугите) и ограниченията на даден софтуер.

Ian Summerville

Процес на Инженеринг на изискванията - 2



- Процес на установяване и документиране на услугите, които *клиент изиска* от системата, както и на *ограниченията*, при които системата да работи и да бъде разработана.



Къде е мястото на ИИ в цялостния процес на разработка на софтуер?



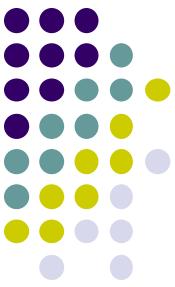
Процес на разработка на софтуерния продукт



- **Действия на ИИ**

Има няколко **основни действия на ИИ**:

- проучване за осъществимост
- извличане и анализ на изискванията
- специфициране на изискванията
- валидиране на изискванията
- управление на изискванията



ИИ – същност 1:

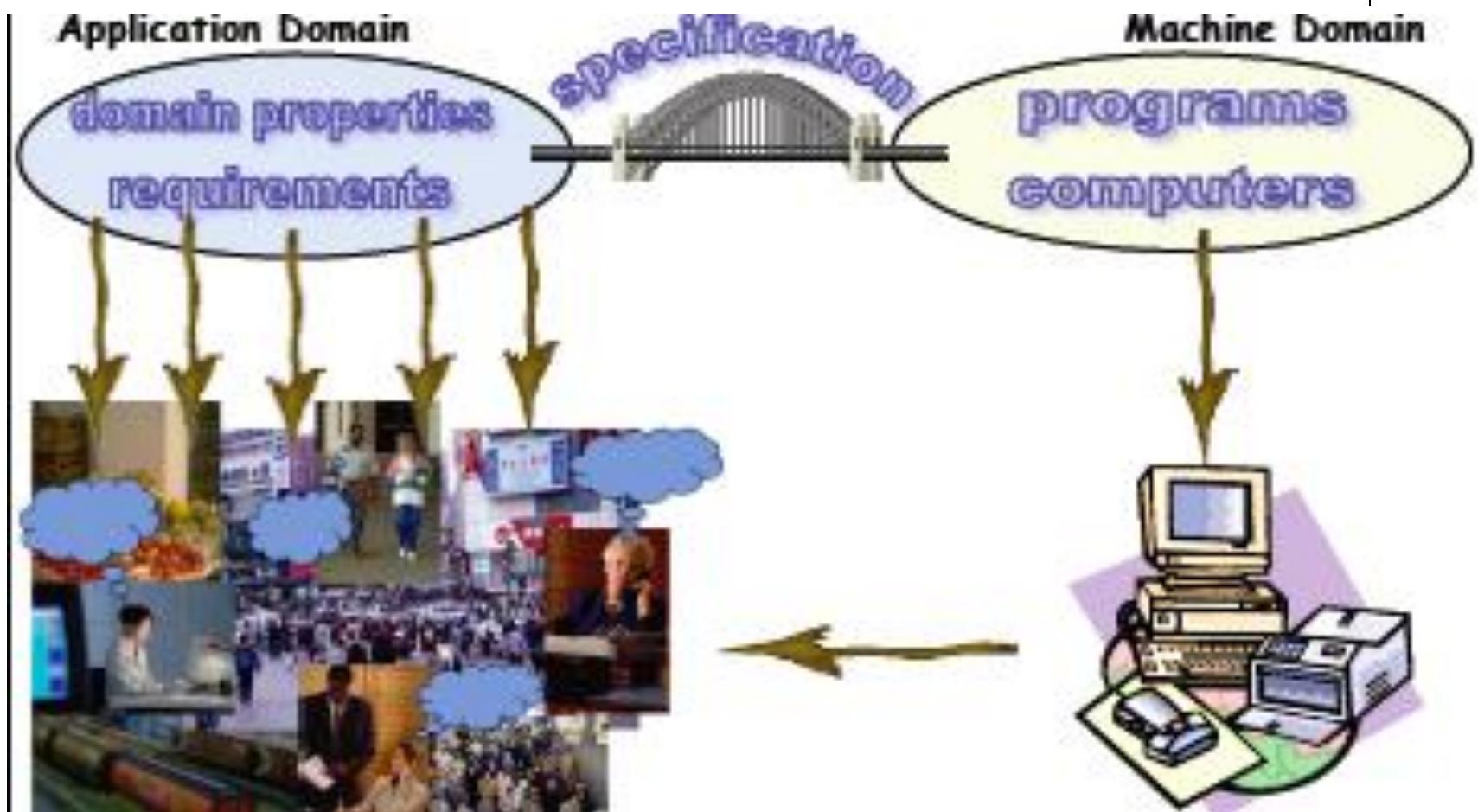
“RE is a set of activities concerned with identifying the purpose of a software system, and the contexts in which it will be used. Hence, **RE acts as the *bridge* between the real world needs of users, customers, and other constituencies affected by a software system**, and the capabilities and opportunities afforded by *software-intensive technologies*.”

Steve Easterbrook

ИИ е в ролята на **мост** между нуждите на клиентите, ползвателите и останалите заинтересовани от работата на създаваната софтуерна система и техническите възможности, предлагани от (интензивните) софтуерните технологии.



Предизвикателства?





ИИ – същност 2

Not a phase
or stage

Communication
is important as
analysis

Quality means
fitness for purpose.

Cannot say
anything about
quality unless you
understand the
purpose.

Why it is “requirements” ?
Why it is “engineering” ?

(Kotonya, Sommerville, p.8)

Requirements Engineering (RE) is a set of activities concerned with identifying and communicating the purpose of a software-intensive system, and the contexts in which it will be used. RE acts as the bridge between the real-world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies.

Need to identify **all the stockholders**
not just the customer and user.

Application of a systematic, disciplined, quantifiable approach to the specification and management of requirements; that is the application of engineering to requirements¹⁵

Designers need to
know **how and where** the system
will be used.

Requirements are
partly about what we need ...
... and partly about what is possible.



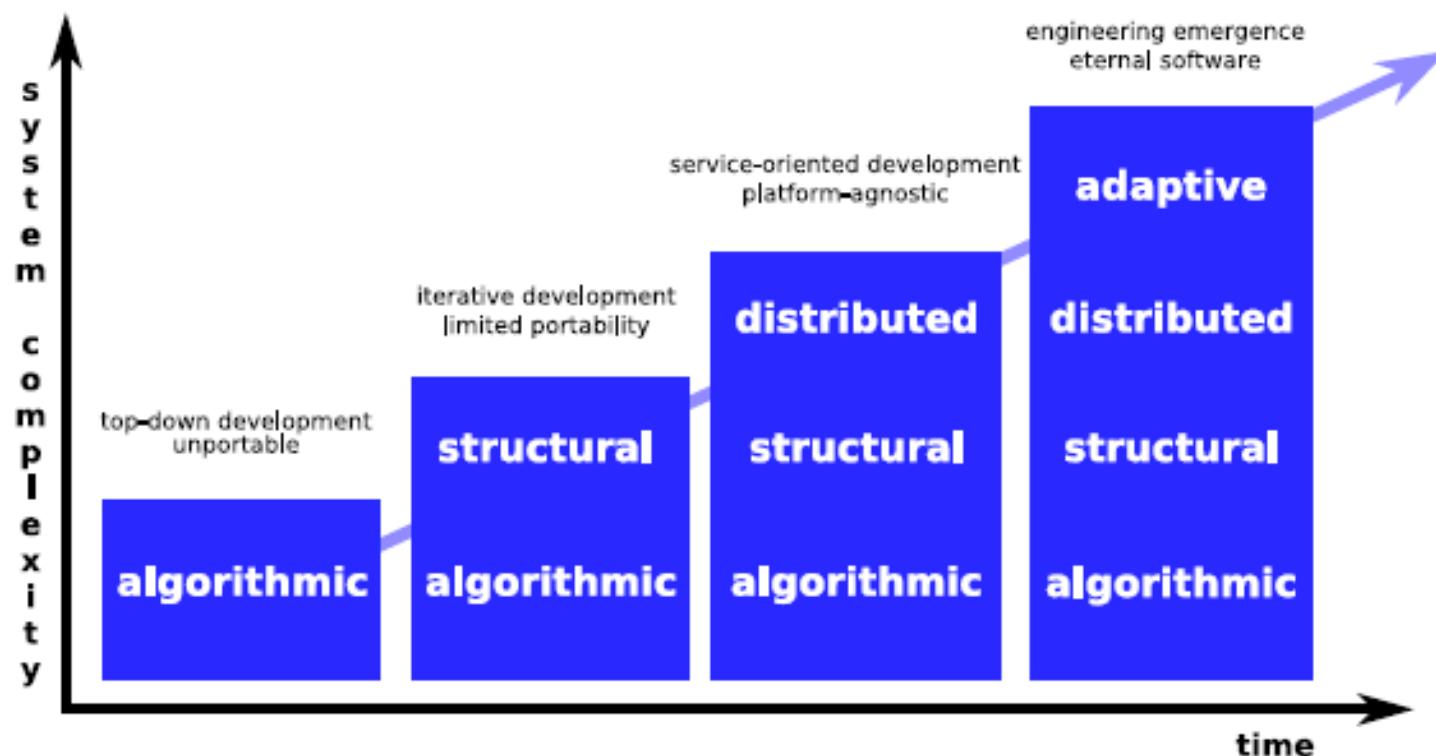
Software-intensive systems – 1

- Software (on its own) is useless
 - Software is an abstract description of a set of computations
 - Software only becomes useful when run on some hardware
 - Software + Hardware = Computing system (**why system?**)
- A computer system (on its own) is useless
 - Only useful in the context of some *human activity* that it can support
 - A new computer system will change human activities in significant ways
- **Software + Hardware + Human activities = Software-intensive system**

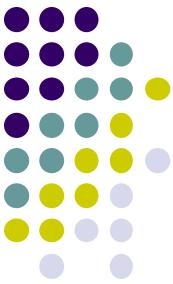


Software-intensive systems 2

- Software makes many challenges during the time and today:
 - It is complex and adaptable
 - It can be rapidly changed on-the-fly
 - It turns general-purpose hardware into a huge variety of useful machines



Каква е връзката между изисквания и проектирането?



- Два основни (*парадоксални* ☺) принципа:

“Requirements and design are interleaved. They should, ideally, be separate processes but in practice this is impossible.”

Somerville

- Полезно е да **разделим** проблема от решението
 - и да документира проблема отделно от всички дизайнерски решения
НО!
- Това разделяне **никога не може да се постигне** в пълна степен
 - защото дизайнът променя света и следователно променя първоначалния проблем ☺.

Изискванията се откриват, а не намират - 1



“Requirements cannot be observed or asked for from the users, but have to be created together with all the stakeholders.”

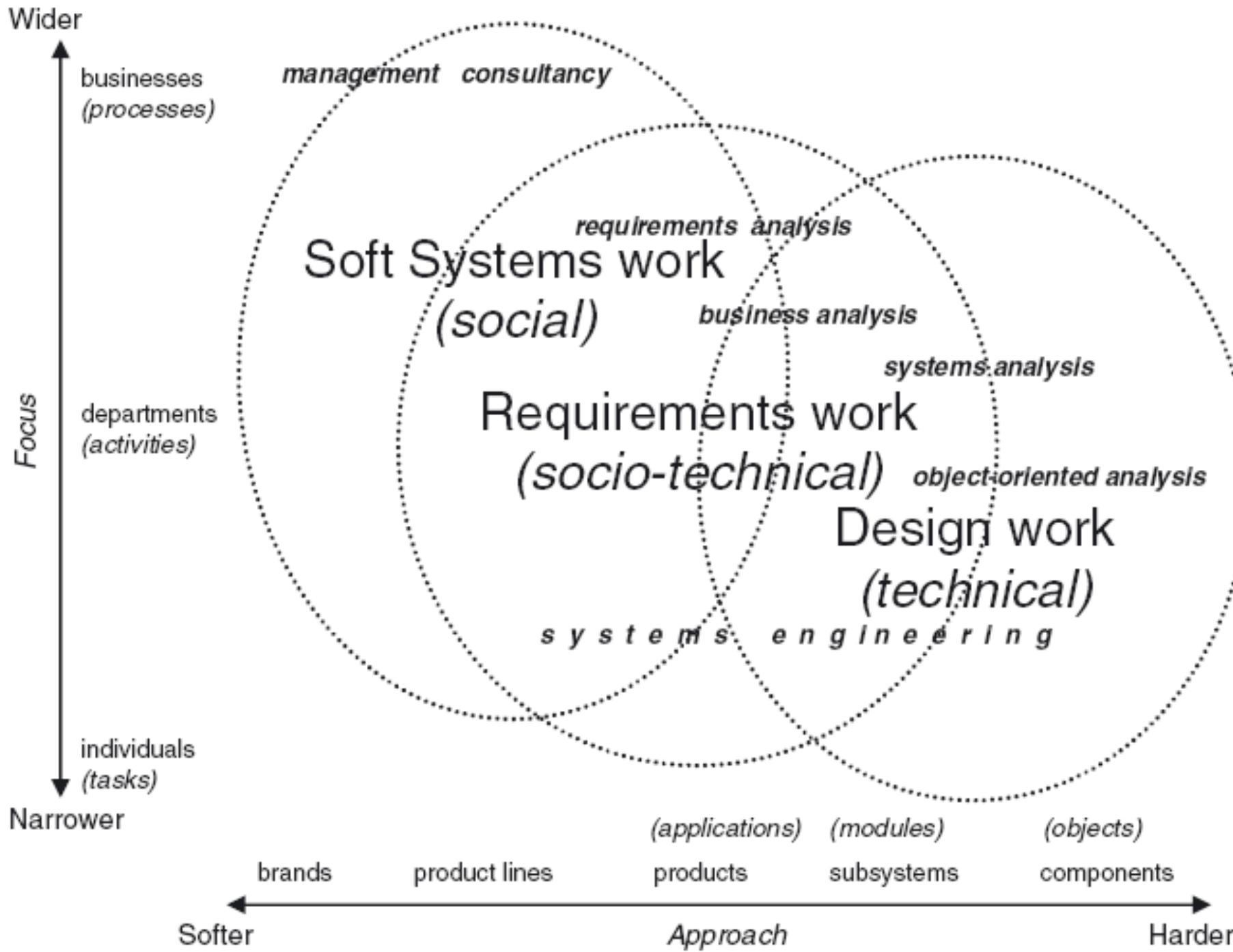
Vesa Torvinen

- So far we used words like requirements:
 - capture (обхващам)
 - gathering (събирам в сбирка, колекция ...)
 - trawling (трапене)
 - elicitation (извлечане, изваждане, разкриване)

Изискванията се откриват, а не намират - 2



- “Откриване” означава много и различни дейности:
 - Търсене в доказателствата (looking at the evidence);
 - Нови идеи (being open to new ideas);
 - Креативност (applying creative effort);
 - Работа в екип (working as a team);
 - Фокусирани въпроси (asking questions that focus the search);
 - Да се намери специфичното (intending to find particular kinds of thing);
 - Да се представи в разумна рамка на разглеждане (fitting whatever is found into a reasoned framework);
 - Отнасяне към подобни случаи (relating whatever is found to similar discoveries).
 - Примери!

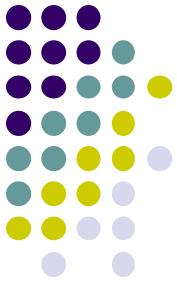




Какво е спецификация на изискванията?

What is a requirement specification?

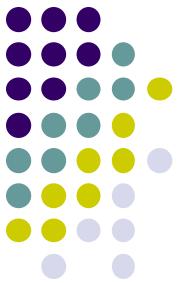
Спецификация на изискванията - 1



- The set of all requirements forms the basis for subsequent development of the system or system component.

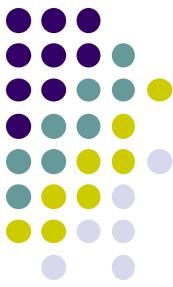
Isn't it all?

Спецификация на изискванията - 2



Спецификацията е повече от списък ...

- Това е **мрежа от свързани, зависими елементи на изискванията**, включващи дефиниции, цели, обосновки, измервания и друга информация
- ‘The requirements’ in the broad sense means a network of *interrelated requirement elements*: a requirement that satisfies a goal, is justified in a **rationale model**, using terms defined in the project dictionary, etc.



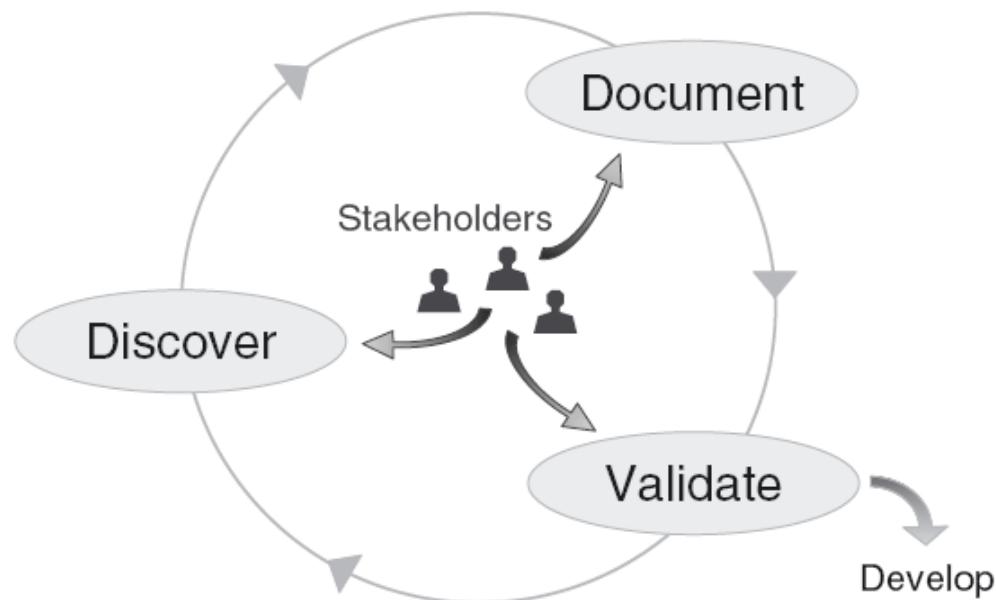
Discovery as search

- Начин на мислене (A good way of thinking about discovery is as a search).
- Структуриране на знанието. (The structure of what you know drives what you discover next).
- **Колкото по-добре е организирано знанието за изискванията, толкова по-добре ще може да се открие нова, което наистина е необходимо.**
- Например:
 - Формулирай целта (на системата, функционалността, услугата).
 - Използвай сценарии, за да изследваш как можеш да постигнеш целта
 - Търси изключенията, който може да се срещнат.



Цикъл на откриването

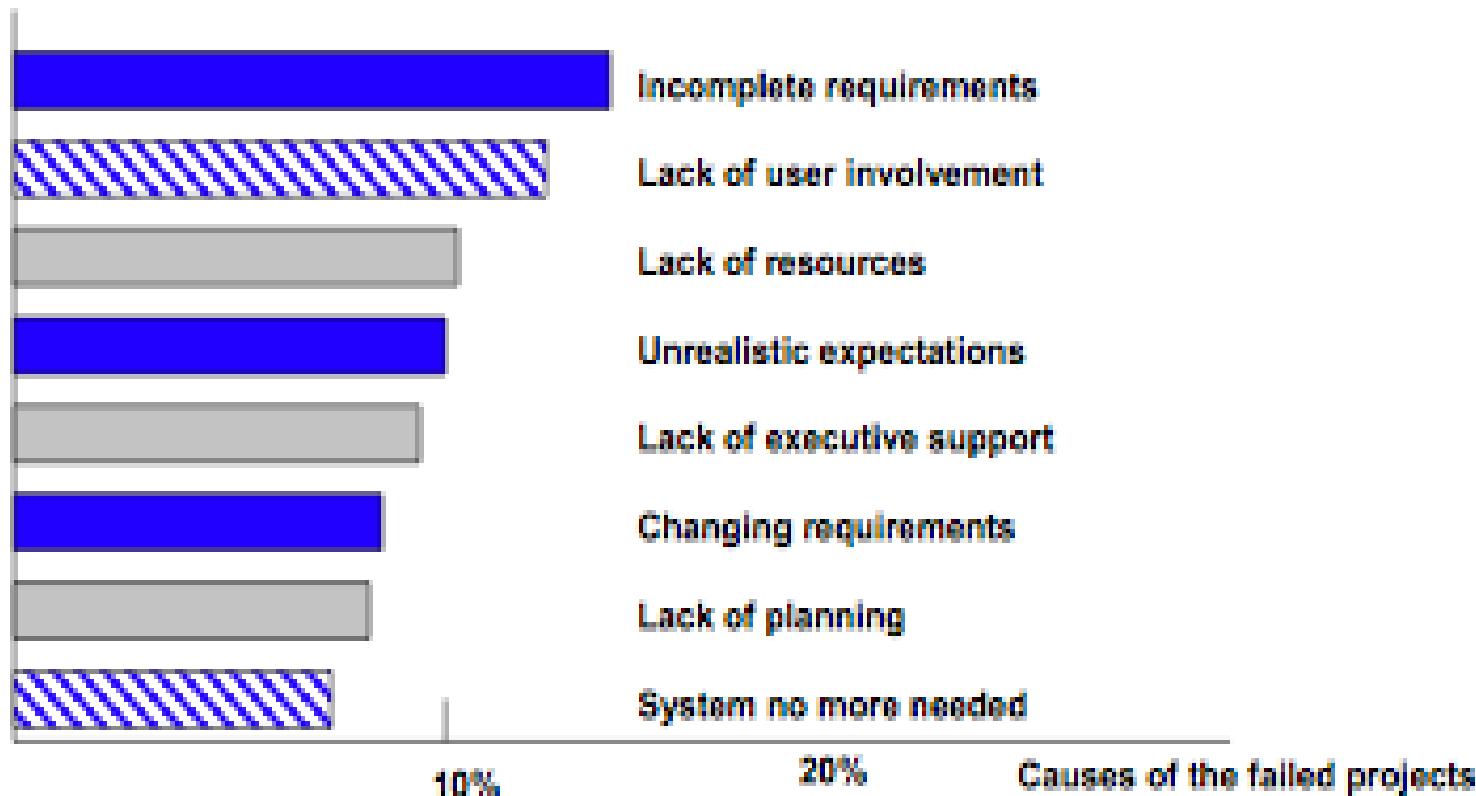
- A generic discovery process can be drawn as a simple inquiry cycle
- An inquiry cycle is more or less what it says:
 - Цикъл от дейности
 - Екипна работа
 - Възможност да се провери ефективността





Зашо изискванията са (много) важни?

- Колко струва ИИ? *About 15% of system development costs?*
- НО!**
- *Непълните и неконсистентни изисквания* са най-честите причини за проблеми на системата.





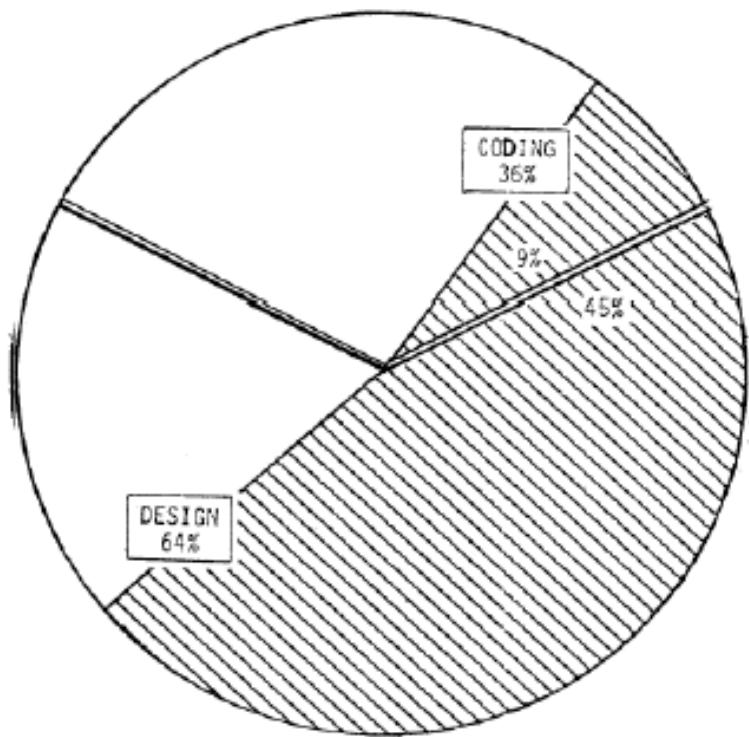
Мотивация за ИИ

- “Errors are more frequent during requirements and design activities and are the more expensive the later they are removed”

Wording by [Endres2003]



Barry Boehm
[Image: Wikipedia]



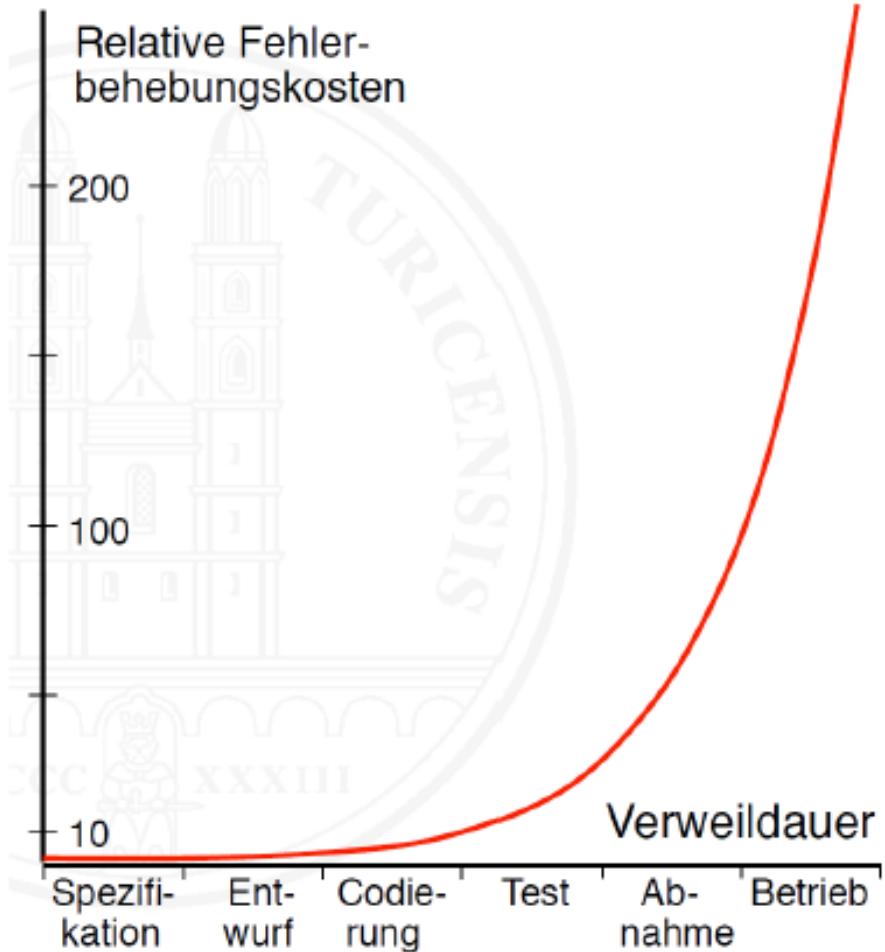
ERRORS FOUND
DURING OR AFTER
ACCEPTANCE TEST

Design: Errors fixed by changing the design
Coding: Errors fixed by changing only the code

From [Boehm 1975]



“Fix it later” is expensive



- Cost for removing errors depends on how long it stays in the software
- The later errors are detected, the more expensive their removal is
 - E.g. due to required redesign and implementation
- Good RE avoids requirements errors and thus saves cost for removing errors later



Importance of requirements engineering

Barry Boehm investigated the cost to fix errors in the development of large software systems.

- **Cost of fixing errors**
 - Typical development process
 - Errors cost more to fix the longer they are undetected
- Causes of
 - **Top three success factors are:**
 - user involvement;
 - executive management support;
 - a clear statement of requirements.

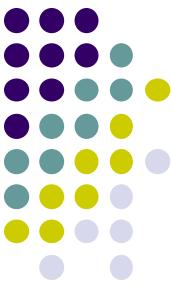
The top three factors leading to failure were:

- *lack* of user input;
- *incomplete requirements* and specifications;
- *changing* requirements and specifications.



Some observations about RE 2

- Perfecting a specification may not be cost-effective
 - Requirements analysis has a cost – (**how much?**)
 - For different projects the cost-benefit *balance* will be different
- The problem statement should *never* be treated as fixed
 - Change is inevitable, and therefore must be planned
 - There should be a way of incorporating changes periodically



Systems Engineering vs. Software Engineering

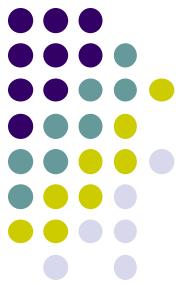
Инженеринга на изискванията се прилага както и към софтуерни системи, така и към системите изобщо.

- For software engineering, it is normally assumed that the way in which the software interacts with the world is fixed, using standardized types of input and output device.
- For systems engineering, no such assumptions are made – the task is to design an entire system, of which the *software is just one component*.



What do requirements engineers do?

Какво правят инженерите (аналитиците) по изискванията?



- Изявяват се в началото на проекта
- Идентифицират проблем, който се нуждае от решение:
 - Недоволството от текущото състояние
 - Нова възможност за бизнес
 - Потенциал за спестяване на разходи
- *Анализът* на изисквания е агент на промяната



Какво правят инженерите по изискванията? -2

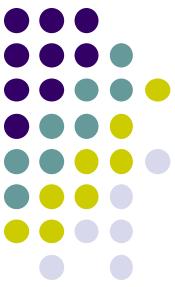
- Инженерите по изискванията трябва да:
 - Идентифицират проблема/възможността
 - Which problem needs to be solved? (identify problem boundaries)
 - Where is the problem? (understand the Context/Problem domain)
 - Whose problem is it? (identify stakeholders)
 - Why does it need solving? (identify the stakeholders' goals)
 - How might a software system help? (collect some scenarios)
 - When does it need solving? (identify development constraints)
 - What might prevent solving it? (identify feasibility and risk)
 - Да бъдат/станат експерти в проблемната област



Какво правят инженерите по изискванията?

– взаимоотношения 1

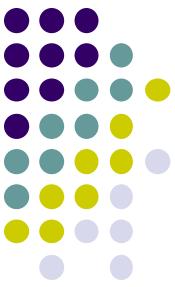
- С други субекти на организацията на процеса
 - Product management
 - Marketing
 - Customer relationship management
- Фокусът е в дейности по търсене и записване на информация :
 - in RE process of making sense of the answers, requirements engineers may *build models, and test them* in various ways.
 - They *will communicate and negotiate* with a variety of stakeholders, to reach agreement on the nature of the problem.
 - They will help to *bridge* the gap between the problem and the solution,
 - and *manage the evolution* that takes place as the problem changes.



взаимоотношения 2

- С други субекти на разработването

- Project management
- Design
- Quality assurance
- System maintenance



Simple but not easy

- Requirements work is simple but not easy:
it is a *craft*, which requires skills.
- ... Fortunately, that can be learned !



■ Lots of sources for today's terminology

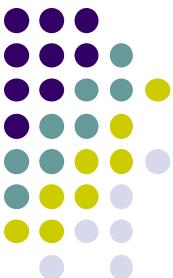
- Textbooks and articles about RE
- IEEE 610.12 (1990) – a slightly aged glossary of software engineering terminology
- IEEE 830-1998 – an outdated, but still cited RE standard
- ISO/IEC/IEEE 29148 (2011) – a new, but still rather unknown RE standard; provides definitions of selected terms, some of them being rather uncommon
- IREB Glossary [Glinz 2013] – influential through IREB's certification activities; used as a terminology basis in this course



Процес на инженеринг на изискванията

Лекция 2

Съдържание



- Процеси на инженеринга на изискванията (ИИ)
- Модели на процеси на ИИ
- Организационни, човешки и социални фактори в процесите на ИИ
- Защо е важно да се подобряват процесите на ИИ
- Модел за подобряване на процесите на ИИ



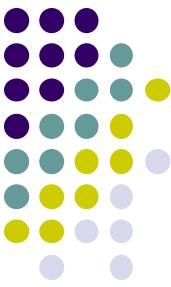
Процес

Дефиниция: Процес е организиран набор от дейности, които преобразуват входа за получаване на изход.

- Описанията на процес са **капсулира знание**, което да бъде **използвано повторно**.
- *Ниво на детайлност* на описанието според сложността на процеса
- Човешкият фактор и адаптация към обстановката, взаимодействие с други процеси, промяна в наличната входна информация.

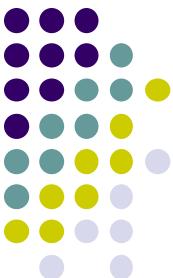
Пр. : Кои са входовете и изходите на процесите, описани в:

- Инструкция за съдомиялна машина
- Готварска книга (рецепта)
- Наръчник за банкови операции (напр. вземане на кредит)
- Наръчник по качеството за разработка на софтуер

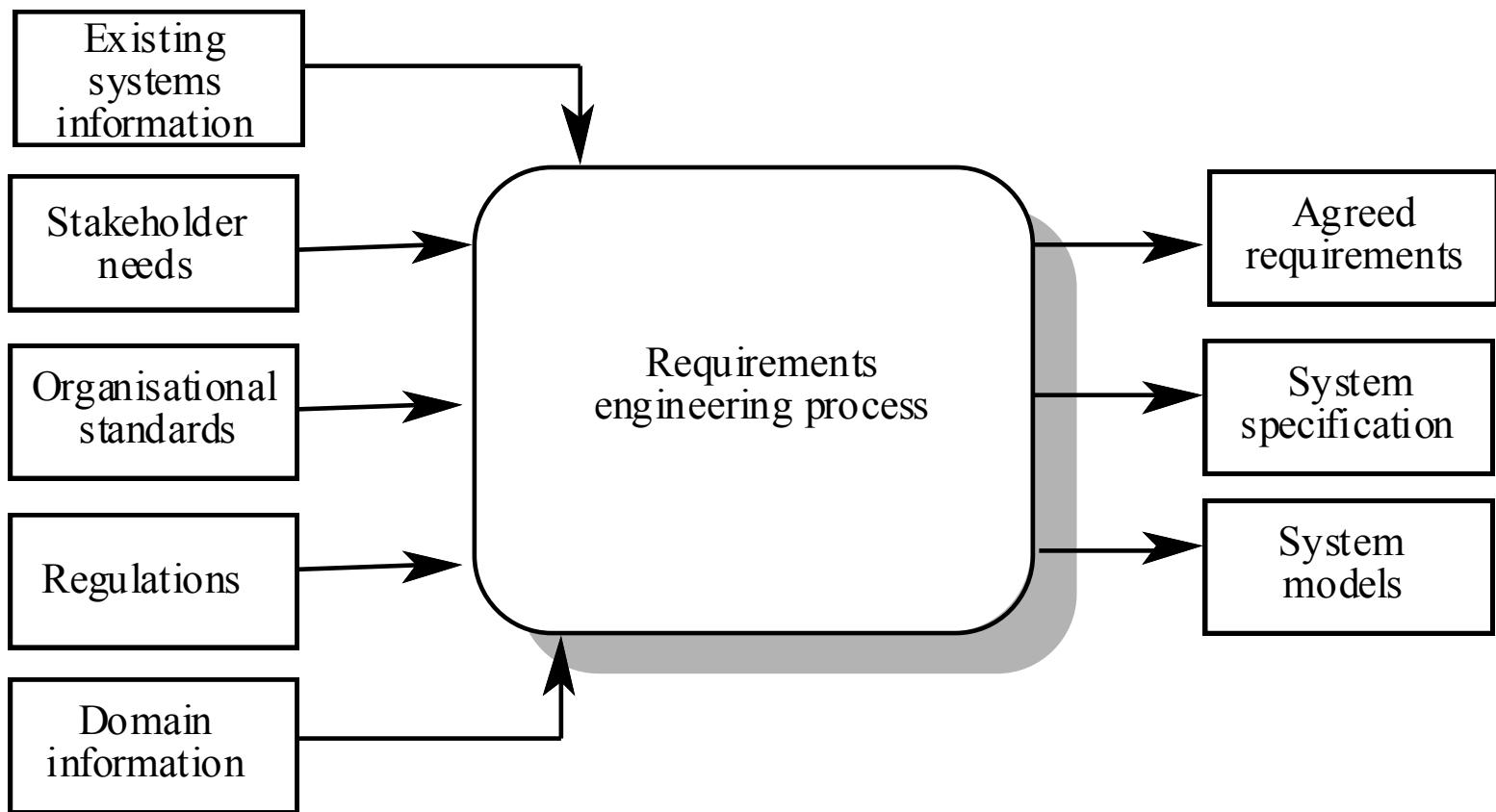


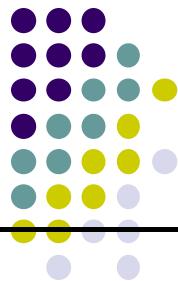
Процес на проектиране

- Процесите на проектиране изискват *творчество, взаимодействие* между много и *различни* хора с различна инженерна преценка, (общи) познания и опит.
- Неточно/непълно дефиниран вход
- Много възможни изходи
- Невъзможна е пълна автоматизация и пълно детайлно описание на процеса
- *Примери*
 - Писане на книга
 - Организиране на конференция
 - Проектиране на процесорен чип
 - **Инженеринг на изискванията**
- ...



Процеса на инженеринг на изискванията е процес на проектиране - описание тип „черна кутия“:





Входове и изходи на описанието

| Input or output | Type | Description |
|-----------------------------|-------------|--|
| Existing system information | Input | Information about the functionality of systems to be replaced or other systems which interact with the system being specified |
| Stakeholder needs | Input | Descriptions of what system stakeholders need from the system to support their work |
| Organisational standards | Input | Standards used in an organisation regarding system development practice, quality management, etc. |
| Regulations | Input | External regulations such as health and safety regulations which apply to the system. |
| Domain information | Input | General information about the application domain of the system |
| Agreed requirements | Output | A description of the system requirements which is understandable by stakeholders and which has been agreed by them |
| System specification | Output | This is a more detailed specification of the system functionality which may be produced in some cases |
| System models | Output | A set of models such as a data-flow model, an object model, a process model, etc. which describes the system from different perspectives |

Примери?



Вариабилност на процеса по ИИ

Няма идеален процес **на ИИ!**

- Процесът на ИИ е различен в голяма степен в отделните организации
- Фактори на вариабилността са
 - *Техническа зрялост*: технологии и методи, използвани за ИИ
 - *Организационна култура*: други дейности, ограничени срокове, ...
 - *Област на приложение*: определя различни подходи при ИИ, *различни вход и изход*.
 - *Дисциплинираността* (управленска и инженерна) на участниците и организацията като цяло.



Модел на процеси на ИИ

Дефиниция: Модел на процес е опростено описание на процес, обикновено представен от конкретна перспектива.

- Нито един модел **не дава** пълно разбиране за процеса (зашо?)
- Видове модели на процес на ИИ:

А. Според детайлността:

Модел на дейностите (Activity model)

А.1 Модели на общо определена дейност (Coarse-grain activity models)

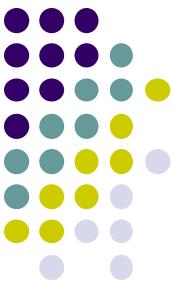
А.2. Модели на детайлно специфицирани дейности (Fine-grain activity models)

Б. Според перспективата:

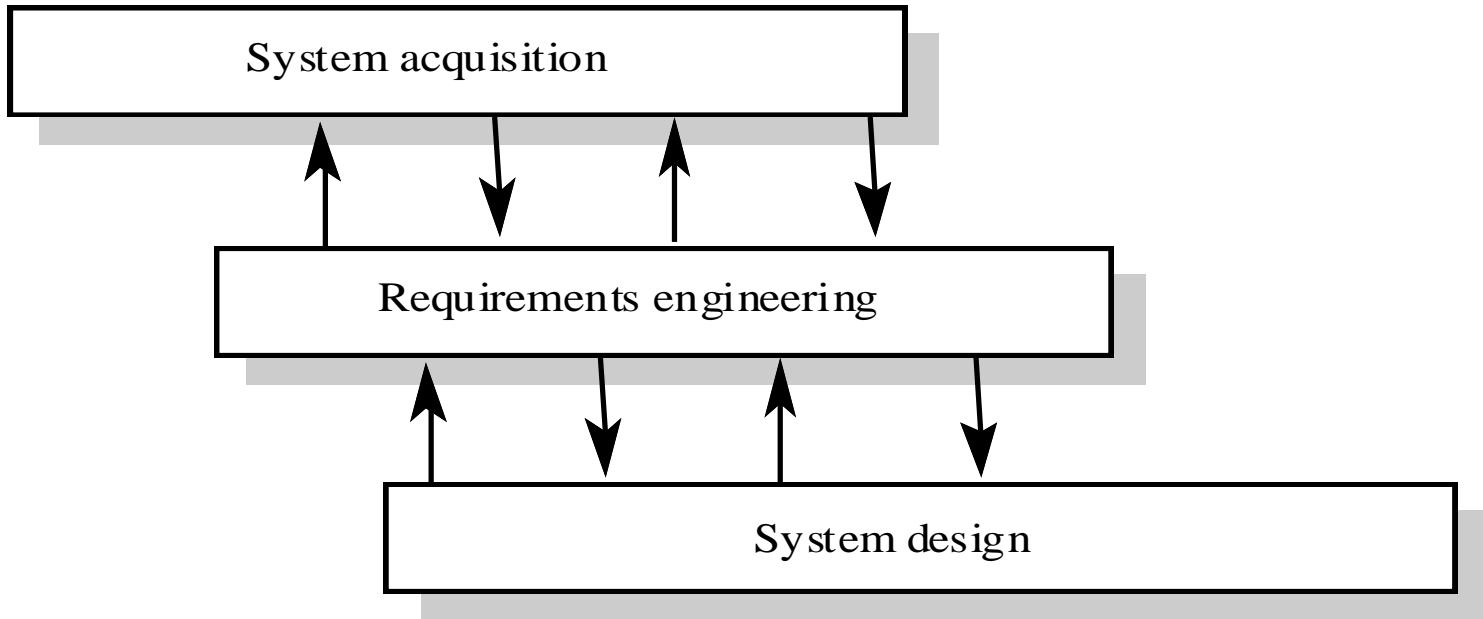
Б.1. Модели роля-действие (Role-action models)

Б.2. Модели същност-връзка: вход, изход и междинни резултати.

От какво зависи изборът на модел?



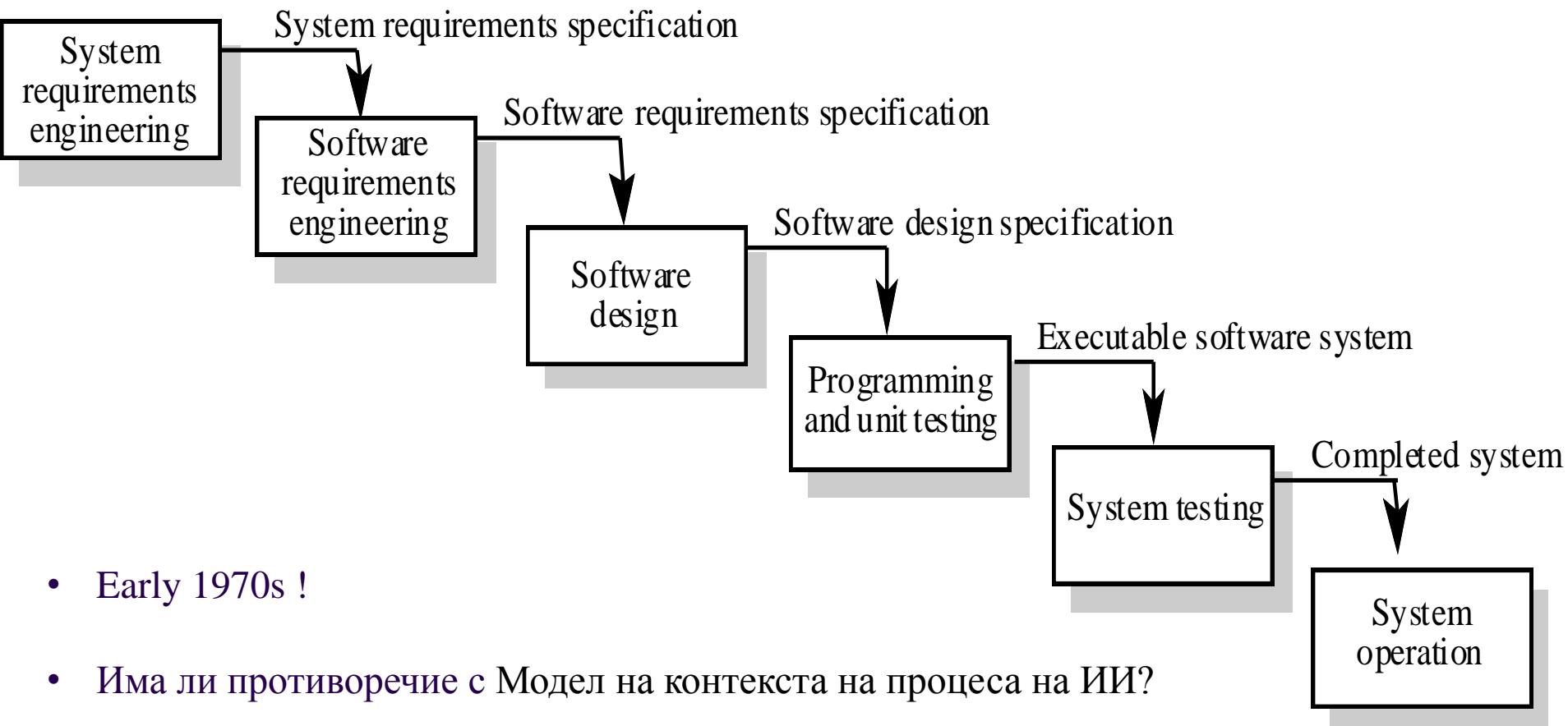
A.1 пример: Модел на контекста на процеса на ИИ дефинира информационните потоци



A.1 V модел на ИИ, V модел на софтуерен процес



A1.1. Пример: Водопаден модел на софтуерния процес /waterfall software life cycle model/

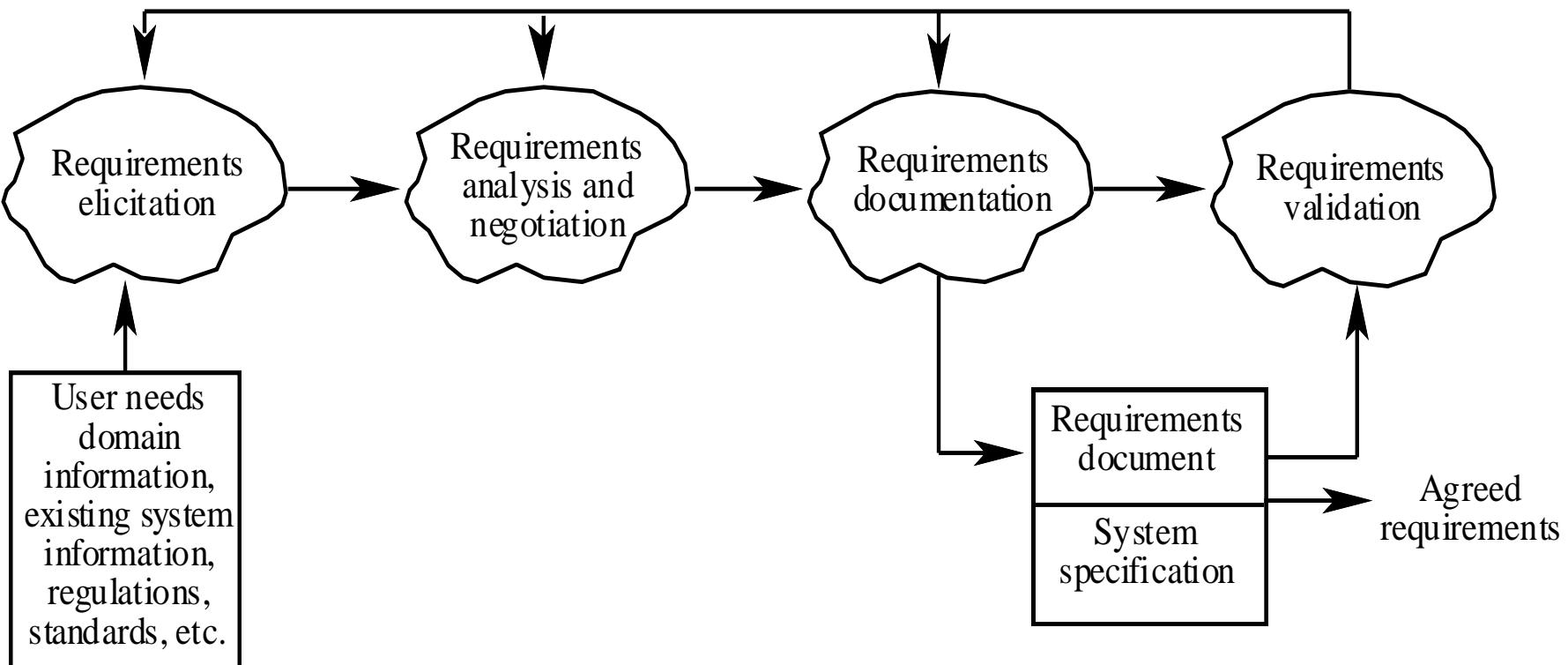


- Early 1970s !
- Има ли противоречие с Модел на контекста на процеса на ИИ?



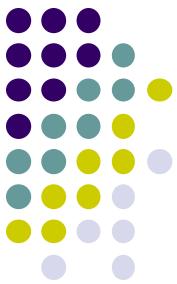
A.1 Пример: Модел на (общите) дейности на инженеринга на изискванията (Coarse-grain activity model of RE)

- общовалиден модел. Защо?



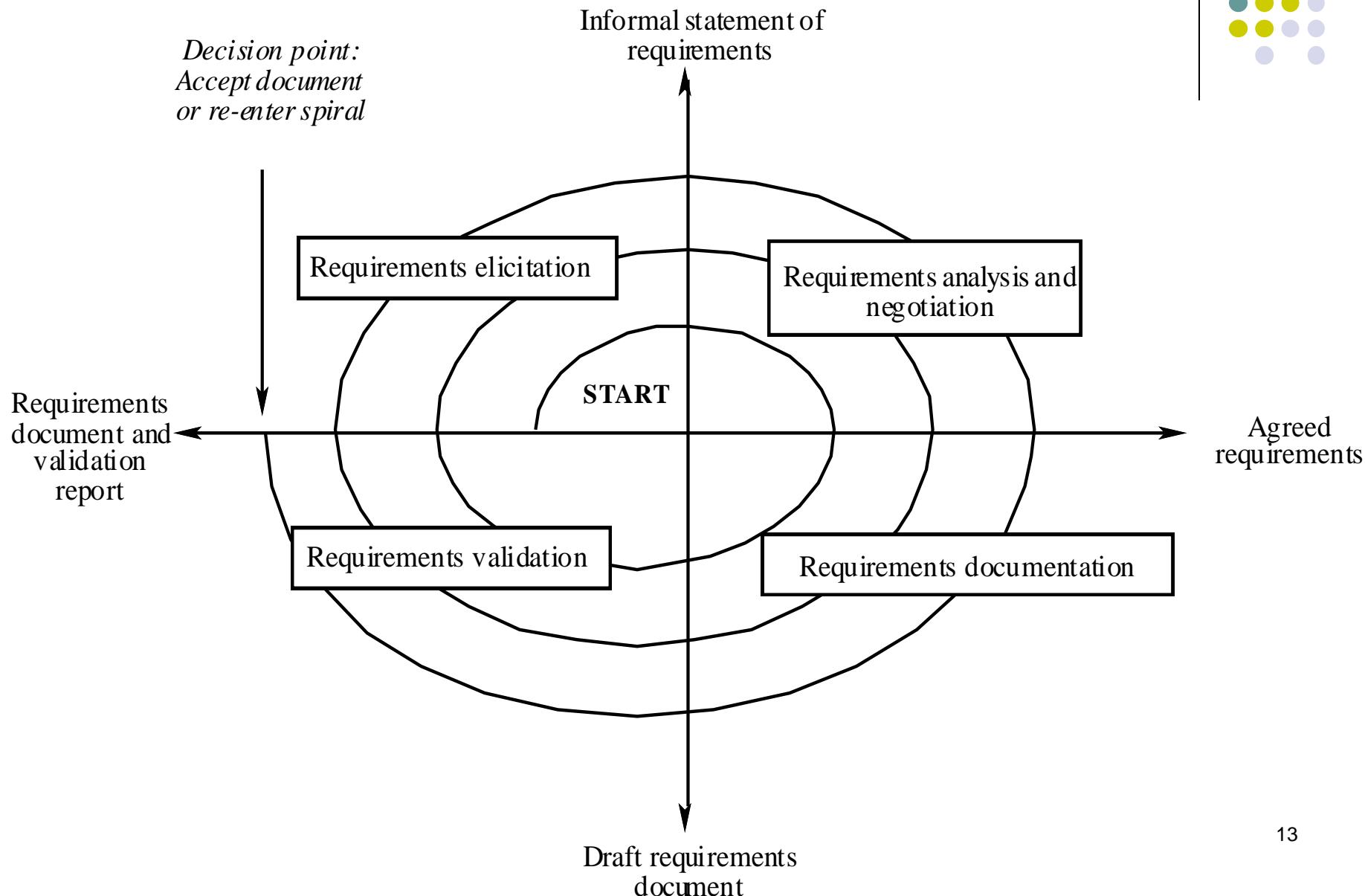
Cloud icon – why?!

Действията на процеса на ИИ



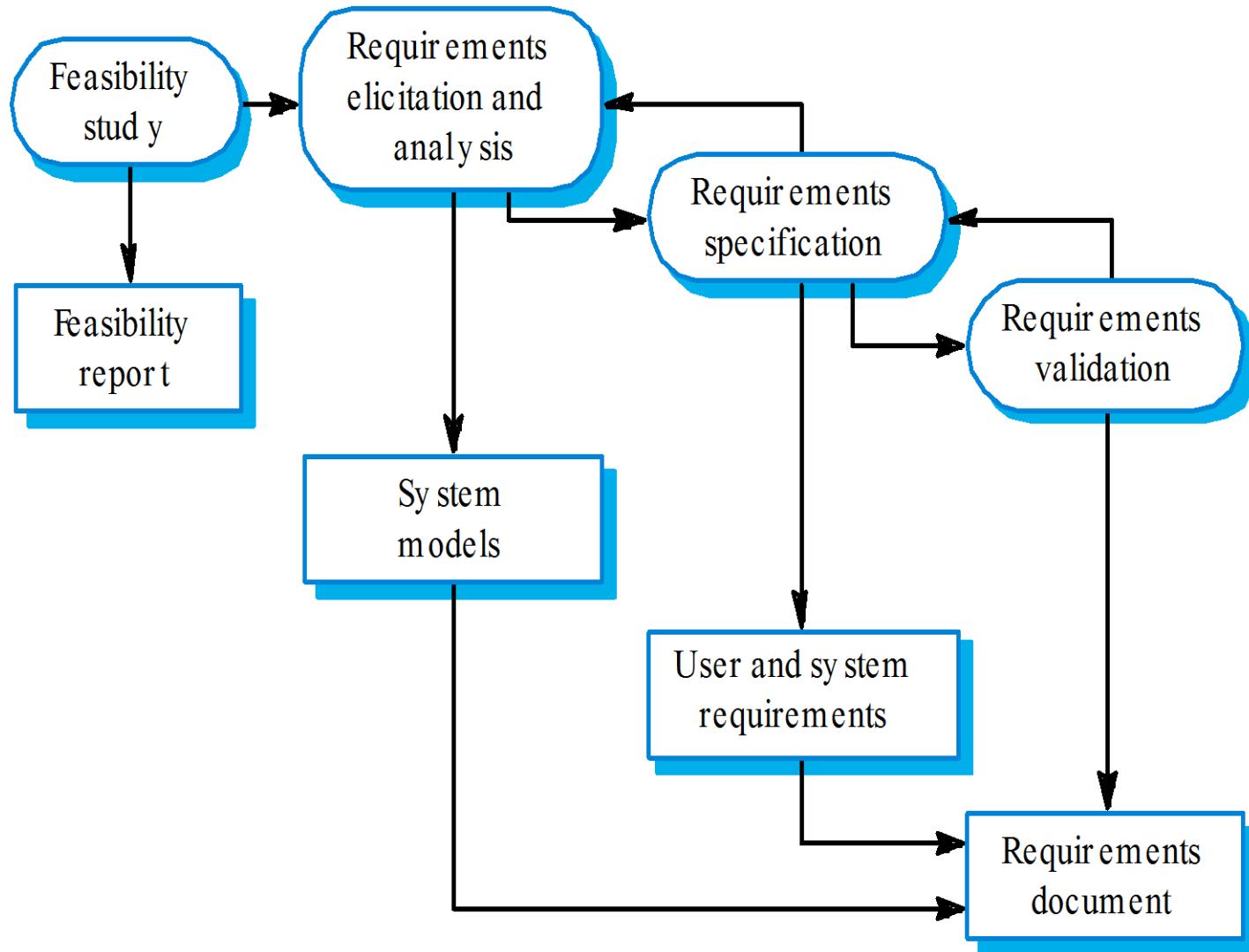
- **Извличане на изискванията (*Requirements elicitation*)**
Изисквания се откриват чрез консултации със заинтересованите страни и чрез други знания и информация.
- **Анализ на изисквания и преговори на изискванията (*Requirements analysis and negotiation*)**
Изискванията се анализират; конфликтите се решават чрез преговори.
- **Документиране на изискванията (*Requirements documentation*)**
Оформя се и се представя документ с изискванията.
- **Валидиране на изискванията (*Requirements validation*)**
Документът на изискванията се проверява за съгласуваност и пълнота.
- **Управлението на изискванията (*Requirements management*)** е успореден с всички по-горе описани процеси и управлява промените в изискванията.

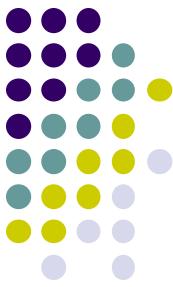
A1.1 Спирален модел на процеса на ИИ – алтернативен начин на представяне на дейностите





A.1. Процес на инженеринг на изискванията by Summerville





Б.1 Актьори в процеса на ИИ. Модел роля-дейност

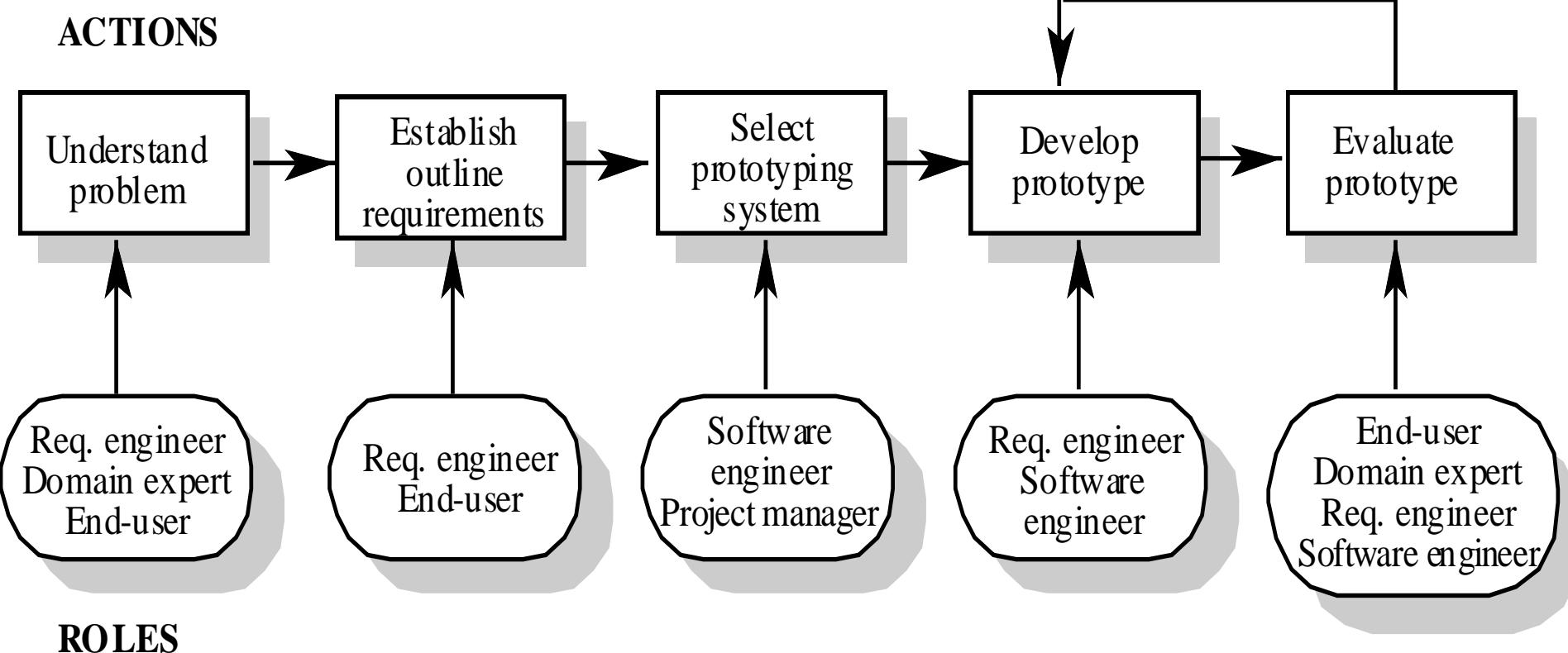
- Актьори в процеса са хората, които участват в ИИ.
- Актьорите обикновено се идентифицират с *техните роли в изпълнението на конкретна дейност*, а не персонално: project manager, system engineer ...
- ИИ включва актьори, които
 - 1) се интересуват от проблема, за да бъде той решен (крайни потребители)
 - 2) актьори, които се интересуват от начина за решаване на проблема (системните дизайнери, разработчици и мениджъри);
 - 3) хора, зависими от съществуването на системата (поддръжка на системата, здравни и регуляторни органи по сигурността за safety-critical systems)

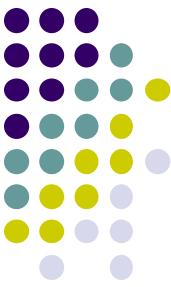
Примери: Дайте примери на актьори за системата на седалков лифт. Ниво на креативност?

Диаграми на ролите (Role-action diagrams (RAD)) е документ, който показва как участниците са ангажирани в различни дейности. Приложение.



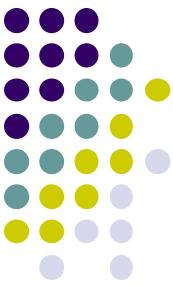
Б.1 Пример: Диаграма на ролите за създаване на прототип (RAD for software prototyping)





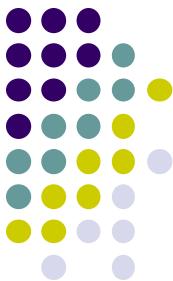
Пример - Описание на ролите -2 (Role descriptions)

| Role | Description |
|-----------------------|---|
| Domain expert | Responsible for providing information about the application domain and the specific problem in that domain which is to be solved. |
| System end-user | Responsible for using the system after delivery |
| Requirements engineer | Responsible for eliciting and specifying the system requirements |
| Software engineer | Responsible for developing the prototype software system |
| Project manager | Responsible for planning and estimating the prototyping project |



Човешки и социални фактори

- Процесите на ИИ са *доминирани от човешки, социални и организационни фактори*, резултат от необходимостта за включване на широк кръг заинтересовани лица от различни области и с различни индивидуални и организационни цели.
Сравнете с други софтуерни процеси (QA, програмисти)?
- Заинтересованите лица могат да имат определен обхват от *технически и нетехнически умения (background)* и от *различни области*.



Примери за различни заинтересовани лица

- **Софтуерните инженери**, отговорни за разработването на системата
- **Крайните потребители**, които ще използват системата
- **Мениджъри** на крайните потребители, които са отговорни за тяхната работа
- **Външни регуляторни служби/органи**, които проверяват дали системата отговаря на законови изисквания
- **Експерти в областта**, които дават съществена допълнителна информация за областта на приложение на системата

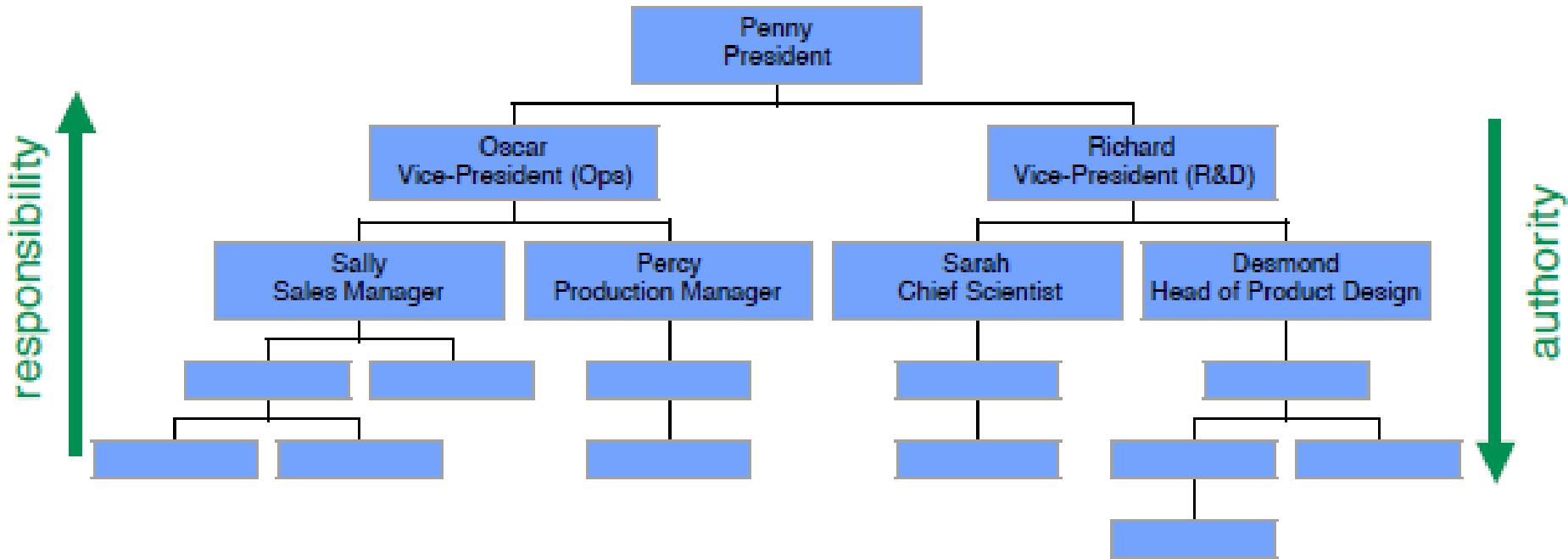


Човешки фактори, които влияят върху изискванията

- Личност и статут на заинтересованите лица
 - Пример: силен/слаб характер
- Личните цели на лицата в рамките на една организация, техните текущи задължения.
 - Пример: трудна за реализиране функционалност...
- Степен на политическо влияние на заинтересованите страни в рамките на една организация
 - Пример: администрация и изпълнителски екип ...



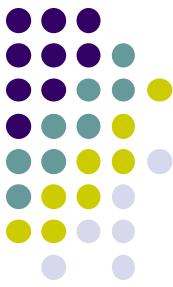
Finding stakeholders: The Org Chart





Подпомагане на процесите на ИИ (Process support)

- CASE (Computer Aided Software Engineering) са инструменти, които *автоматизират* процесите на софтуерното инженерство (след 1980)
- Най-добре разработени CASE инструментите поддържат *общи* за организациите дейности, които са *добре познати и ясни* като напр. за програмиране, проектно планиране, функционално-ориентирано проектиране, тестване и използването на структурни методи – примери?
- CASE за ИИ са по-ограничени, заради степента на неформалност и променливост на процеса.



CASE инструменти за ИИ

Два типа CASE инструменти:

1. за целите на *моделиране и валидиране* при изграждане на модели на системи, използвани за специфициране на системата и за проверка за пълнота и съгласуваност на системата.

- SADT или специализирани езици за описание на изискванията (RSL);
- Спецификации! (а не инструменти): UML модели, VDM, Z notation

2. за *управление*: помагат управлението на база данни от изисквания и подпомагат управлението на промените на тези изисквания.

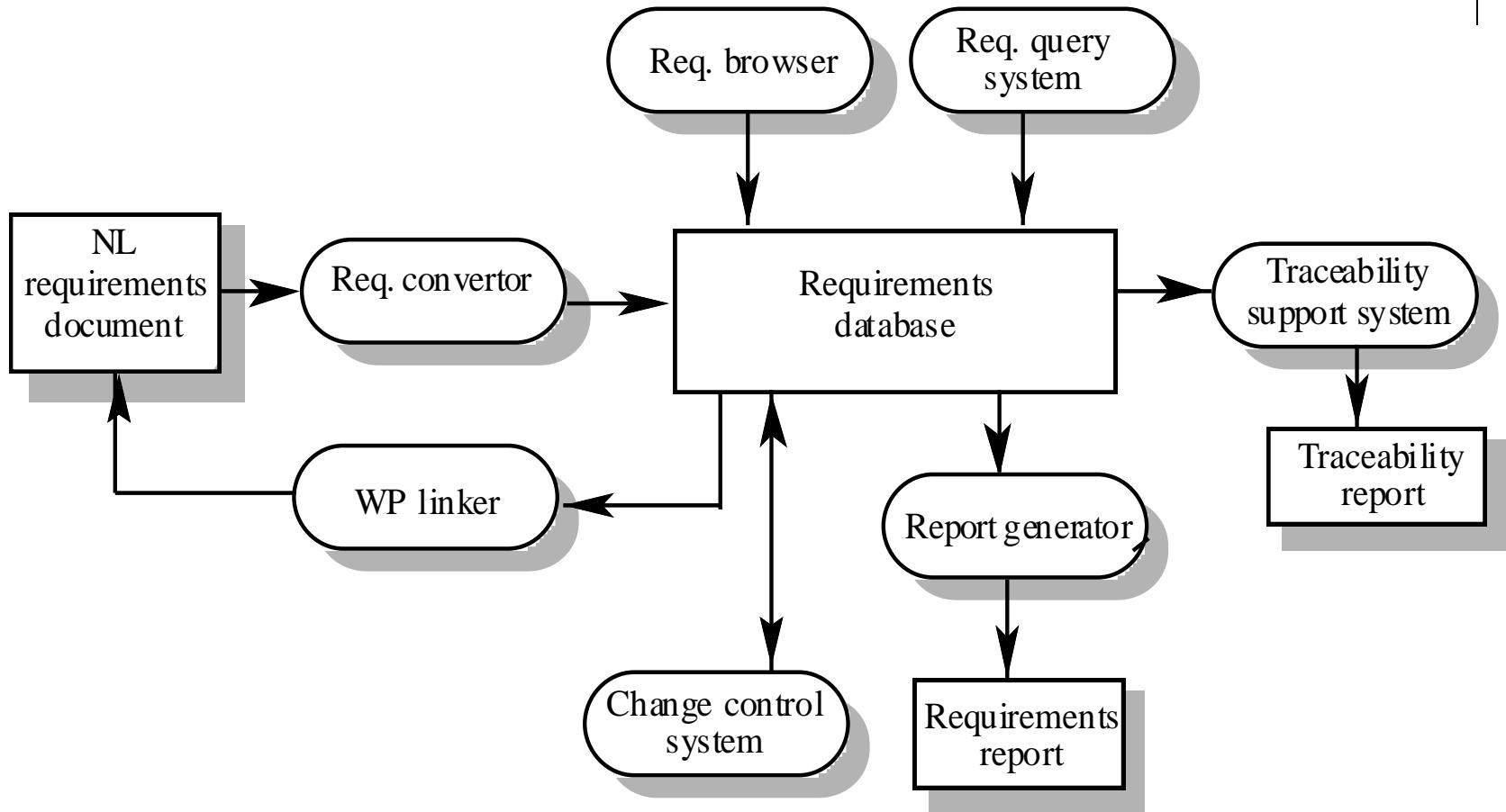
- DOORS, modeling by RML; Requisite Pro,...

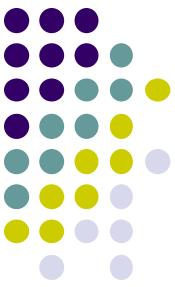
Няма общи стандарти и унификация.

- Използването на CASE за извлечане на изискванията е много ограничено. *Защо? Примери?*

Пример:

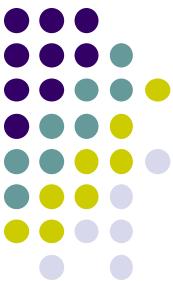
A requirements management system - 1





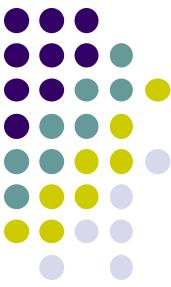
Requirements management system tool - 2

- Requirements browser
- Requirements query system
- Traceability support system
- Report generator
- Requirements converter and word processor linker
- Change control system



Подобрения на процеса на ИИ

- Модифициране на процесите на ИИ (след 1980 г.)
- Цели
 - подобряване на качеството на изискванията
 - намаляване на времето за изработка на спецификацията (Schedule reduction) с цел ускоряване на процеса
 - намаляване на използваните ресурси
 - + специфични подобрения в отделните организации



Планиране на процесите за подобрение на ИИ

Пример: completely rethinking or Kaizen? CASE bad experience.

Затова се изисква прагматичен подход, минимизиращ риска:

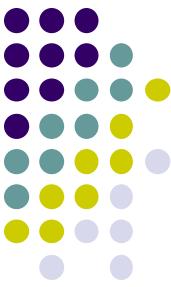
- Какви са *проблемите* с текущите процеси?
- Какво се *цели* да се подобри? *Реалистичност*
- **Как** трябва да се *въведат* подобренията, за да може да се постигнат тези цели?
- Как могат въведенията / подобренията да бъдат *контролирани и управлявани*?



Проблеми на процесите на ИИ

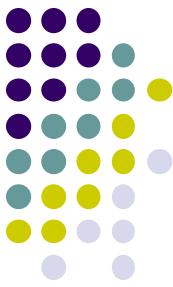
- Липса на участие от заинтересовани страни
- Бизнес нужди, които не са разгледани - ИИ е разгледан повече като техническа дейност, а не като бизнес процес и цел
- Липса на (техники за) *управление на изискванията* като основен (под)процес на ИИ
- Липса на дефинирани отговорности на хората, включени в ИИ
- Проблеми с комуникацията

В резултат: удължени графици и лошо качество на документа на изискванията.



Нивото на зрялост на компанията

- Нивото на зрялост на компанията е важен фактор за подобрение на процесите на ИИ
- CMM (Capability Maturity Model, US Department of Defense's Software Engineering Institute, 1988-1995)
 1. **Initial level** – undisciplined process
 2. **Repeatable level** – basic cost and schedule management procedures
 3. **Defined level** – both management and engineering activities are documented, standardized and integrated
 4. **Managed level** (оценка на качеството на процесите и продукта, чрез което се управлява процеса на разработката)
 5. **Optimizing level** – continuous process improvement strategy



Зрялост на процеса на ИИ

- Зрялост на процеса на ИИ – начин по който организацията дефинира ИИ чрез добри практики - подходящи методи, техники; стандарти; tools support
- Модел с 3 нива (1997, Sommerville and Sawyer)
 - **Initial level** – do not have defined RE process
 - **Repeatable level** – have defined standards for requirements description, policies for req. management
 - **Defined level** - defined RE process model based on good practices; improvement programme

Таксономия на изискванията

Лекция 3

Основни теми

- **Таксономия на изискванията**
- **Класификация на нефункционалните изисквания**
- **Изисквания за критичните системи**

Видове изисквания според нивото на детайлност на описанието

Sommerville, 2000

- **Изисквания на клиента и бизнес изисквания.** Somerville ги дефинира като потребителски изисквания (*User requirements*).
- **Системни изисквания** (*System requirements or functional specification*)
- **Спецификация** на софтуерните изисквания (*Software design specification*)

» За кого са предназначени и как са оформени документите на отделните видове изисквания?

Бизнес изисквания 1

- Бизнес изискванията представлят целите на системата от *високо ниво* на организацията или клиента, който е поръчал системата.
- Те дефинират *визията* и *обхватъ* на системата.

Karl E. Wiegers, 2003

Бизнес изисквания 2

- Бизнес изискванията *описват защо* организацията *реализира* системата и *целите*, които организацията се стреми да постигне.
- Бизнес изискванията обикновено се формулират от:
финансовия спонсор на проекта или
клиентите или
мениджърът на крайните потребители или
отдела по маркетинг или
човека, който има *виждане/идея за продукта*

Бизнес изисквания 3

- Необходимо е да се разграничават **два съществено различни типа:**
 - 1) *Същински бизнес изисквания*
 - 2) *Изисквания на продукта*

1) Същински бизнес изисквания – 1

- Те представят **резултати (deliverables)** - стоки или услуги от разработката на системата, които ще се предоставят след завършването на системата.
- Материални и нематериални резултати, които *създават стойност, като обслужват бизнес целите* чрез:
 - разрешаване на проблеми
 - използване на възможности и
 - посрещане/отговаряне на предизвикателства.
- Обикновено има **много възможни начини за тяхното изпълнение**

1) Същински бизнес изисквания – 2

- Същинските бизнес изисквания са РЕАЛНИТЕ изисквания, които доставят/принасят *стойност*, когато бъдат изпълнени
- Същинските бизнес изисквания са *концептуални* и съществуват в бизнес средата, затова трябва да бъдат открити.
- Бизнес изискванията се разглеждат от гледна точка на бизнеса или на потребителя и се изразяват на езика на бизнеса или на потребителя.

1) Същински изисквания: потребителски изисквания

- Потребителските изисквания описват *целите на потребителите* или *задачите*, които потребители трябва да могат да извършват с продукта
- Подходящи *начини за представяне* на потребителските изисквания са *потребителски случаи, описания на сценарии и таблици със събития и отговори.*
- Потребителските случаи описват, какво ще може да извърши потребителят със системата.

2) Бизнес изисквания: изисквания към продукта

- Изискванията към продукта се определят от гледна точка на продукта, определен от заинтересованите лица. Това ***вероятно е един от възможните начини за изпълнението*** на бизнес изискванията.
- Изискванията към продукта определят начин на *проектиране*, който обаче не е ограничен до техническите детайли.

Зашо?

- Те доставят стойност само ако *удовлетворяват* СЪЩИНСКИТЕ бизнес изисквания.

Системни изисквания – по-детайлно описание на изискванията

- *Детайлно описание на услугите и ограниченията, което се характеризира с пълнота и консистентност*
- Обикновено *съдържа структурни модели;*
- Описва изисквания за продукта като система т.е. *описва съществуващите зависимости и връзки.*
- Основа за *сключване на договор*

Пример

- Потребителско (бизнес) изискване:
 1. Софтуерът трябва *да осигурява представянето и достъпа до външни файлове, създадени от други софтуерни инструменти.*
- Системни изисквания:
 - 1.1 На потребителя трябва да бъде дадена възможност да *дефинира вида на външните файлове.*
 - 1.2 Всеки тип външен файл може да *има асоцииран инструмент*, който да бъде приложен към него.
 - 1.3 Всеки тип външен файл може да бъде *представен със специфична икона* на дисплея на потребителя.
 - 1.4 Системата трябва да предостави възможности на потребителя *да дефинира* представяната икона.
 - 1.5 Когато потребителят избере икона, представляща външен файл, *ефектът на този избор е да приложи съответния инструмент* към външния файл, представен с избраната икона.

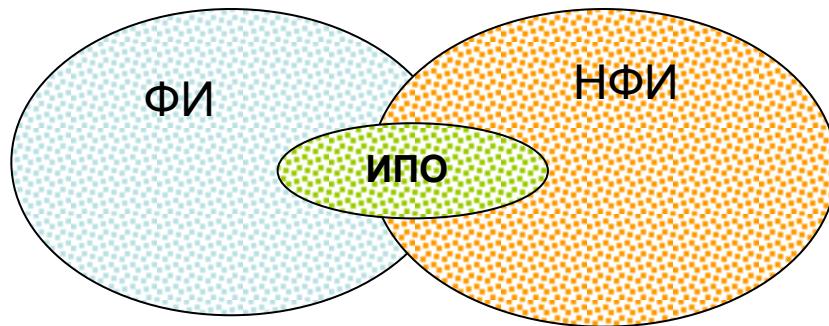
Класификация на софтуерните изисквания – хоризонтална класификация

- Изискванията могат да бъдат (Sommerville, 2001):
 - A) Функционални изисквания (ФИ)**
 - Б) Нефункционални изисквания (НФИ)**
 - В) Изисквания, произтичащи от приложната област (ИПО)**



Зависимост между софтуерните изисквания

Според Sommerville, 2001



Според Klaus Pohl, 2010 г. и IEEE 830-1998, както и други автори, класификацията на изискванията е:

- А) Функционални
- Б) Нефункционални = качествени + ограничения

А. Функционални изисквания 1

- Функционалните изисквания описват какви **функционалности/услуги** трябва да предоставя системата на клиенти и/или на други системи.
- В някои случаи ФИ указват какво *не* трябва да прави системата.
- **(отново за) Вертикално разделяне:**
 - Когато изискванията се изразяват като *потребителски изисквания*, те обикновено се описват по по-абстрактен начин.
 - *Системните функционални изисквания обаче, описват в детайли системната функция: нейния вход и изход, изключения и т. н.*

Sommerville, 2007

Функционални изисквания 2

- Едно функционално изискване описва необходимото поведение от гледна точка на *нужните дейности: как системата трябва да реагира на определени входни данни, какво трябва да е поведението на системата при определени ситуации и каква изходна информация трябва да предоставя за състоянието на всеки обект/компонент/единица преди и след извършването на дадено действие.*
- Функционалните изисквания дефинират *границите* на решенията за даден проблем.

Функционални изисквания 3

- Функционалните изисквания се документират чрез три допълващи се, но и частично припокриващи се *перспективи*:
 - Перспектива на **данные**
 - **Функционална** перспектива
 - Перспектива на **поведение**
- Проблеми (взаимно изключващи се) при описанието:
 - Ако е твърде общо, преобладава неяснота/двусмислие;
 - Ако е твърде специфично може да затрудни проектирането, получава се обемист документ
- Принципно спецификацията на функционалните изисквания трябва да бъде едновременно **пълна** и **консистентна**. Практически това е трудно изпълнимо...

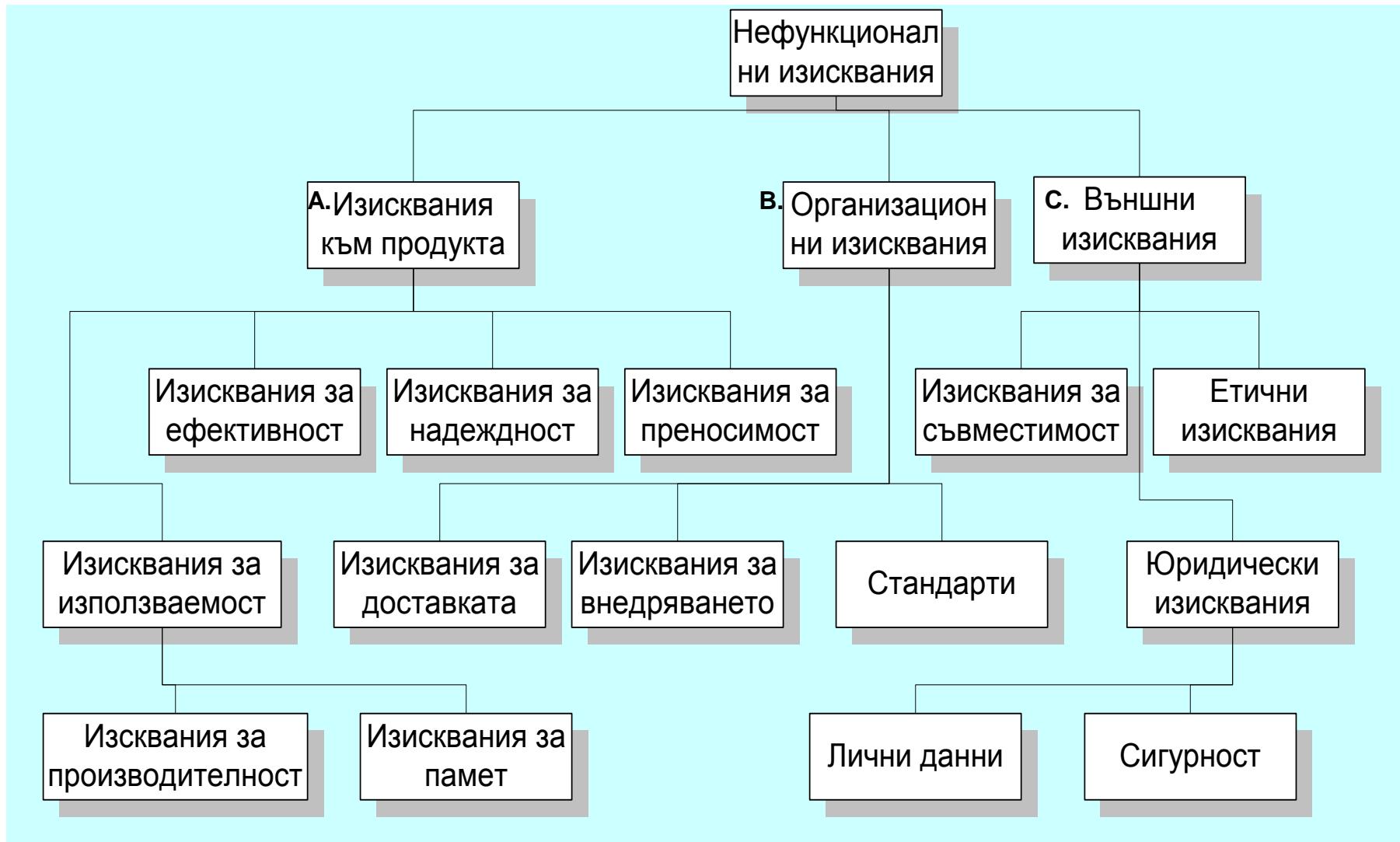
В) Нефункционални изисквания: Дефиниция

- **Нефункционални изисквания** са изисквания, които не се отнасят до функционалността на системата
- **НФИ** поставят:
 - ограничения върху софтуерния продукт и
 - върху процеса на разработката,
 - уточняват *външните ограничения*, с които трябва да е съобразен продуктът.
- **НФИ се отнасят до системата като цяло**, а не до отделна нейна функционалност.
(критичност на НФИ?)

Типове нефункционални изисквания

Класификация на НФИ

/по Sommerville/



А. НФИ: изисквания към продукта /качествени изисквания

- Изискването за качество описва *някаква качествена характеристика, която софтуерното решение (отделна услуга или система) трябва да притежава или да има като поведение.*

Примери:

- Кратко време за отговор
- Леснота на употреба
- Висока надеждност
- Ниски разходи за поддържане

Изисквания за качеството 2

- Изискването за качество *дефинира свойство на цялата* система (системния компонент или услугата).
- Типове качествени свойства
 - С основно значение за **потребителите**:
 - наличност, ефективност, гъвкавост, цялостност
 - съвместимост, надеждност, устойчивост, използваемост
 - С основно значение за **разработчиците**:
 - възможност за поддържане и обновяване, преносимост, възможност за повторно използване, възможност за лесно тестване

Примери за НФИ – изисквания към продукта

- The System service X shall have an *availability* of 999/1000 or 99%. (*reliability*)
- System Y shall process a minimum of 8 transactions per second. (*performance*)
- The executable code of System shall be limited to 512 Kbytes. (a requirement for *maximum memory size* of the system).
- The system shall be developed for PC and Mac platform. (*portability*)
- The system must encrypt the communication using RSA algorithm. (*security/сигурност*)

Качествени системни изисквания 1

■ Наличност

- наличността се отнася до *процента от времето*, през което системата е налична за употреба и работи изцяло

■ Ефективност

- Ефективността е мярка за това *колко добре система оползотворява/използва хардуерните ресурси като процесорно време, памет или честотна лента*

■ Гъвкавост

- Гъвкавостта показва *колко усилия са необходими*, за да се разшири системата с нови възможности

■ Цялостност

- Цялостността означава не само работата като цяло на системата, но колко добре е защитена тя срещу *непозволен достъп, нарушаване на поверителността на данните, загуба на информация и заразяване с вреден софтуер*

Пример: Цялостността осигурява местна поддръжка за инструменти за моделиране и симулации и съчетаване с инструменти за моделиране като [Simulink](#) и [MATLAB](#), предоставящи възможност за ~~23~~100 проследяване между всички активи/обекти/

Качествени системни изисквания 2

■ Съвместимост

- Съвместимостта показва колко лесно системата може да *обменя данни или услуги* с други системи.

■ Надеждност

- Според стандарта IEEE 1998: *Способността на една система или компонент да извършива своите функции при указаните условия за определен период от време.*

■ Устойчивост/Здравина

- Устойчивостта е способността на дадена система или компонент да продължи да функционира, когато трябва *да се справи* с определени входни данни, *нарушения* в свързаните системи или компоненти, или *непредвидени* условия на работа

Качествени системни изисквания 3

■ Използваемост

- Използваемостта измерва усилията, които се изискват от потребителите, за да подготвят входните данни, да работят със системата и да интерпретират върнатите данни от системата.

■ Възможност за поддържане и обновяване

- Възможността за поддържане и обновяване показва *доколко е лесно да се поправи недостатък* или да се направи промяна в системата.

■ Преносимост

- Преносимостта се отнася до усилията за извършване на *миграция* на система или компонент от една операционна среда в друга.

Качествени системни изисквания 4

- **Възможност за повторно използване**
 - Възможността за повторно използване показва *доколко е възможно един компонент да се използва* в други системи, различни от тази, за която е разработен първоначално.
- **Възможност за тестване**
 - Възможността за тестване показва степента на леснотата, с която софтуерните компоненти или цялата система *могат да се тестват за откриване на дефекти/недостатъци*.

- *Съществува зависимост - често е противоречие между различните качествени характеристики.*
- There is no a clear distinction between functional and non-functional requirements. Whether or not a requirement is expressed as a functional or a non-functional requirement may depend:
 - on the **level of detail** to be included in the requirements document
 - the **degree of trust** which exists between a system customer and a system developer.

Пример: Класифицирайте следното изискване. Формулирайте го по-детайлно.

- R: The system shall ensure that data is protected from unauthorised access.
Conventionally, this would be considered as a non-functional requirement.

In slightly more detail as follows:

- R1: The system shall include a user authorisation procedure where users must identify themselves using a login name and password. Only users who are authorised in this way may access the system data.

In this form, the requirement looks rather more like a functional requirement.

Примери за измерими метрики

| Свойство | Метрика |
|------------------|---|
| Производителност | обработени транзакции в секунда; време за отговор на потребителските входни данни |
| Надеждност | честота на настъпването на провал; средно време за провал |
| Наличност | вероятност за провал при поискване на системните услуги |
| Размер | килобайтове |
| Използвае - мост | времето, необходимо за изучаване на 80% от възможностите; броя на грешките, направени от потребителите за определен период от време |
| Преносимост | брой на целевите системи |

Въпроси за изискванията за качество

- **Производителност**
 - Има ли ограничения върху *скоростта за изпълнение, времето за отговор?*
 - Какви мерки за *ефективност* са подходящи за използването на ресурсите и времето за отговор?
 - Какво количество данни ще преминават през системата?
 - Колко често ще бъдат получавани или изпращани данни?

Въпроси за изискванията за качество – 2

- **Използваемост**
 - Какво обучение ще се изисква за всеки тип потребител?
 - Колко лесно трябва да бъде за даден потребител да разбира и да използва системата?
 - Колко трудно трябва да бъде за един потребител да използва системата неправилно?

Въпроси за изискванията за качество – 3

■ Сигурност

- Трябва ли да се контролира достъпът до системата или информацията?
- Трябва ли данните за всеки потребител да бъдат изолирани от данните за друг потребител?
- Трябва ли потребителските програми да бъдат изолирани от други програми или от операционната система?
- Трябва ли да се вземат предпазни мерки срещу кражба и нарушения?

Въпроси за изискванията за качество – 4

■ Надеждност и наличност

- Трябва ли системата да открива и изолира грешки/повреди?
- Какво е определеното *средно време* между повредите?
- Има ли максимално позволено време за рестартиране на системата след повреда?
- Колко често ще се извършва архивиране (backup) на системата?
- Трябва ли архивните копия да се съхраняват на друго място?
- Трябва ли да се вземат предпазни мерки срещу повреда от вода или огън?

Въпроси за изискванията за качество – 5

■ Възможност за поддържане и обновяване

- Поддържането и обновяването на системата ще включва ли подобряване на системата или ще се грижи само за поправяне на грешки?
- Кога и по какви начини може да бъде променена системата в бъдеще?
- Колко лесно трябва да бъде добавянето на нови възможности за системата?
- Колко лесно трябва да бъде преместването на системата от една платформа на друга?

Въпроси за изискванията за качество – 6

■ Прецизност и точност

- Колко *точни* трябва да бъдат изчисленията с данните?
- До каква *степен на точност* трябва да се извършват изчисленията?

В. НФИ: Организационни изисквания (изисквания на процеса на разработка)

- **Ограничението** представлява организационно или технологично изискване, което определя начина, по който ще бъде разработена системата.

Robertson and Robertson, 2006

- Изисквания, които са **ограничения върху процеса на разработката**, резултат на съществуващи политики и процедури в организациите на разработчиците и на клиентите.

Ограничения по имплементацията – технологични ограничения

- **Видове технологични ограничения.** Ограниченията се отразяват върху възможните начини за разработване на дизайна и изграждането на продукта (език за програмиране, метод за проектиране, технология на разработка; CASE инструменти ...)
 - **Ограничение върху дизайна**
 - *решение за дизайна*
 - **Ограничение на процеса**
 - Ограниченията върху процесите са ограничение върху *техниките* или *ресурсите*, които могат да се използват за изграждането на системата

Видове организационни изисквания

В.1. Методи и стандарти, които трябва да се следват

Пример: Процесът на разработка трябва да е ясно дефиниран и трябва да бъде в съответствие със стандарт ISO 9000.

В.2. Изисквания за имплементация – език за програмиране, метод за проектиране, технология на разработка;

CASE инструменти, които трябва да бъдат използвани;

Пример R: Потребителският интерфейс на LIBSYS ще се осъществи като прост
HTML без фрейми и Java аплети.

В.3. Изисквания по доставянето – срокове и отчети, които трябва да бъдат предоставени

B.3. Изисквания по доставянето - пример

■ Време за доставка/Разходи

- Има ли определено *разписание* за работата по разработването на системата?
- Има ли *ограничение* за паричната сума, която трябва да се използва за разработката или за хардуер и софтуер?

C) НФИ: Външни изисквания

- Външните изисквания са резултат на *фактори, вънни за системата и за процеса* на създаването ѝ.
- Те могат да са наложени както върху продукта, така и върху процеса на разработване.
- **Външни фактори:**
 - Необходимостта системата да работи с *други системи и организации (Interoperability requirements)*
 - *Правила и закони* (за здравеопазване, сигурност и защита на данните, *ethical requirements*) – *legislative requirements*
 - Съобразяване с икономически и физически закони (*economic constrain*)

- Бизнес правилата често включват **корпоративни политики, правителствени наредби, индустриални стандарти, счетоводни практики и изчислителни алгоритми**

Те задават ограничения върху това, кой може да извършва определени потребителски случаи или указват, че системата трябва да съдържа определена функционалност, така че да бъдат спазени съответните правила

(Примери: библиотечна система; GDPR)

- External requirements rarely have the form “the system shall...” or ‘the system shall not...’. Rather, they are descriptions of the system’s environment which **must be** taken into account.

Външни изисквания: примери

Пример I. *Medical data system*: The requirement described comes from the need for the system to conform to data protection legislation

R: The organisation's data protection *officer must certify* that all data is maintained according to data protection legislation before the system is put into operation.

Пример II.

R: Due to current condition defined by the insurance company, only security technician is allowed to deactivate the control function of the system.

(Въпроси за външни) ограничения върху дизайна

- **Физическа среда**
 - Къде трябва да се разположи оборудването/съоръженията?
(Пр. Chairlift access control system)
 - Дали местоположението е *едно* или са *няколко*?
(Пр. Комуникационните системи)
 - Има ли ограничения, които се налагат от *околната среда*, като например температура, влажност или магнитни смущения?
(Пр. Chairlift access control system)
 - Има ли някакви ограничения върху *размера* на системата?
 - Има ли някакви ограничения върху *електрозахранването, отоплението или климатизацията*?

(Въпроси за) ограничения върху дизайна – 2

- **Интерфейси** (*пример: IoT проекти*)
 - Дали *входните данни* идват от една или повече други системи?
 - Дали *изходните данни* се изпращат към една или повече други системи?
 - Има ли установлен *начин*, по който трябва да бъдат *форматирани* входните/изходните данни?
 - Има ли определен *носител* за данните, който трябва да се използва?
- **Потребители**
 - Кой ще използва системата?
 - Ще има ли *различни* типове потребители?
 - Какво е *нивото на уменията* на всеки потребител?

Проблеми, произтичащи от нефункционалните изисквания - обобщение:

- Някои ограничения (време за отговор и др.) са резултат на определени решения в етапа на проектирането и **не могат да бъдат точно формулирани на етапа на ИИ.**
- Изисквания, свързани с **човешкия фактор** са субективни и могат да бъдат идентифицирани чрез емпирични анализи.
- **ФИ и НФИ са тясно свързани** и не могат да бъдат разделени
- НФИ са взаимнозависими.
- Няма правила, с които да дефинираме „оптимални“ нефункционални изисквания.

Изисквания, произтичащи от приложната област

- Дефинират се изисквания, резултат от областта на приложение, а не от изискванията на клиента
 - Информацията за *предметната* област на приложението.
 - Основни *природни* закони.
- Понякога тези изисквания отразяват основите на изгражданата система
 - могат да бъдат функционални и нефункционални.

Пример:

1. Train protection system
2. Системата трябва да има стандартен потребителски интерфейс за всички бази данни, основан на Z39.50 (функциј. изискване)

Изисквания, произтичащи от приложната област: примери

Пример: *Train protection system:*

The time required to bring the train to a complete halt is computed using the following function:

The deceleration of the train shall be taken as:

$$g_{\text{train}} = g_{\text{control}} + g_{\text{gradient}}$$

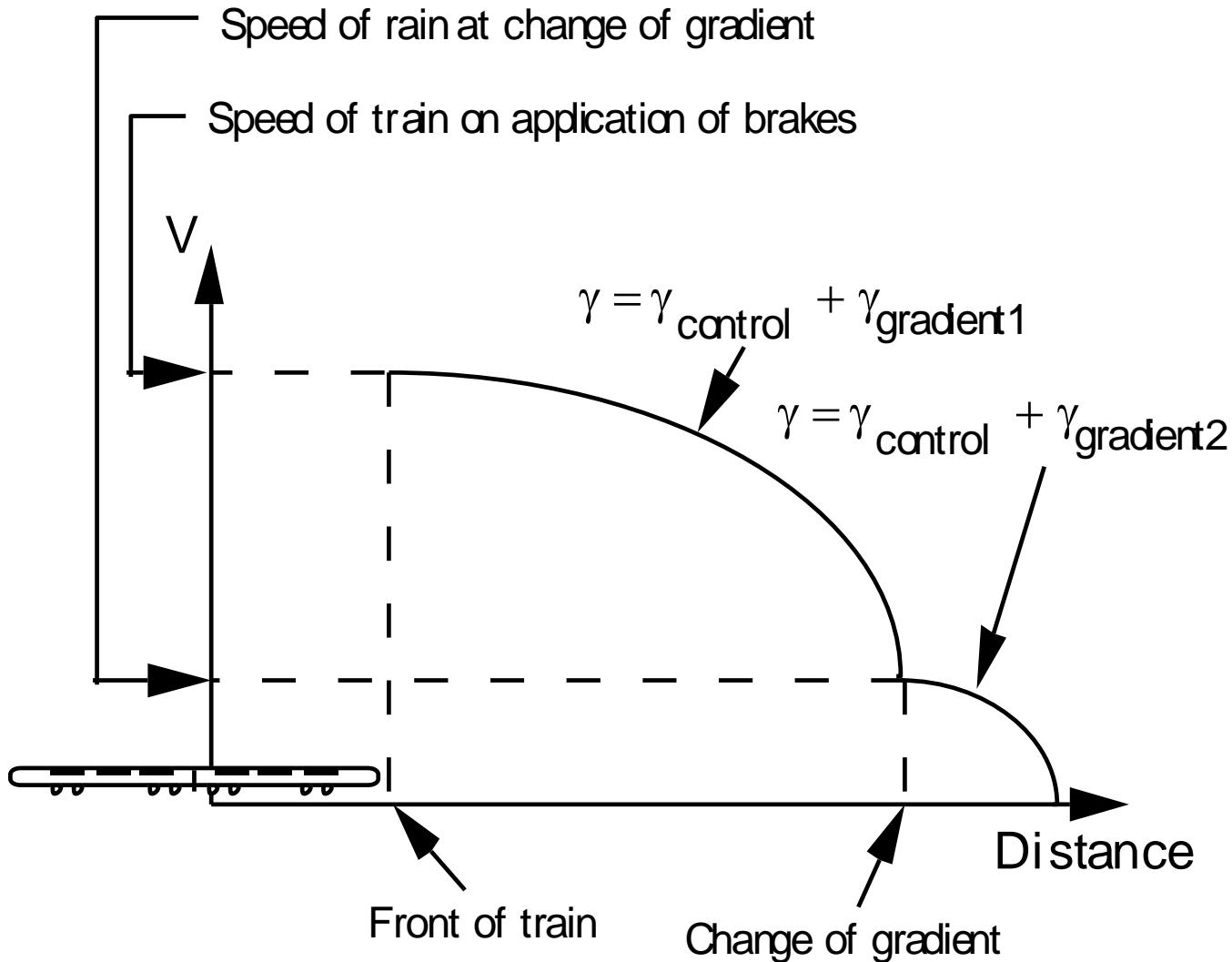
where:

$$g_{\text{gradient}} = 9.81 \text{ ms}^2 * \alpha$$

where α are known for **different types of train.**

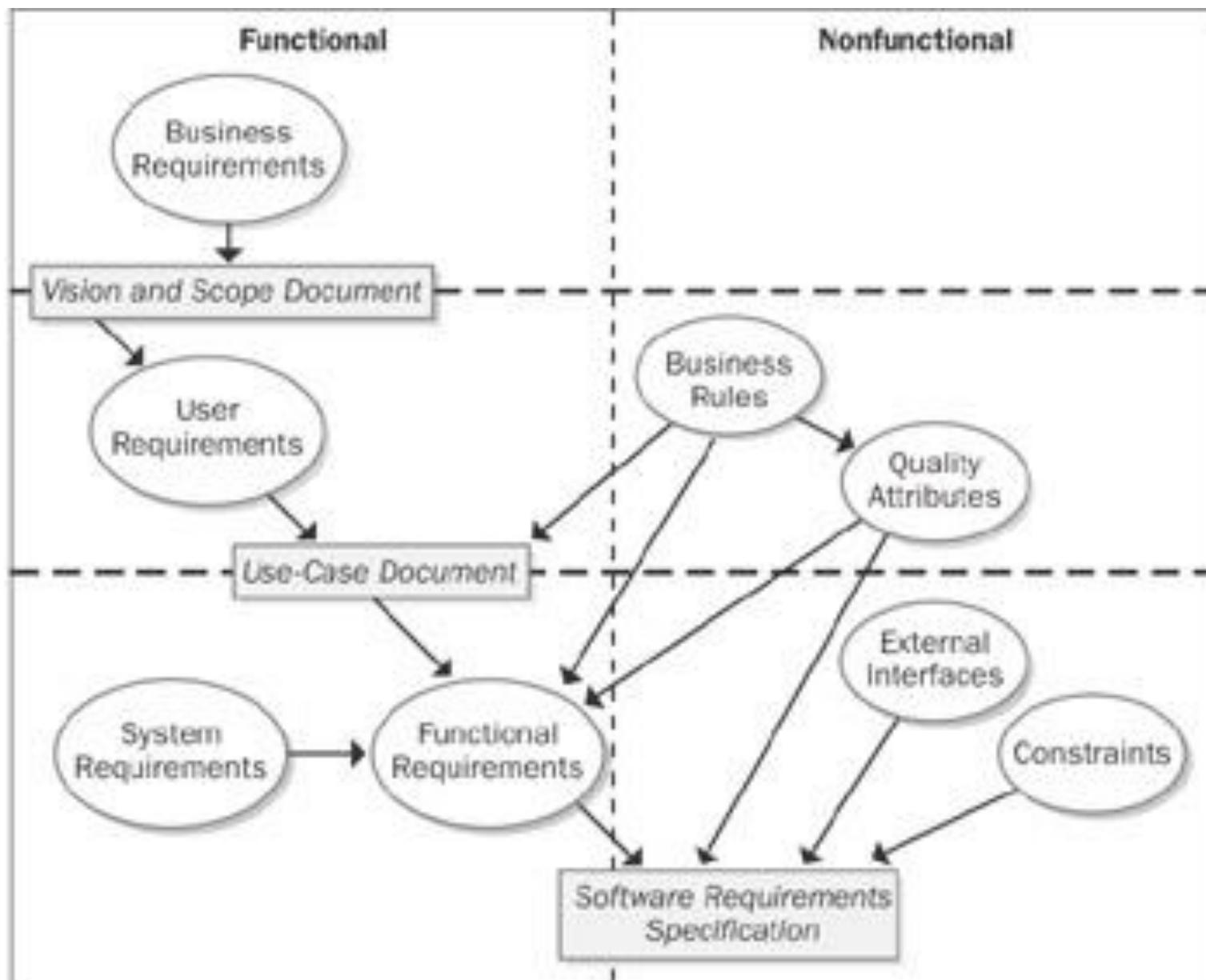
g_{control} - this value being parameterised in order to remain adjustable.

Външни изисквания: примери (продължение)



Нива на изискванията: Връзки между информацията за различните видове изисквания

Karl E. Wiegers, 2003



Класификация на нефункционални изисквания според различни автори

- Sommerville класифицира нефункционалните изисквания по следния начин:
 - за продукта
 - за процеса
 - външни
- Качства, които софтуерът трябва да притежава – Boehm [1976]
- Изисквания, които не се отнасят до поведението – Davis [1992]
- Функционални и качествени изисквания (Klaus Pohl)

Видове НФИ

The ‘IEEE-Std 830 - 1993’ lists 13 non-functional requirements to be included in SRS.

1. Performance requirements
2. Interface requirements
3. Operational requirements
4. Resource requirements
5. Verification requirements
6. Acceptance requirements
7. Documentation requirements
8. Security requirements
9. Portability requirements
10. Quality requirements
11. Reliability requirements
12. Maintainability requirements
13. Safety requirements

Изисквания за критични системи

- **Видове критични системи**
 - Критични бизнес системи
 - Повредата/провалът на системата нанася икономически щети на бизнеса
 - Критични системи за изпълнението на важни задачи/мисии
 - Провалът на системата означава, че не може да се изпълни определена мисия/задача
 - Критични системи за безопасността
 - Провалът на системата застрашава човешкия живот или нанася значителна вреда на околната среда

Нефункционални изисквания за критичните системи

- Надеждност (reliability)
- Експлоатационни качества (performance)
- Сигурност (security)
- Използваемост (usability)
- Безопасност (safety)

Надеждност

- Изискванията за надеждност са ограничения върху поведението на системата по време на работа
 - **Наличност** - Налични ли са услугите на системата, когато са необходими на крайните потребители?
 - **Честота на случаите на провал**
 - Колко често системата не може да предостави услугите, очаквани от крайните потребители?
 - **Количествени измервания** за надеждност

Експлоатационни качества (performance)

- Изискванията за производителност определят *скоростта* на работа на системата
 - **Изисквания за отговор**, които указват допустимото време за отговор на системата при входни данни от крайните потребители
 - **Изисквания за скоростта** на предаване/пропускателната способност, които определят какво количество данни трябва да се обработи за определено време
 - **Изисквания за синхронизация**, които определят
 - колко бързо системата трябва да *събира* входните данни от сензорите преди те да бъдат заменени от следващите входни данни
 - колко бързо трябва да се *извеждат* изходните данни, за да бъдат обработени от други системи

Сигурност

- Изискванията за сигурност са включени в системата, за да **не се допуска неразрешен достъп** до системата и нейните данни и за да **се осигури цялостността** на системата при случайни или злонамерени повреди
- *Примери*
 - *Разрешенията за достъп* до системните данни могат да се променят само от администратора на данните за системата
 - Всички системни данни трябва да се *архивират* на всеки 24 часа
 - Външната комуникация между системния сървър за данни и клиентите трябва да бъде *криптирана*
- Сигурността е важно условие за *безопасност*

Използваемост

- Изискванията за използваемост имат за цел да специфицират потребителския интерфейс и взаимодействието между крайните потребители и системата
- **Измерими атрибути** на използваемостта
 - Изисквания за вписване
 - Колко години на опит има натрупан за работа с приложенията/с-мата
 - Изисквания за обучение
 - Необходимото време за изучаване на възможностите на системата
 - Изисквания за справяне с грешки
 - Указва се честотата на грешките от страна на крайните потребители на системата
 - Привлекателност (неизмерим атрибут)
 - “приятност” при използването на системата

Безопасност

- Изискванията за безопасност са “няма да”-формулировка на изисквания, които изключват опасни ситуации от възможното пространство с решения за системата
- *Примери*
 - Системата няма да разрешава извършването на работа, ако предпазното устройство за работа не е в готовност/изправност
 - Системата няма да работи, ако външната температура е под 4 градуса по Целзий

Инженеринг на изискванията

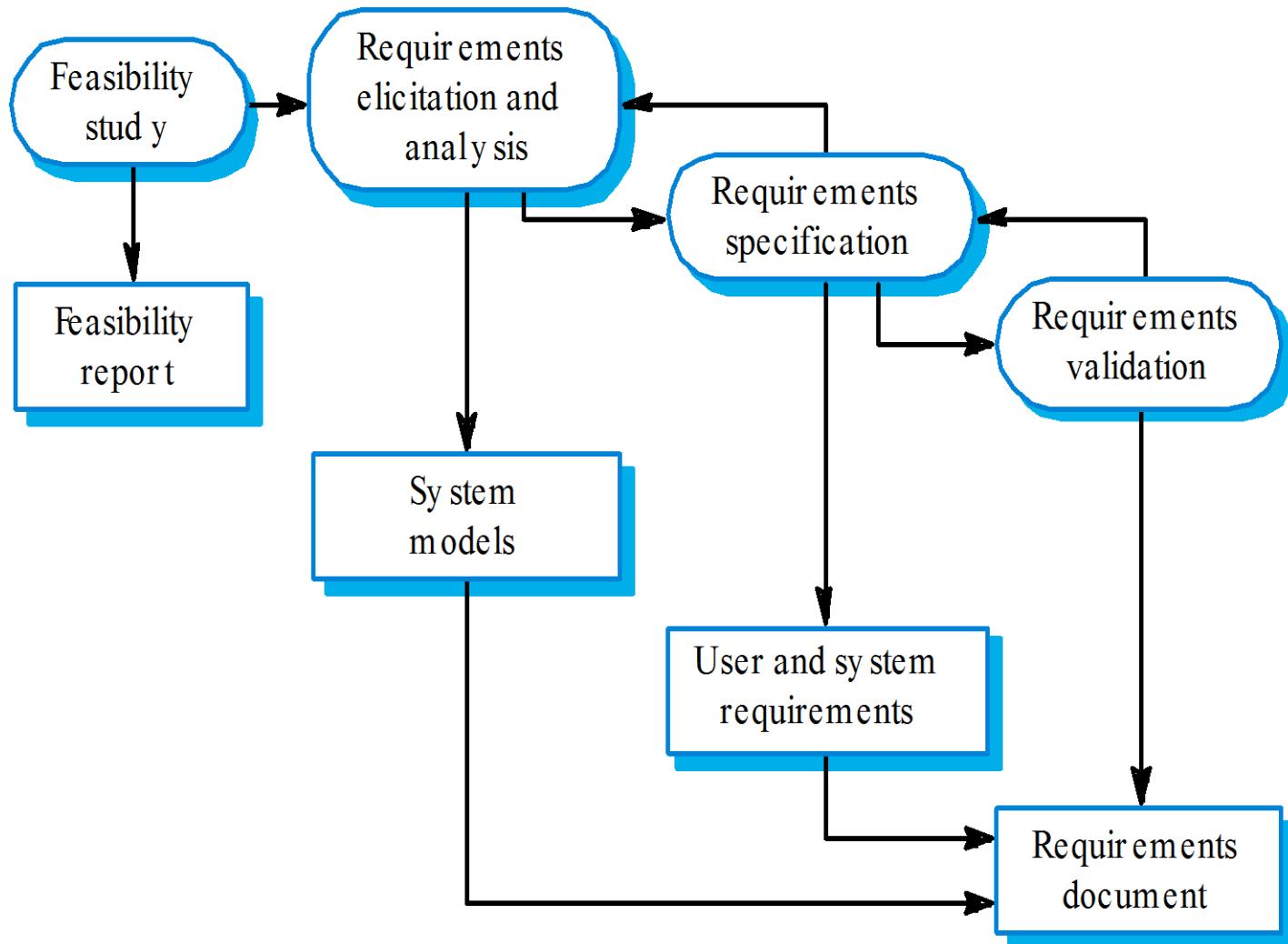
Извличане на изискванията

Лекция 4

Съдържание

- **Изходна информация. Граница и контекст на системата**
- **Цели и процес на извлечането на изискванията**
- **Източници за идентифициране на изискванията**
- **Техники за извлечение на изискванията**
 - Сценарии и Потребителски случаи – основни принципи и понятия
 - ...
- **Проблеми при идентифициране на изискванията**

Процес на инженеринг на изискванията



Идентифициране на изискванията – начална информация 1

Начало е:

**Съществува “проблем”, който трябва да се реши,
зашпото има:**

- неудовлетвореност от текущото състояние
- нова възможност за бизнеса
- възможност за спестяване на разходи, време, ресурси и

др.

Предпроектни проучвания /или проучвания за осъществимост/ (Feasibility studies)

Входна информация: Общо описание на системата; как тя ще бъде използвана

Изходна информация: Доклад-становище дали системата да бъде разработвана:

- *принос* за организацията, за която е предназначена
- *реалистичност* - може ли да бъде разработена с използване на наличните ресурси и ограничения
- възможност за *интегриране* с други системи.

Пример: приложение за подпомагане на възрастни хора (Covid 19), складова система

Извличане на изискванията – общи принципи (Requirements elicitation)

Дефиниция: Процес на търсене и откриване (идентифициране), консолидиране (обобщаване) на софтуерните изисквания от наличните източници процес на извлечение на изискванията.

- *Включва работа с клиентите*, за да се:
 - проучи приложната област
 - внимателно анализиратисканията на заинтересованите лица и ограниченията като бизнес процеси, законодателство и др.
- *Участници в процеса*: крайни потребители, мениджъри, инженери, които участват в поддръжката, експерти в областта, търговски съюзи, т.е. всички заинтересовани страни.

Действия по идентифицирането на изискванията – четири измерения

- Разбиране на приложната област**

Знанията за общата област, в която се прилага системата;

Източници на информация

- Разбиране на проблема**

Детайлите за специфичния клиентски проблем,

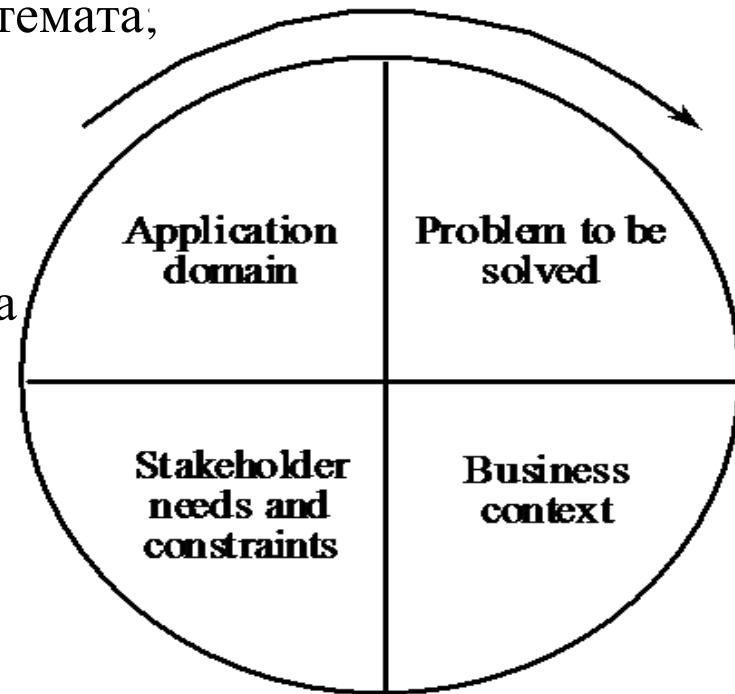
Знанията могат да се предоставят от различни хора

- Разбиране на бизнеса**

Как си взаимодействат системите и
как спомагат за общите бизнес цели.

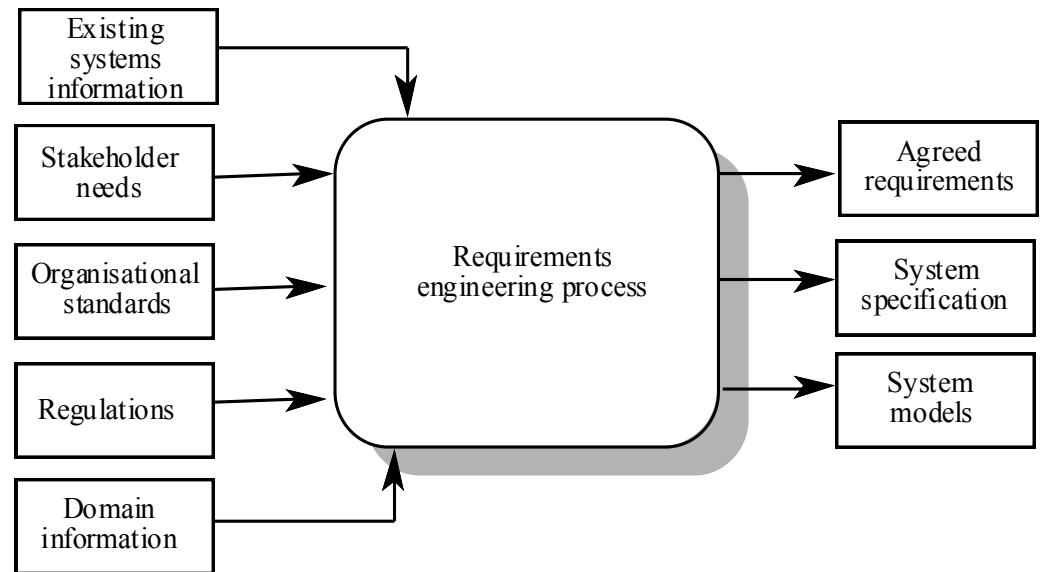
- Разбиране на нуждите и ограниченията на
заинтересованите лица**

Различните заинтересовани лица имат различни изисквания
и могат да ги изразят по различни начини.



Източници на информация за извличане (идентифициране) на изискванията

- заинтересованите лица
- информация за областта
- съществуващи системи
- документи (за организационни стандарти, регулатии, др.)
- наблюдения



Изходна информация за идентифициране на изискванията

Заинтересовани страни/лица

- Определяне на носителите на (бизнес) изискванията (problem owners)

Граници

- Определяне на обхвата на проблема

Цели

- Определяне на критериите за успех
- Използване на конкретни примери за идентифициране проблема

Идентифициране на изискванията – начална информация

• Събиране на достатъчно информация за дефиниране на “проблема”

На кого е проблемът? (идентифициране на заинтересованите страни
Кой проблем трябва да се разреши? (определение на границите на
проблема)

Къде е проблемът? (изясняване на контекста/проблемната област)
Зашо е необходимо да се разреши? (определение на целите на
заинтересованите страни)

Как се проявява проблемът? (събиране на сценарии)
Кога трябва да се реши? (определение на ограниченията върху разрабо
Какво може да попречи на разрешаването? (определение на
осъществимостта и риска)

• Да станеш експерт в проблемната област (или просто съвети ☺)

Научете се, как да се ориентирате бързо в нова проблемна област.
Използвайте своето първоначално незнание като извинение, за да задавате
въпроси.
Признавайте експертните знания в областта на хората, с които разговаряте.

The
journalist's
technique:
What?
Where?
Who?
Why?
When?
How?
(Which?)

Заинтересовано лице/страна

Заинтересованото лице е всеки, който по някакъв начин е засегнат или оказва влияние върху продукта/системата:

- човек
- организация
- група от хора, които имат обща заинтересованост

Зainteresовани страни

- Анализ на заинтересованите лица:
Идентифициране на ролите на заинтересованите лица.
- Примери на заинтересовани лица

Потребители

за тях са важни възможностите и функционалността на новата система

Дизайнери

искат да изградят перфектна система или да използват съществуващ код

Системни аналитици

искат да “разберат изискванията правилно”

Персонал, който отговаря за обучението и поддръжката за потребителите

искат да се уверят, че новата система е използваема и управляема

Бизнес анализатори

искат да се уверят, че “ние сме по-добри от конкуренцията”

(Технически) автори

те ще подготвят потребителски наръчници и друга документация за новата система

Мениджър на проекта

иска да завърши проекта навреме, в рамките на определения бюджет, и да бъдат постигнати всички подцели

“Клиентът”

Иска да получи най-доброто за инвестираните парични средства!

Определяне на заинтересованите лица. Прилагани техники

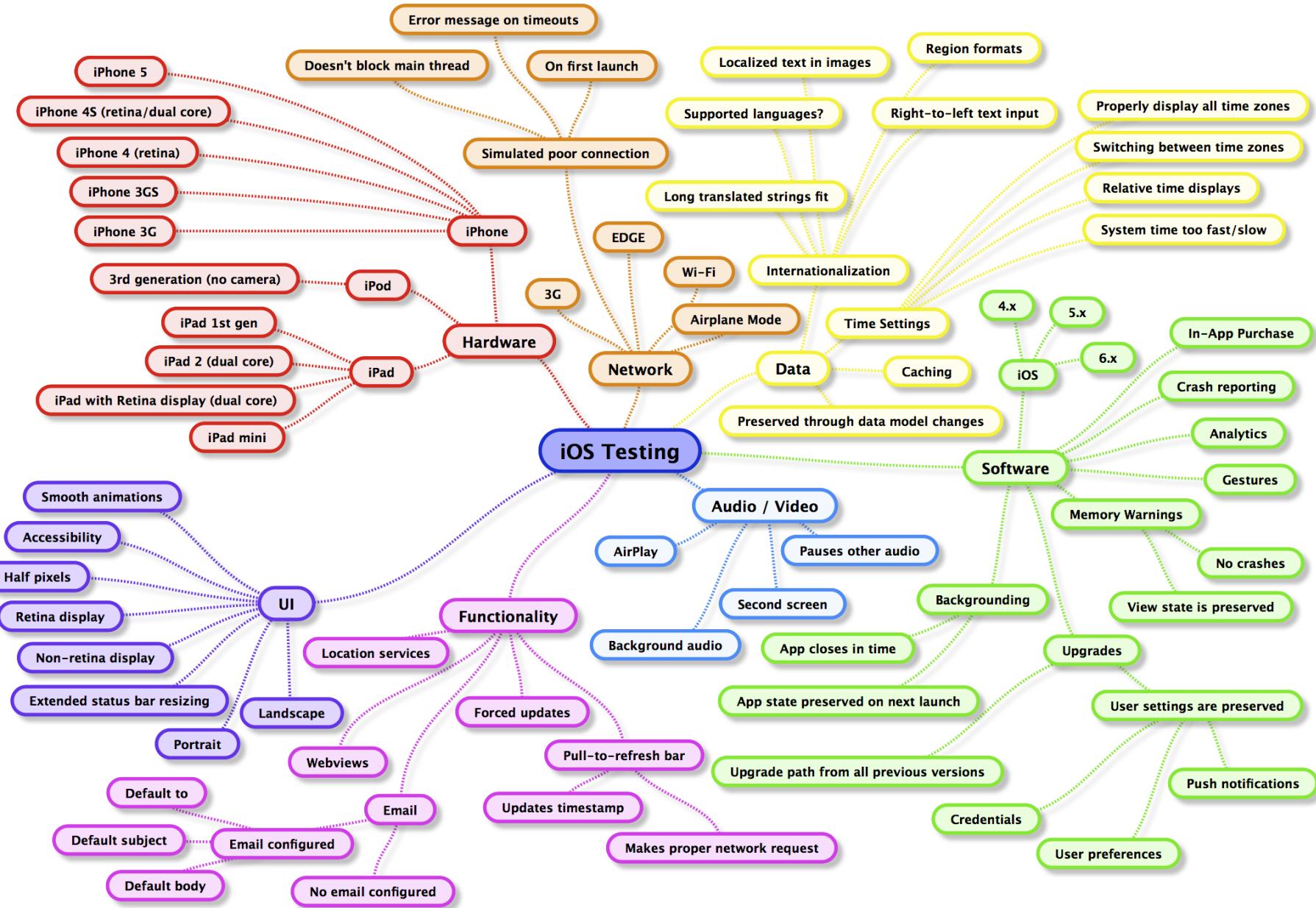
- списъци
- въпроси: **кой** знае, работи, чувства, получава полза или вреда ...
- класифициране на заинтересованите страни и
- идентифициране на конкретен човек за отделните роли
- **мозъчна атака (brainstorm)** срещи за откриването на идеи
- **mind-maps** за формиране и откриване на идеи



[Yu 1997]
[van Lamsweerde 2001]

Glinz and Wieringa 2007]

Пример: mind-map за тестване на Операционна система



Влияние на заинтересованите лица

- **Положително**
предоставят изискванията, пари, знание
- **Отрицателно**
*противопоставят се на продукта и/или
предоставят погрешни изисквания, спират ресурси,
необходими за ИИ*

Никой от източниците не трябва да се забравя!

- Пропускането на някой от източниците може да се окаже фатално при идентифицирането на изискванията
- Фактори за избора на източник:
 - знаещ/осведомен човек
 - безопасност при събиране на информацията
 - политически и/или икономически ограничения
- *Забележка:* Използване на списък и/или диаграма за целите, връзките и рационалността на всяко з.л.

Context analysis

Context analysis

Determine the system's context
and the context boundary

Identify context constraints

- Physical, legal, cultural,
environmental
- Embedding, interfaces



Photo © Universitätsklinikum Halle (Saale)

Identify assumptions about the context of your system and
make them explicit

Map real world phenomena adequately...

- ... on the required system properties and capabilities
- ... and vice-versa

Дефиниция на граница на системата

- Границата на системата отделя планираната система от заобикалящата я среда.
- Обхватът на системата се определя от границата на системата.

Продуктът (разработваната система) може да се моделира и да се променя по време на разработката.

НО !

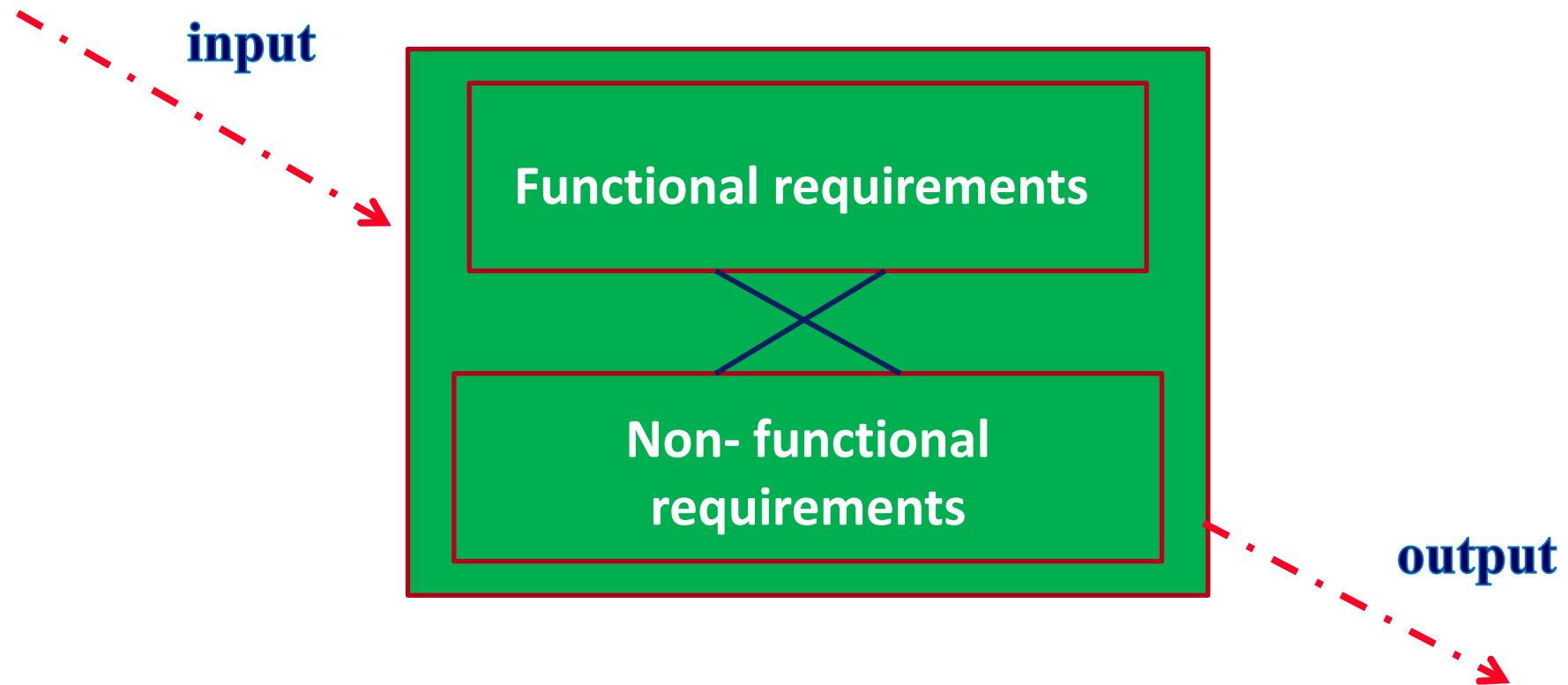
Средата *не* (не зависи от нас) може да се променя в рамките на този проект.

Дефиниция на контекст на системата

- **Границата на контекста разделя съответната част на средата на планираната система от частта, която е без значение.**
- Контекстът на системата е тази част от средата на системата, която е от значение за изясняването и дефинирането на системата.
- Частта от света извън контекста е без значение в проекта.

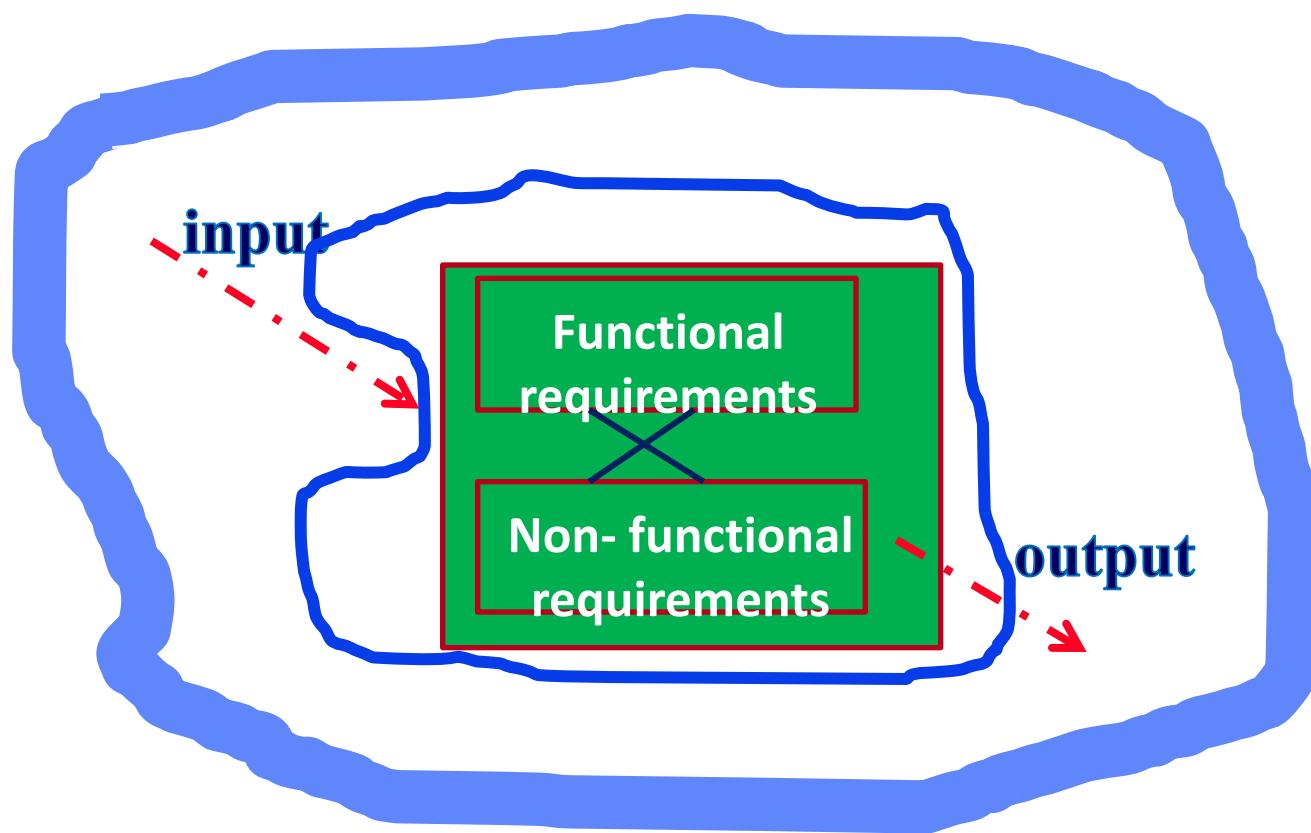
Контекст

Системата/продуктът и границата на системата са засегнати от контекста:



Определяне на границите

Границата на системата и на контекста могат да се променят по време на процеса на ИИ:



Boundary zones are fact of life.

Диаграма на контекста

Диаграмата на контекста се състои от:

- (Черна) **кутия**, представляща системата;
- **Кутии с етикети** за всеки от външните източници на информация и за
- **получателите на информация**
- Именувани интерфейси между източниците и **получателите на информация от** системата.

Инженерингът на изискванията се ръководи от контекста на системата и от източниците на изискванията:

- Хора
- Съществуващи системи
- Процеси
- Събития
- Документи

В началото са идеи, желания, нужди, решения, предложения, възможности

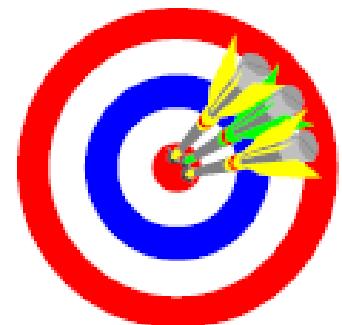
Това трябва да се превърне в: *цели и изисквания*

Goal analysis

Knowing your destination is more important than the details of the timetable.

Before eliciting detailed requirements, the general goals and vision for the system to be built must be clear

- What are the main goals?
- How do they relate to each other?
- Are there goal conflicts?



Цели

Крайната цел е намерение относно подцелите, свойствата или употребата на системата.

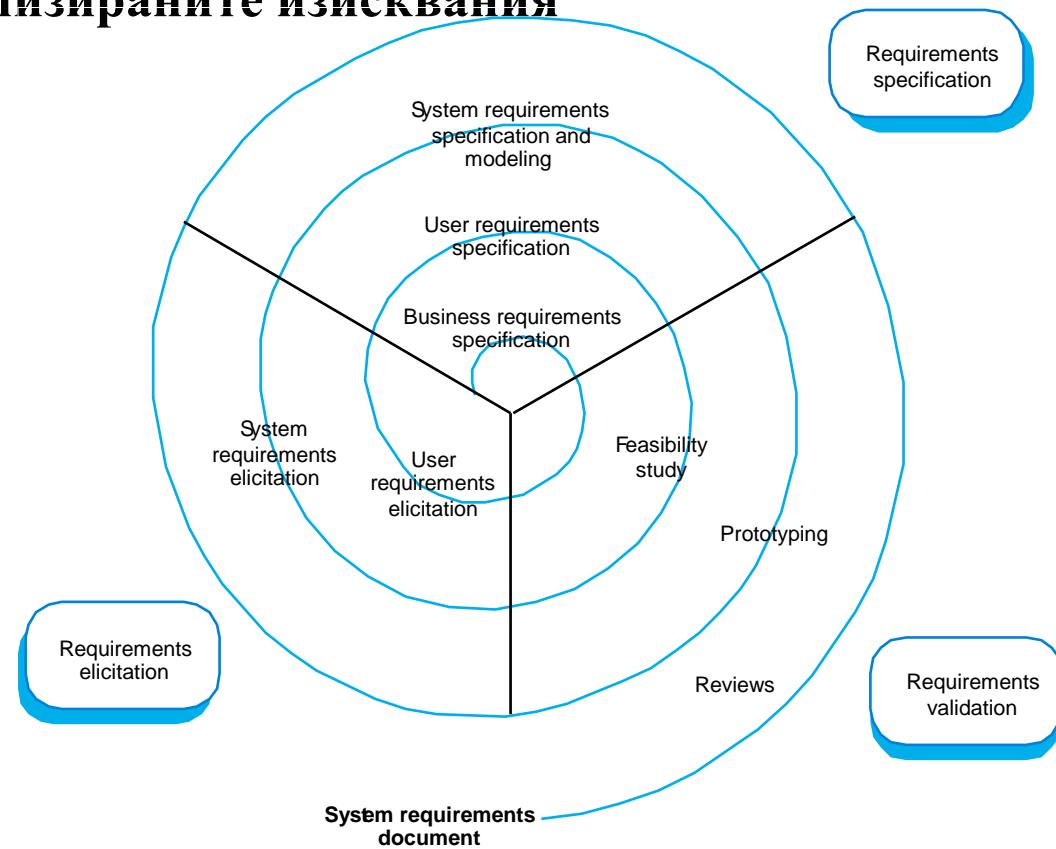
- Високо ниво – крайна цел
- Подцели – декомпозиция на крайната цел (**refinement-усъвършенстване**)

Трябва да се установят подцелите на организацията, включително и общите цели на бизнеса, описание на проблема, който трябва да се реши, защо е необходима системата и ограниченията върху системата.

- Целите дефинират съответни сценарии

Итеративен процес на идентифицирането, анализ и документиране на изискванията

- Резултатът от идентификацията на изискванията е **чернови документ** с изискванията за системата
- Това е входна информация за **етапа на анализ на изискванията**
- **Документиране на вече анализирани изисквания**



Проблеми на идентифицирането на изискванията

- Заинтересованите лица (ЗЛ) *не знаят* какво точно искат.
- ЗЛ изразяват своите изисквания *на свой собствен език*.
- Различните ЗЛ могат да имат *противоречиви изисквания*.
- *Организационните и политически фактори* може да окажат влияние върху системните изисквания.
- Изискванията *се променят* по време на процеса на анализ. Може да се появят *нови ЗЛ*; бизнес средата се изменя.
- Знанията за приложната област трябва да се съберат *от различни източници* като наръчници, ръководства, хора
- *Няма достатъчно време*
- *Жаргоните и терминологията* варират за различните компании
- *Хората не искат да се променят*

Техники за извличане на изискованията

Особености на прилаганите техники:

- Задаване на въпроси
- Колаборативна работа
- Изграждане (на прототип) и разиграване (сценарий, др.)
- Наблюдение

Видове техники за извличане на изискованията

1. Сценарии, случаи на употреба
2. Интервюта, въпросници, анкети
3. Мозъчна атака
4. Семинари (срещи) за изискованията; FAST
5. Soft systems methods
6. Наблюдения и социален анализ
7. Прототипиране
8. Други: Повторно използване на изискованията

Сценарии и потребителски случаи

1. Сценарии - 1

- Сценариите са **примери от реалния живот** за това как може да се използва системата.
- Те трябва да включват
 - Описание на началното състояние;
 - Описание на редовната последователност на събитията;
 - Описание на това, какво може да се сгреши;
 - Информация за други успоредни дейности;
 - Описание на състоянието, когато сценарият завърши.

Сценарии - 2

■ Сценарий

Точно определена последователност на взаимодействие между актьора и системата

Обикновено са кратки (например между 3 и 7 стъпки)

Описват: *положителни* (необходимо/нормално поведение)
отрицателни (нежелателни взаимодействия)

Могат да бъдат *показателни* (описват текущата система) или *незадължителни* (как трябва да бъде)

Предимства

Много са *естествени*: обикновено заинтересованите лица ги използват спонтанно

Пример: “да предположим, че съм приет в болница – какво се случва по време на моето приемане?”

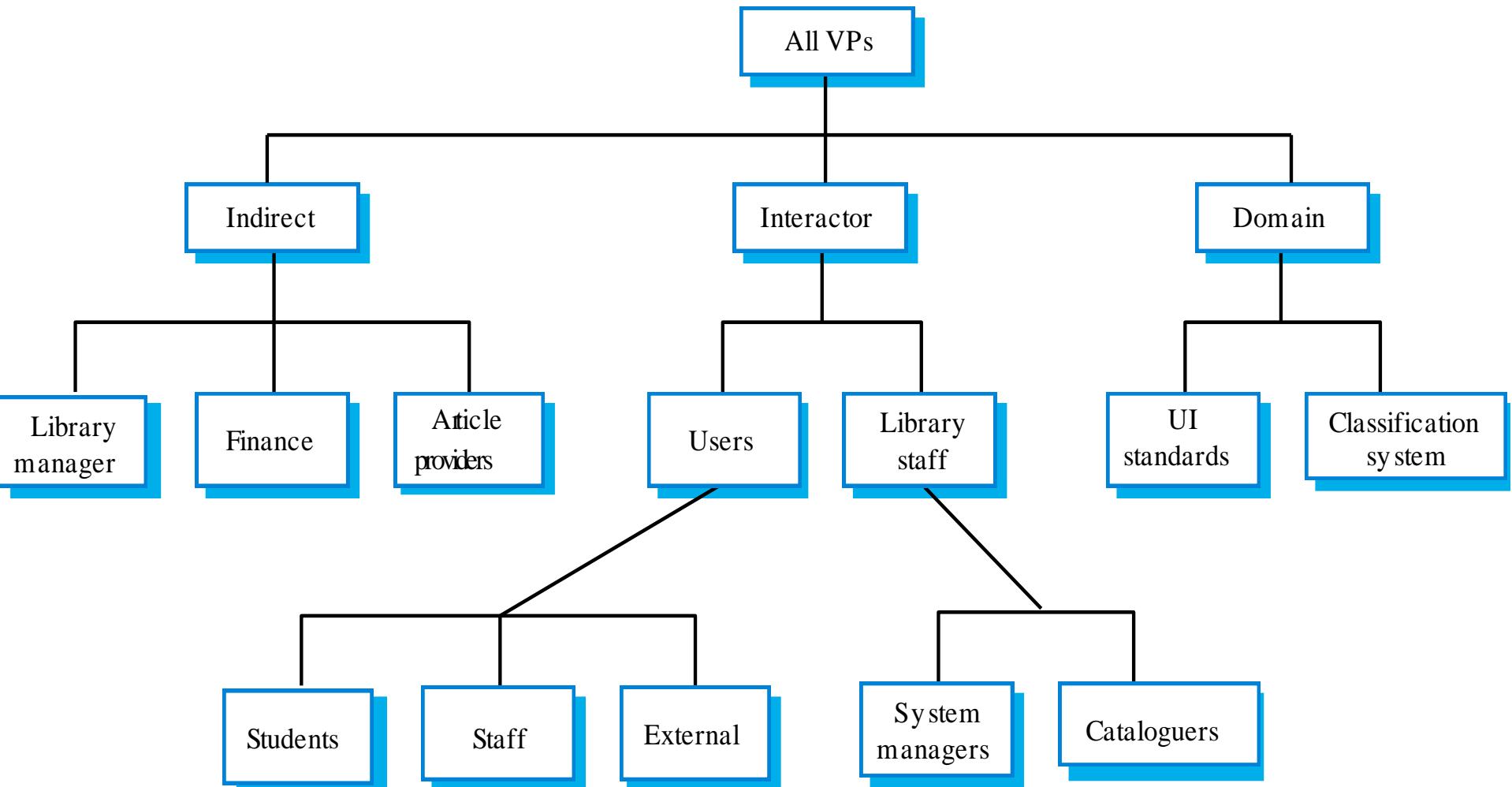
Типичен отговор: “Вие, или човекът, който ви придружава, ще говори с лицето на receptionията. Трябва да покажете своята здравна карта и да обясните кой ви е насочил към болницата. След това...”

Кратките сценарии са много добри за бързо илюстриране на специфични взаимодействия

Недостатъци

Липса на структура; Трудно се проверяват за пълнота

Пример: LIBSYS library system



Пример: LIBSYS сценарий (1)

Първоначално предположение: Потребителят е вписан в системата LIBSYS и е открил документа, който съдържа копие на статията.

Нормален: Потребителят *избира статията*, която ще бъде копирана. След това системата изисква от потребителя да *предостави информация за абониране* за документа или да укаже как ще плати за статията. Други възможни методи за плащане са с кредитна карта или чрез предоставяне на номер на сметка. След това на потребителя се *представя форма за авторски права*, която той/тя трябва да попълни и която запазва детайли за трансакцията, след което потребителят я изпраща на системата LIBSYS.

Формата за авторски права *се проверява* и ако е валидна, PDF версията на статията се зарежда в работната област на LIBSYS на потребителския компютър. Потребителят се уведомява, че статията е налична. От потребителя се иска да избере принтер и се *отпечатва* копие на статията. Ако статията е била отбелязана като “само за принтиране”, тя се изтрива от потребителската система, след като потребителят потвърди, че принтирането е завършило.

Пример: LIBSYS сценарий (2)

Какво може да се обърка: Потребителят може *да не успее да попълни* правилно формата за авторските права. В такъв случай, формата трябва отново да се покаже на потребителя за извършване на корекции. Ако формата отново е погрешна, на потребителя се отказва исканата статия.

Заплащането може да не бъде прието от системата. На потребителя се отказва исканата статия.

Свалянето на статията може да прекъсне. Прави се нов опит до постигане на успех или потребителят прекратява сесията.

Може *да не е възможно да се принтира статията*. Ако тя не е отбелязана като “само за принтиране”, тогава тя се задържа в работното пространство на LIBSYS. В противен случай, статията се изтрива и потребителският акаунт *is credited* с цената на статията.

Други дейности: Едновременно сваляне на други статии.

Състояние на системата при завършването: Потребителят е вписан. Свалената статия е изтрита от работната област на LIBSYS, ако е била отбелязана само за четене.

Потребителски случаи (структурен метод за идентифициране и анализ на изискванията)

- **Потребителските случаи** са техника, базирана на сценарии, която идентифицира *актьорите* в едно *взаимодействие* и която описва самото взаимодействие.
- **Наборът от потребителски случаи:** описва всички възможни взаимодействия със системата.
- **Диаграмите на последователността** добавят детайли към потребителските случаи като показват последователността на обработката на събитията в системата.
- **Диаграма на дейностите**

Потребителски случаи - допълнение

Потребителските случаи са предложени в Objectory method (1993). Те групират един главен сценарий със съответните алтернативни сценарии и изключения.

Деф.: Спецификацията от последователности от действия, включително варианти от последователности и последователности от грешки, които една *система*, *подсистема*, или *клас* могат да извършват като си взаимодействат с външни обекти, за да предоставят услуга или стойност.

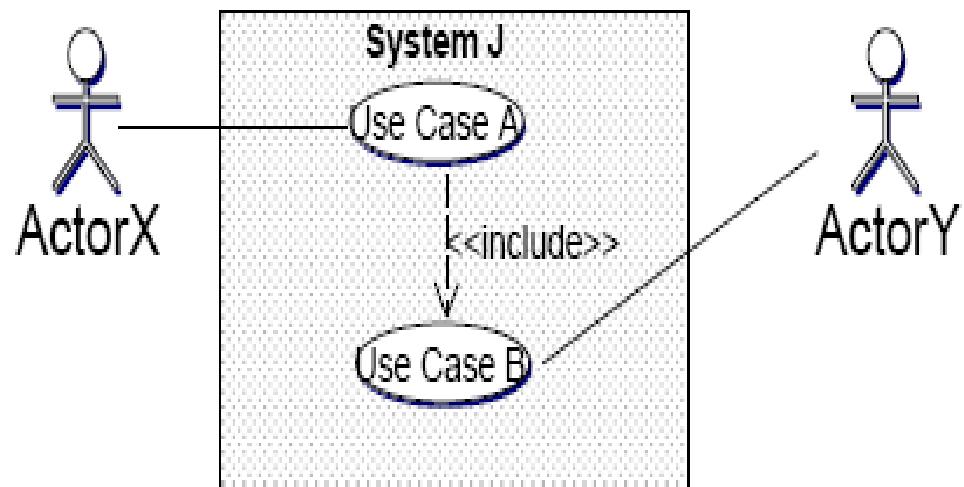
Диграма на потребителските случаи

Диаграмата на потребителските случаи за една система представя:

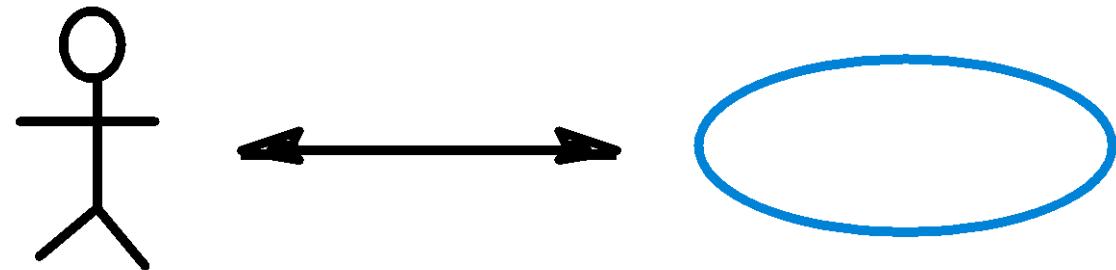
- Границата на системата;
- Потребителски случаи (от високо ниво в системата);
- Актьори: хора или системи от контекста
- Връзки между потребителските случаи и актьорите;
- Връзки (ако има такива) между потребителските случаи и/или между актьорите (generalization, extend, include).

Потребителски случаи в UML

- UML предоставя графично представяне на потребителските случаи, наречено *Диаграма на потребителските случаи*.
- То позволява графично изобразяване на:
 1. актьори
 2. връзки
 3. зависимости
 4. генерализации (обобщения)
 5. пакети
 6. границата на системата.

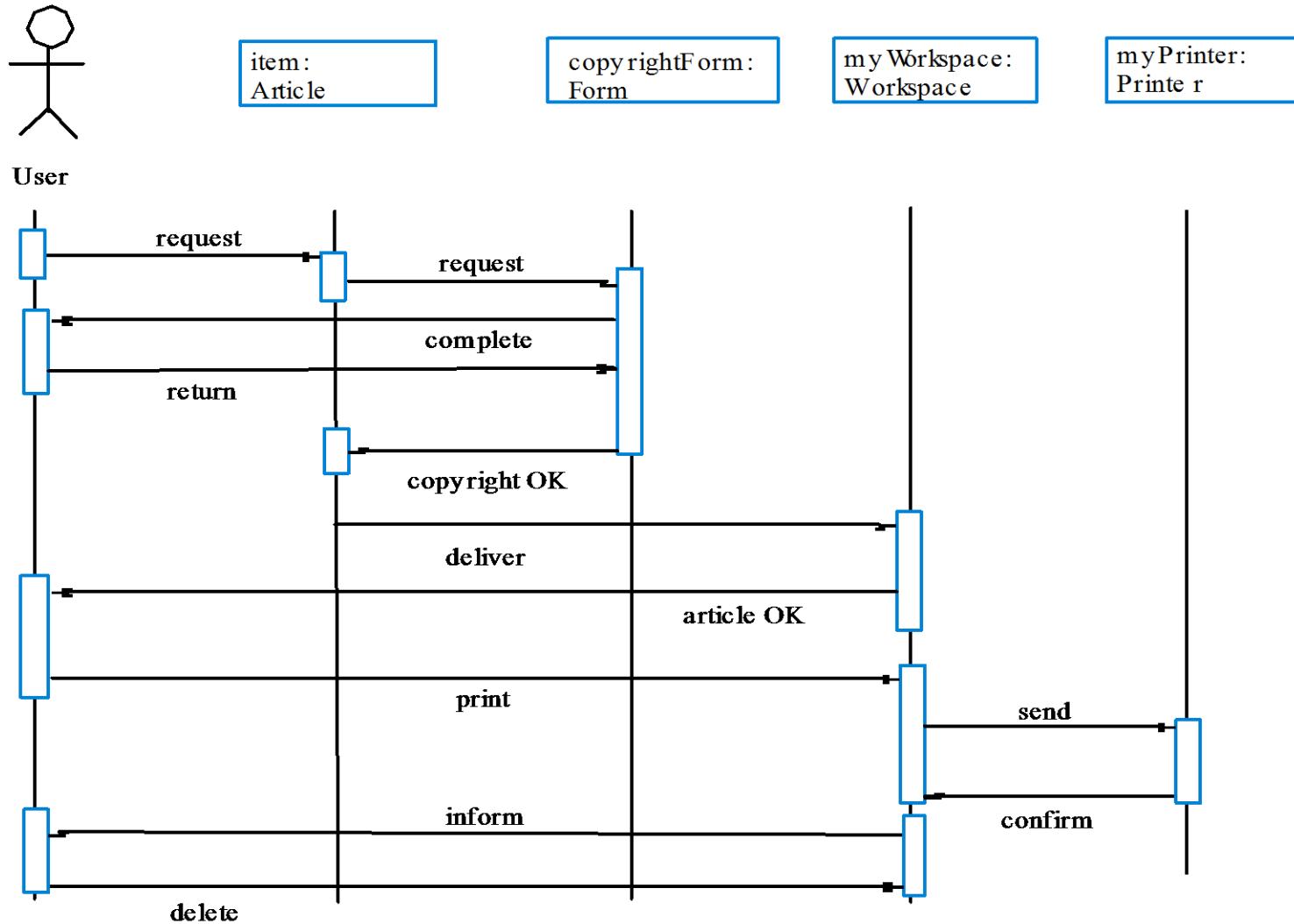


Пример: Потребителски случай за принтиране на статья

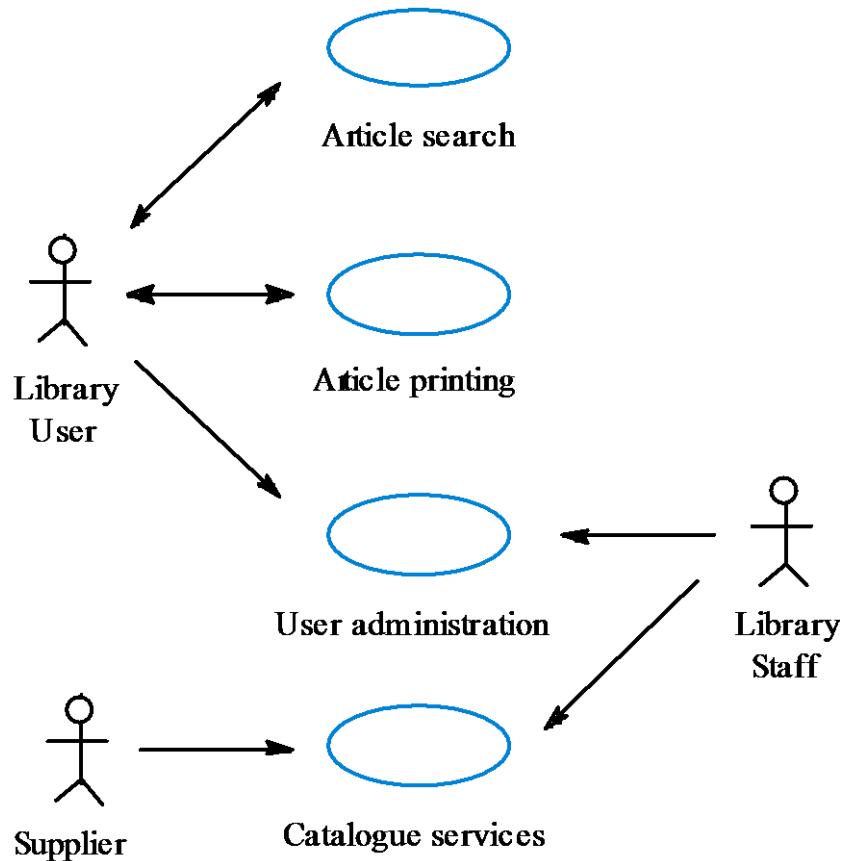


Article printing

Пример: Последователност за случай “принтиране на статия”



Пример: Потребителски случаи в LIBSYS



2. Интервюта -1

- ❑ Интервюирането е основен източник за изискванията
- ❑ Интервюирането изиска:
 - търпение,
 - изслушване,
 - разбиране и
 - задаване на въпроси в различни посоки.

Интервюта - 2

- ❖ Изискванията се обсъждат с различни заинтересовани лица, за да се изгради разбиране за техните изисквания.
- ❖ Видове интервюта
 - Затворени интервюта - търсят се отговори на предварително подгответи въпроси
 - Отворени интервюта. Няма предварително установлен ред на въпросите. Изискванията се обсъждат в диалог.

.... but blurred ...

Съществени елементи на интервюирането

- Интервюиращите трябва да бъдат *непредубедени и да не провеждат интервюто с предварително формирани идеи за това какво се изиска.*
- На заинтересованите лица *трябва да се даде стартова точка* за дискутиране.
- Трябва да се има предвид организационната политика – *много от реалните изисквания може да не се обсъждат* заради политически (в общия смисъл) ограничения.

Характеристики на интервюирането

Интервюто се фокусира върху въпроси за:

Бизнес случай за проекта

Функционални изисквания за проекта

Рискове

Ограничения

Зainteresовани лица

Какво не може да се открие по време на интервю?

- изясняване на областта на системата
- знание за организацията, стандарти
- рядко са напълно достатъчни за инженеринга на изискванията

3. Мозъчна атака /Brainstorming/

- **Метод за генериране на идеи в група**
- **Brainstorming е работа в екип, която предизвиква откриването на знания от всеки участник**
 - надграждане на идеите на другите,
 - освобождаване на мисленето и
 - разглеждане на проблема от различни гледни точки
- Помага при разглеждането на алтернативи и правене на правилни избори
- Едно от най-добрите средства да се определи дали получените изисквания са правилни или не
- Включва не само дискусия, но и сесия за *съвместно управление на знания*
- **Особености на метода**

4. Работни срещи (Requirements workshops)

- Анализаторът обединява основните заинтересовани страни, за да анализира системата и да разработи решението.
- Изискванията се идентифицират съвместно от заинтересованите страни
- В идеалния случай се провеждат в контролирана среда чрез обмен на знания.
- Ръководителят на срещата поддържа **фокусирана** дискусия, **протоколчик** документира дискусията.
- Често са необходими **множество срещи**, за да приключи процеса успешно.
- *Недостатъци:* Може да се окаже трудно да се съберат заедно всички необходими заинтересовани страни едновременно.

Техника за специфициране с облекчено приложение /Facilitated Application Specification Technique (FAST)/

Има за цел да покрие празнината между мисленето на *разработчиците* и желанието на *клиентите*. Всички заинтересовани страни се считат за екип, за който се организира:

- Среща на неутрално място
- Правила за подготовка и участие
- Дневен ред – формално + неформално
- Водещият управлява срещата
- Механизъм на дефинирането (начин на провеждане на срещата)

- *Идентифициране на проблем, предлагане на решение, уговоряне на подходи определяне на решение като набор от изисквания*

Каква е разликата с мозъчна атака?

5. Soft Systems methods (SSM)

- Чрез тях се получават *неформални модели* на цялостната социално-техническа система
- SSM обръщат внимание на това, че софтуерните системи са вградени в един по-широк човешки и организационен контекст
- **Това не са техники за детайлно идентифициране на изискванията. Те са по-скоро начин да се разбере даден проблем и неговия контекст в организацията**
- (Задължително) се съчетават с други техники на ИИ

Етапи на SSM

1. Оценка на проблемната ситуация
2. Описание на проблемната ситуация
3. Абстрактна дефиниция на системата от избрани **гледни точки**
4. Концептуално моделиране на системата (human activity models)
5. Сравнение модел/реален свят
6. Идентификация на промяната
7. Препоръки за действие

Пр. Система за оповестяване на наводнения

6. Наблюдение и социален анализ

- Значимост на техниката – защо и кога се прилага; исторически корени.
- **Етнографът прекарва известно време в наблюдение на хората по време на работа и така си изгражда представа за това, как се извършва работата.**
- **Етнографията може да се използва за извлечане на:**
 - а) социални изисквания, но за конкретен краен потребител (Пр.)**
 - б) организационни изисквания (Пр.)**

Пример: „Екраните“

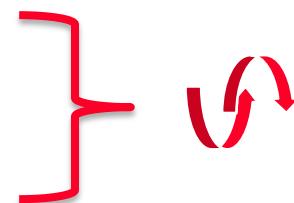
Подход и характеристики на етнографското изследване

Етнографските проучвания са ефективни, когато *работата е по-обширна, сложна и динамична, отколкото е представена чрез опростените системни модели.*

- ❖ Хората не трябва да обясняват или да говорят за своята работа, а да бъдат наблюдавани. (*примери*)
- ❖ Етнографските проучвания зависят от личния характер на етнографа, типа на процеса и на участващите хора.

Насоки за етнографското изследване

- Приемете, че хората *са добри* във вършенето на своята работа. Търсете нестандартните начини за работа.
- Отделете време *да опознаете хората и да установите връзка на доверие*.
- Водете си *подробни бележки* за всички практики. Анализирайте ги и правете заключения въз основа на тях.
- Видео и аудио записи могат да са полезни, ако не са много дълги.
- Организирайте редовно *разбор (debriefing session)*, при което етнографът говори с хора извън процеса
- Комбинирайте етнографията с *други техники за идентификация защо?* :
 - Комбинирайте с отворени интервюта
 - Комбинирайте с прототип



Етнографски перспективи – механизъм за организиране и структуриране на етнографския архив

Три основни гледни точки:

- **Обстановката на работата**

Това описва *контекста и физическото местоположение* на работата и как хората използват обектите, за да изпълняват задачите си.

Например, при проучване на хелпдеск.

- **Социална и организационна перспектива**

Да се разкрие *какво се случва ежедневно* в работата според *различните участници*. Тази гледна точка се опитва да *обедини (различието) на* всички възприятия.

- **Процеса на работа**

Представя работата като поредица от дейности, при което информацията тече от една дейност към друга.

7. Повторно използване на изискванията (Requirements reuse)

- Повторното използване включва вземане на изискванията, които са били разработени за една система и използването им за друга.
- *Спестява време и усилия*, тъй като изискванията вече са били анализирани и валидирани в други системи.
- Повторното използване на изискванията *е неформален процес*, но **по-систематично** използване може да доведе до по-голямо спестяване на разходите (*Пр. Design patterns*)).

Възможности за повторно използване

- Когато изискването се отнася до *предоставяне на информация за приложната област (ограничения на системата)* Пр.
- Когато изискването се отнася до *стила* на представянето на информацията (*Пр. характеристики на интерфейса*). Повторното използване води до съгласуваност на стила между приложенията.
- Когато изискването отразява *политиките на компанията*, като например *политиката за сигурност, за съхранение на лични данни, етика на диалога с потребители ...*

Повече от 50% от изискваният може да попаднат в тази група!

Възможни са и проблеми (какви?).

Други изследвания

Идентифицирането на изискванията може да включва и *проучване на документи:*

- Индустриски стандарти, закони, и/или наредби
- Литература за продукта (собствени или на конкуренцията)
- Документация на процеси и инструкции за работа
- Заявки за промяна, доклади за проблемите или помощни доклади
- Научени уроци от предишни проекти и изработени продукти
- Отчети и други резултати от съществуващи системи

Инженеринг на изискванията

Извличане на изискванията – прототипиране. Анализ на изискванията

Лекция 5

- Техники за извлечение на изискванията (продължение)

Прототипиране

- Анализ на софтуерните изисквания
- SMARTT изисквания

8. Прототип

Дефиниция: Прототипът е **начална версия на система, която може да се използва за експериментиране.**

- ✓ Хардуерни системи
- ✓ Софтуерни системи

Цели и роля на прототипирането в различни етапи на софтуерната разработка

- **Изяснява и допълва (завършва) изискванията**
-
- **Може да прerasне в цялостен краен продукт** (в определени случаи) посредством последователност от работни цикли.
- **Изследва проектни решения и алтернативи** като начини на
 - взаимодействие с потребителя,
 - оптимизация на използването на софтуера,
 - оценка на технически подходи.

Прототипите в дейностите на ИИ - особености

- ***Бързото разработване*** на прототипа е много важно, за може да бъде наличен *в ранния етап* на процеса на идентификация.
- ***За целта:***
 - реализират се само определени функционалности
 - не се реализират някои качествени изисквания (RT, memory, error handling...)
 - може да не се реализират някои основни процеси на „класическата“ разработка (мениджмънт, тестване ...);

Прототипите в подкрепа на дейности на инженеринга на изискванията

Прототипирането може да се прилага при два процеса на ИИ (Boehm, 1984 г.).

- ❖ **извлечане на изискванията и**
- ❖ **валидиране на изискванията**

1. Прототипите са полезни за *идентификацията на изискванията*, тъй като потребителите могат да експериментират със системата и да посочат нейните силни и слаби страни. Те имат *нещо конкретно*, което да критикуват.
2. Чрез прототип при *валидиране на изискванията* лесно могат да се коригира/оптимизира първоначалния вид на системата.

system reqmt

software req.

arch. design

det. design

coding

testing

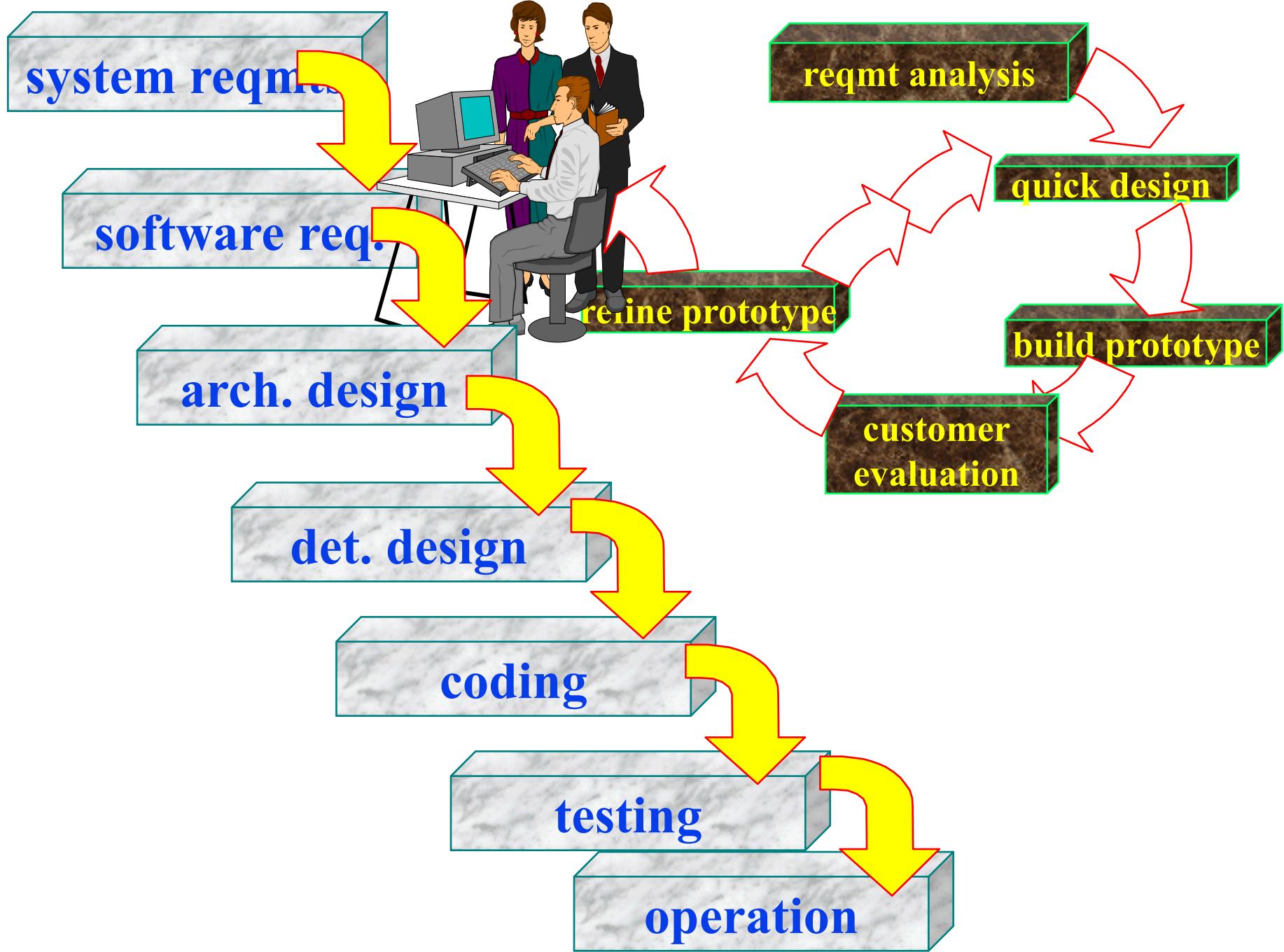
operation

reqmt analysis

quick design

build prototype

customer evaluation



Видове прототипи – използване - 1

- **Throw-away прототип**
 - подпомага специфицирането
 - бърза разработка и кратко “съществуване”, което обуславя занижено качество на изпълнение
 - преизползване на отделни компоненти
- *Приложение:* хардуерни системи; за големи системи; Wizard of Oz, mock-ups;
- **Еволюционен прототип** – не се предоставя детайлна спецификация, но изиска прецизност в изработката.
 - основна техника в съвременните софт. технологии- **зашо?**
 - за малки и средни по големина системи, но не и за големи проекти
(зашо?)

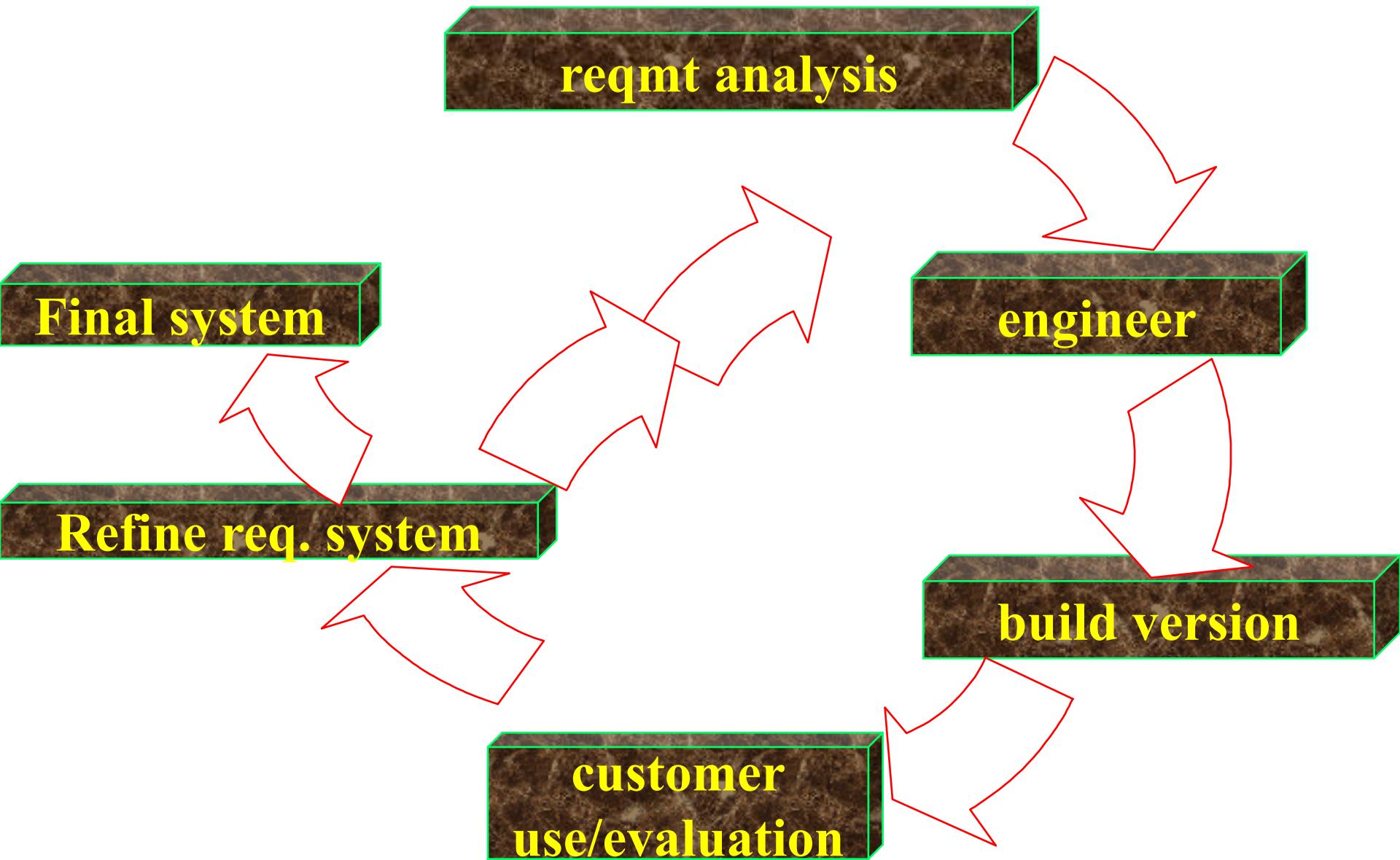
Приложение - e-commerce приложения, web-site разработване

Boundaries ...?

Модел на Throw-away прототипирането



Модел на еволюционното прототипиране



Видове прототипи и видове изисквания, които те прототипират - 1

- **Throw-away прототипиране**
 - идентифицира/реализира изисквания, които *създават най-много затруднения на клиентите и които са най-трудни за разбиране.*
 - обикновено се ползва *само* в помощ на *идентифициране* на изискванията.
 - бърза реализация.

Видове прототипиране и видове изисквания, които те прототипират 2

- Еволюционно прототипиране
 - целта му е *бързо* да предостави *работеща система* на клиента.
 - Затова изискванията, които трябва да се поддържат от началните версии са тези, които са *добре разбираеми* и които могат да предоставят полезна функционалност за крайните потребители.

Vague boundaries ...

- **Хоризонтален (по поведение) прототип or mock-up**

Не се “гмурка” във всички слоеве на архитектурата и не реализира реална работа.

Първично описание на част от *интерфейса*; изследва *специфични* страни на бъдещата система.

На фокус е *общата визия*; широк кръг изисквания и работни потоци

Пример: Western movie.

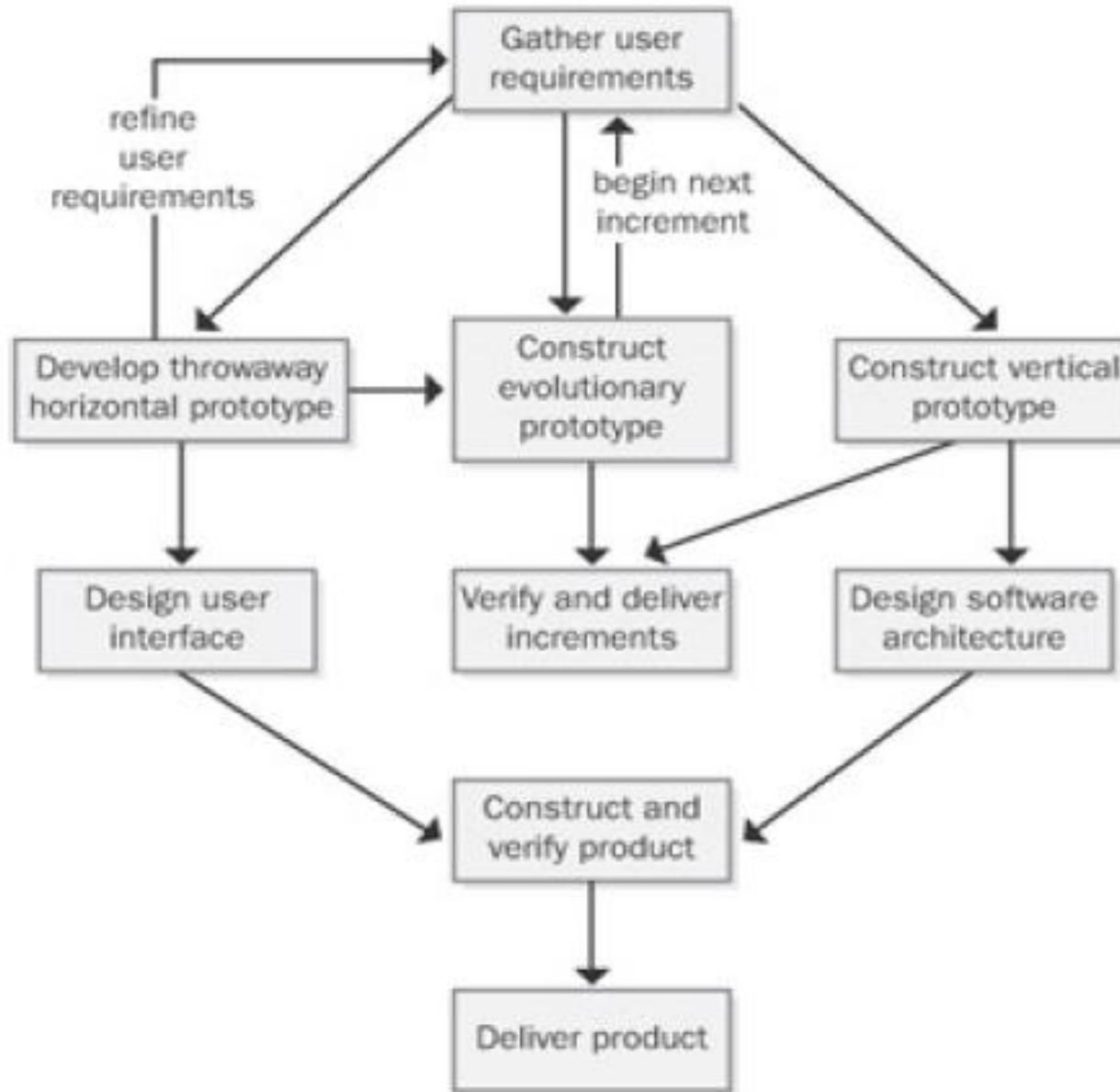
- **Вертикален (structural or proof of concept) прототип**

Реализира отделна част от системата в цялостност на интерфейс и технически услуги.

Прилага се в случай на неизяснени въпроси от *архитектурата*, *оптимизиране* на алгоритъма, *схемата на БД*, ограничения за време...

Пример (Karl E. Wiegers): „Вертикален прототип, който имплементира само част от потребителския интерфейс и съответната сървърна функционалност (на UNIX), ни позволи да оценим комуникационните компоненти, производителността и надеждността на предложената от нас клиент/сървър архитектура. Експериментът беше успешен, както и реализацията, базирана на тази архитектура.“

Възможни начини за включване на прототипирането в разработването на софтуера (Karl E.Wiegers, 2003)



Typical applications (1)

| | Throwaway | Evolutionary |
|------------|--|--|
| Horizontal | <ul style="list-style-type: none">Clarify and refine use cases and functional requirements.Identify missing functionality.Explore user interface approaches. | <ul style="list-style-type: none">Implement core use cases.Implement additional use cases based on priority.Implement and refine Web sites.Adapt system to rapidly changing business needs. |

Typical applications (2)

| | Throwaway | Evolutionary |
|------------|--|--|
| Vertical | <ul style="list-style-type: none">• Demonstrate technical feasibility. | <ul style="list-style-type: none">• Implement and grow core client/server functionality and communication layers.• Implement and optimize core algorithms.• Test and tune performance. |
| Horizontal | <ul style="list-style-type: none">• Create a minimum viable product (MVP) to validate market demand. | <ul style="list-style-type: none">• Integrate with existing systems and infrastructure.• Refine user interface and experience based on feedback. |

Други ползи от прототипирането

- Позволява *детайлно разглеждане* на изискванията като открива несъгласуваност и пропуски.
- *Оценява осъществимостта* на системата и използваемостта ѝ преди да се правят големи разходи за разработката.
- (единствения) **ефективен** начин за разработване на *потребителския интерфейс*.
- Може да се използва за:
 - създаване на *тест* на системата
 - за съставянето на *документация*
 - за *обучение*

Подходи за *бързо* прототипиране

- **Прототипиране на хартия**
създава се хартиен mock-up на системата и се използва за системни експерименти
- **‘Wizard of Oz’** прототипиране
отговорите на системата за определени входни потребителски данни се симулират от човек
- **Автоматизирано** прототипиране
използва се език от четвърто поколение или друга среда за *бързо* разработване, за да се създаде изпълним прототип

Кога ще използвате всеки един от тези подходи?

Разработване на изпълним прототип

- ✓ Езици от четвърто поколение, базирани на системи за бази от данни.
- ✓ Езици за визуално програмиране като Visual Basic или ObjectWorks, в които има *готови обекти*.
- ✓ Интернет базирани прототипи, основани на уеб браузъри и езици като Java – *готови интерфейси, сегменти (аплети)*, които се стартират автоматично при зареждане в браузера.

Прототипирането на интерактивни системи е много по-лесно отколкото прототипирането за критични системи и системи в реално време (**зашо?**).

Проблеми и разходи при прототипирането

- **Разходи за обучение** – разработването на прототипи може да изиска *специални инструменти*
- **Разходи за разработката** – зависят от вида на прототипа. Разходите в началото на софтуерния процес, но ги намалява в по-късните етапи (**зашо?**).
- **Разширен график за разработка** – разработката на прототип *може* да увеличи времето за работа, но времето за прототипиране може да се възвърне в последствие, тъй като се избягва преработката на софтуер.
- **Незавършеност:** 1) Прототипът не е завършена система!
2) **Не е** възможно да се прототипират *критични* системни изисквания;
3) **Не е** възможно да се прототипират някои качествени изисквания;

Оценяване на прототипа

Специфични (**what**) въпроси:

- Дали прототипът реализира функционалността?
- Липсва ли някоя функционалност?
- Има ли някакви състояния на грешки, които прототипът не засяга?
- Има ли ненужни функции?
- Колко логична и завършена е навигацията?
- Бяха ли много сложни някои от задачите, реализирани с него?

Правилните хора оценяват прототипа в подходящи перспективи:

- Включват се ***опитни, но и неопитни*** представители на потребителската група.

Ограничения при разработването на прототип

Работещата версия има примамлив вид.

но !

- Не правете един throw-away прототип по-сложен отколкото е необходимо, за да бъдат постигнати прототипните цели.
- Не се поддавайте на натиска от страна на потребителите.

Същевременно

Не е необходимо да изхвърляте прототип, а да бъде запазен за повторно използване.

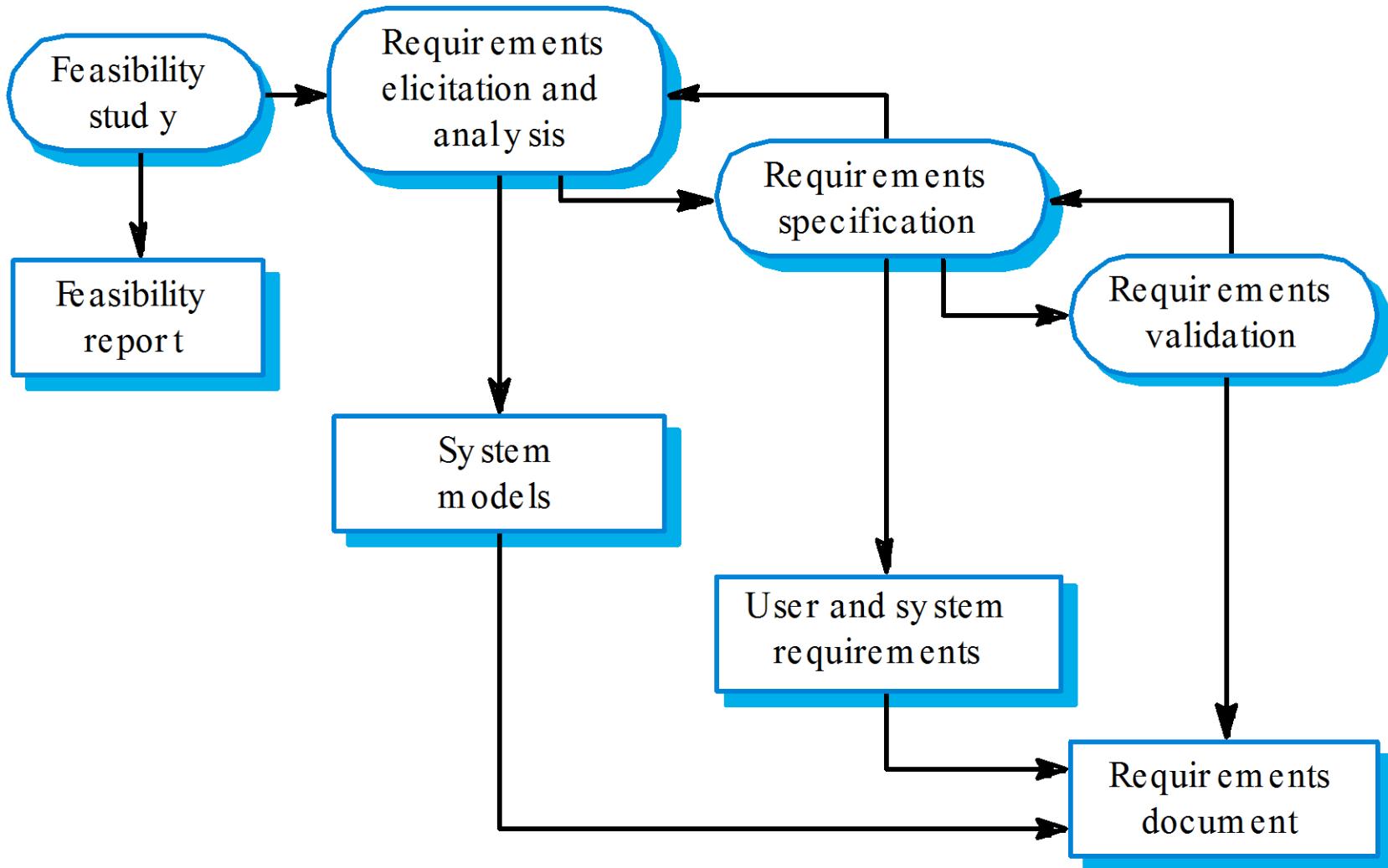
Той обаче няма да бъде включен в крайния продукт. Поради тази причина може да го наричате *no releasable* прототип.

Коя е най-добрата техника за извлечане на
изискванията?

Анализ на изискванията

- **Анализ на изискванията**
- **Процес на анализ на изискванията**
- **Техники за анализ на изискванията**
- **Договаряне на изискванията**

Процес на инженеринг на изискванията

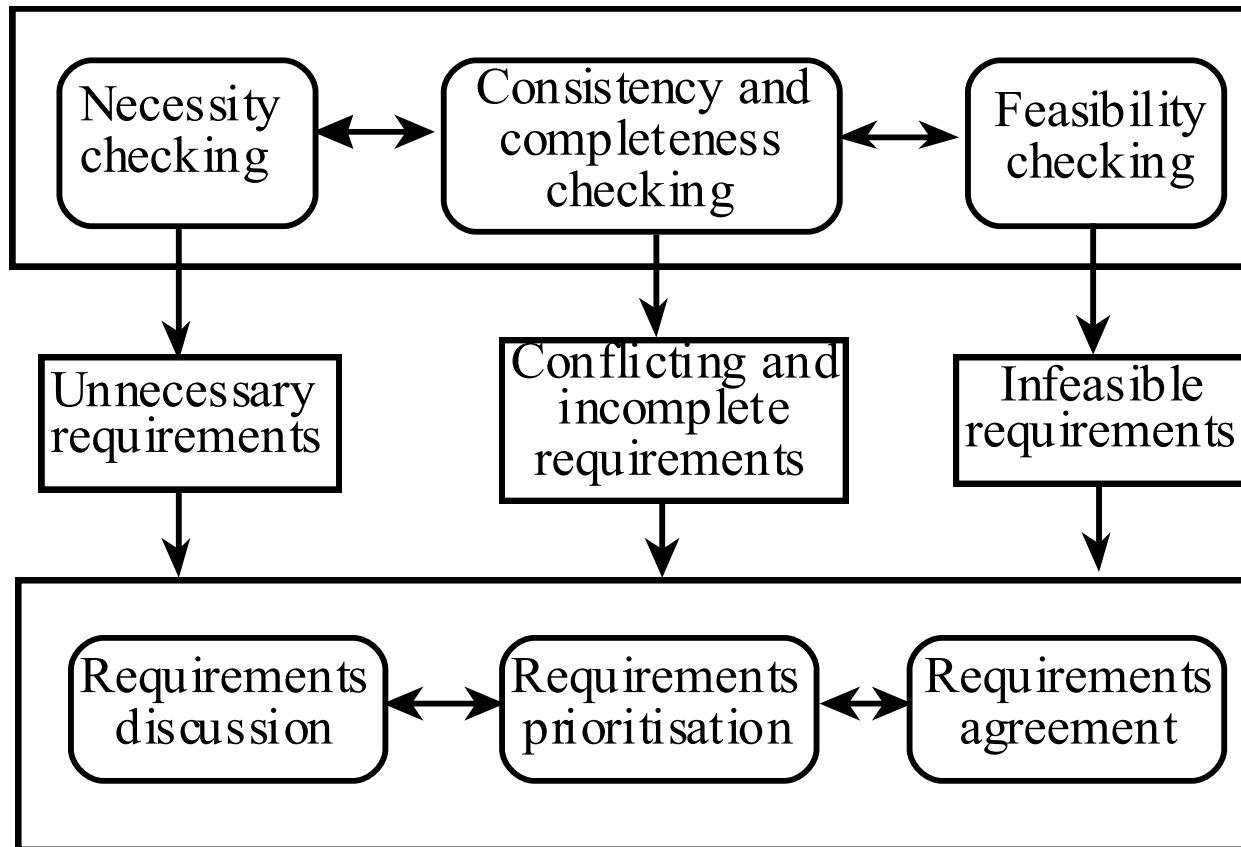


Анализ на изискванията

- Целта на анализа е да **открие проблеми** като *незавършеност и несъгласуваност* в идентифицираните изисквания, както и да обсъди и **разреши** със заинтересованите лица в процеса *на преговорите*.
- Анализ и извличане на изискванията – взаимна зависимост (*interleave*)
- Анализът е зависим от опита и експертизата на хората, които са включени в процеса на анализ.

Процеси на анализ и преговори на изискванията

Requirements analysis



Requirements negotiation

Проверки на анализа

- **Проверка за необходимост**

Анализира се необходимостта от изискването, за да не допускаме изисквания, които *не допринасят за постигането на бизнес целите* на организацията или не спомагат за решаването на специфичния проблем, който се адресира от системата.

- **Проверка за съгласуваност и завършеност**

Изискванията се **cross-checked** за съгласуваност и завършеност. Съгласуваност означава, че изискванията не трябва да си противоречат; завършеност означава, че никои от необходимите услуги и ограничения не са пропуснати.

- **Проверка за осъществимост**

Изискванията се проверяват за това дали *са изпълними в контекста на бюджета и плана* за разработването на системата.

Анализ на изискванията

Каква е разликата между анализ на изискванията и валидиране на изискванията?

Анализ на изискванията - фактори

Функционалните изисквания се представят *на съответното ниво на детайлност*.

Пример:

- 1) Един уеб сайт, който се изгражда на стъпки (инкрементално) от малък, добре синхронизиран екип, може да бъде и с ограничена документация.
- 2) За една сложна вградена система, която ще бъде outsourced отдалечно, е необходима точна и детайлна SRS.

Техники за анализ на изискванията

1. Списък (Checklist) (1)

- Списък от ключови и достатъчно **общи** въпроси за оценка на всяко едно изискване.

Въпроси (*колко?*) за **a)** “стандартни” системи?

b) критични системи?

- Систематичен подход, опит.
- Списък (таблица):
Requirement x Checks (limited number) + comments

Списък (примерен) за анализа 1

- **Premature design**

Включва ли изискването предварителна информация за проектиране или за реализацията?

- **Обединени изисквания (Combined)**

Дали описание на едно изискване описва единствено изискване или може да се раздели на няколко различни изисквания?

- **Ненужни изисквания (Unnecessary)**

Дали изискването е ‘gold plating’? Т. е., дали изискването е козметична добавка към системата и в действителност не е необходимо.

- **Използване на нестандартен хардуер**

Изискването предполага ли използването на нестандартен хардуер или софтуер?

Списък (примерен) за анализа 2

- **Следване на бизнес целите**

Дали изискването е в съгласие с бизнес целите, дефинирани в уводната част на документа с изискванията?

- **Неяснота на изискванията (ambiguity)**

Може ли да се прочете по различен начин от различните хора? Какви са възможните интерпретации на изискването?

- **Реалистичност на изискванията (realism)**

Изискването реалистично ли е при дадената технология, която ще се използва за реализацията на системата?

- **Възможност за тестване на изискванията**

Може ли изискването да се тества, т. е. дали е изразено по такъв начин, че QA инженерите могат да съставят тест, който да покаже дали системата изпълнява това изискване?

Техники за анализ на изискванията:

SMART(T) Изисквания

(„Software Engineering Notes“, vol.20, n.2, 1995, pp. 42-47)

Specific (Специфични)

Отделни, т. е. множество елементи (изисквания/стъпки/и др.) не са обединени в един и не се дублират

Measurable (Измерими)

Могат да се определят количествено по някакъв начин (дори и да е само с TRUE/FALSE за една или повече качествени мерки)

Attainable (Достижими)

Могат да бъдат постигнати в рамките на физическото съществуване на системата (реалистичност на функционални и нефункционални изисквания)

Realisable (Реалистични)

Могат да бъдат реализирани в рамките на съществуващите ограничения (време, ресурси, и др.)

Traceable (Проследими)

Възможност за проследяване на изискването от неговото замисляне, спецификация до неговия дизайн, реализация и тестване; връзка с други.

Testable (Възможност за тестване)

Може да се провери чрез съответните средства за тестване

Характеризиране на добрите изисквания - 1

Характеристики на добрите изисквания – различни според различните автори. Тези характеристики могат да послужат и за приоритизиране на изискванията. Следните характеристики обаче са общоприети

| | |
|------------------------------|---|
| <i>Неделимо</i> (Cohesive) | Изискането се отнася само за едно единствено нещо. |
| <i>Пълно</i> | Изискането е пълно изказано без липсваща информация . |
| <i>Съгласувано</i> | Изискането не противоречи на никое друго изискане и е напълно съгласувано с цялата външна документация. |
| <i>Без връзки</i> (атомарни) | Изискането е <i>атомарно</i> , т.е., не съдържа връзки. |
| <i>Проследимо</i> | Изискането отговаря на определена бизнес нужда или част от нея според искането на заинтересованите лица и е достоверно документирано. |

Характеризиране на добрите изисквания - 2

Текущо

Изискването не е остаряло с времето.

Осъществимо

Изискването може да се реализира според ограниченията на проекта.

Недвусмислено Изискването е изказано сбито без използване на технически жargon. То изразява конкретни факти.

Интерпретира се по един единствен начин.

Отрицателните твърдения са забранени.

Задължително Изискването представя характеристика, дефинирана от заинтересованите лица.

Проверимо Реализацията на изискването може да се определи чрез *един от четири възможни метода:* изследване, демонстрация, тест или анализ.

Пример:

"Background Task Manager ще предоставя съобщения за статуса на редовни интервали не по-малки от всеки 60 секунди."

- Какви са съобщенията за статус?
- При какви условия и по какъв начин се предоставят на потребителя?
- Ако се показват, колко дълго остават видими?
- Времевият интервал не е ясен и фразата "всеки" обърква израза.

Тези крайни интерпретации са съгласувани с първоначалното изискване, но определено не са това, което потребителят е имал предвид.

Поради тези проблеми, изискването не е проверимо.

Следователно ...

Решение:

Един начин за решение е да се пренапише предишното изискване, след като се получи повече информация от клиента:

1. Background Task Manager (БТМ) ще показва съобщенията за статуса в предназначената за целта област от потребителския интерфейс.

1.1 Съобщенията ще се обновяват **на всеки 60 плюс секунди** след като започне обработката на background задачата.

1.2 Съобщенията ще остават **видими непрекъснато**.

1.3 Когато комуникацията с background процеса е възможна, БТМ ще показва какъв процент от background задачата е завършен.

1.4 БТМ ще показва съобщение “Завършен“, когато background задачата е изпълнена.

1.5 БТМ ще показва съобщение, ако background задачата е спряла (stalled).

1. Защо изискването е разделено на множество изисквания?
2. Преработените изисквания не уточняват, как ще бъдат показвани съобщенията за статуса. Защо?

Това е въпрос на дизайна; ако бъде зададен върху изискването, ще ограничи дизайна и работата на разработчика.

Прибръзаните ограничения върху опциите за дизайна пречат на програмистите и могат да доведат до suboptimal дизайн на продукта.

НО!

- *analysis paralysis*:

Не можете да удължавайте да усъвършенствате изискванията прекомерно.

- Целта е да се напишат изисквания, които са *достатъчно добри*, за да позволят на екипа да продължи с дизайна и конструкцията при допустими нива на риска.

2. Взаимодействия между изискванията

- Една много важна цел на анализа на изискванията е да открие **взаимодействията** между изискванията и да посочи **конфликтите и припокриванията** между изискванията
- **Матрицата на взаимодействие** на изискванията показва как изискванията си взаимодействват помежду си.
Изискванията се подреждат по редовете и по колоните на матрицата
 - ✓ За изисквания, които си противоречат, се попълва 1
 - ✓ За изисквания, които се припокриват, се попълва 1000
 - ✓ За изисквания, които са независими, се попълва 0

Матрица на взаимодействието

| Requirement | R1 | R2 | R3 | R4 | R5 | R6 |
|-------------|------|----|------|------|----|------|
| R1 | 0 | 0 | 1000 | 0 | 1 | 1 |
| R2 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3 | 1000 | 0 | 0 | 1000 | 0 | 1000 |
| R4 | 0 | 0 | 1000 | 0 | 1 | 1 |
| R5 | 1 | 0 | 0 | 1 | 0 | 0 |
| R6 | 1 | 0 | 1000 | 1 | 0 | 0 |

За тази матрица дефинирайте:

- взаимодействията на R1
- независими изисквания
- открийте броя на конфликтите за от всеки тип за всяко изискване. Задайте подходящи формули.

Взаимодействие на изискванията

- Когато не може да се вземе еднозначно решение за конфликт, се попълва 1. Защо?
- Какво означава съществуването на голям брой припокривания и/или конфликти?
- Размерът на матрицата е ограничен – (вероятно максимум 200 изисквания). Защо?

Приоритизиране на изискванията с използване на MoSCoW

- MoSCoW е техника за приоритизиране, която се използва за постигане на съвместно споразумение със заинтересованите лица за важността, която те поставят върху доставянето на всяко от изискванията

First developed by Dai Clegg of Oracle UK Consulting; in CASE Method Fast-Track (RAD)

M – ЗАДЪЛЖИТЕЛНО ТРЯБВА да го има.

S – ТРЯБВА да го има, ако въобще е възможно.

C – МОЖЕ да го има, ако не влияе на нещо друго.

W – НЯМА да го има този път, но БИХА искали да го има в бъдеще.

3. Договаряне на изискванията

Обсъждане на изискванията

Изискванията, които са били отбелязани като проблематични, се обсъждат.

Участващите заинтересовани лица представят своите възгледи за изискванията.

Приоритизиране на изискванията

Спорните изисквания се приоритизират, за да се идентифицират критичните изисквания и да се подпомогне процеса по вземане на решение.

Споразумение за изискванията

Откриват се решенията на проблемите при изискванията и се уговоря компромисен набор от изисквания. Обикновено това включва извършване на промени в някои от изискванията.

Бележки: Разпространение на копие от резюмето на дискусията.
Ръководителят на срещата е независим човек !

Договаряне на изискванията

- **Договарянето на изискванията е процес на обсъждане на конфликтите между изискванията и постигане на споразумение, с което всички заинтересовани страни са съгласни: организационни нужди, специфични потребителски изисквания, ограничения, бюджет на проекта и планиране.**
- Разногласията за изискванията *са неизбежни*, когато една система има много заинтересовани страни. Конфликтите не са '*неуспехи*', а отразяват нуждите и приоритетите на различни заинтересовани лица
- При планирането на процеса за инженеринг на изискванията е важно да се остави *достатъчно време за преговори*. Откриването на приемлив компромис може да отнеме време

Срещи за преговори - стъпки

1. *Информационен етап*, на който се обяснява същността на проблемите, свързани с едно изискване.
2. *Етап на дискутиране*, на който участващите заинтересовани лица обсъждат как могат да се решат тези проблеми.

Всички заинтересовани лица, които са засегнати от изискването, трябва да имат възможност да коментират. *На този етап могат да се назначат приоритети за изискванията.*

3. *Етап на разрешаване*, на който се уговарят действията, относящи се до изискването.

Тези действия могат да бъдат изтриване на изискването; да се предложат специфични промени в изискването или да се извлече допълнителна информация за изискването.

Преговори за изискванията

- Скъп и времеотнемащ процес. Защо?
- Различните аналитици могат да имат **различни решения** на проблемите. Защо?
- Невъзможност тези процеси да се структурират, систематизират и автоматизират. Изискват се самостоятелни решения, опит..., дипломатичност
- Преговорите за изискванията понякога са повече въпрос на **политически решения**

Пример: Изискване на привилегирован достъп на мениджърския състав до локални данни. Същевременно, персоналът от охраната може да иска единствено мениджърът да има такъв достъп. Конфликтът би се решил в преговори.

Ключови моменти

- Идентифицирането на изискванията включва познаване на приложната област, специфичния проблем, който се решава, организационните нужди и ограничения и специфичните улеснения, от които се нуждаят заинтересованите страни.
- Процесите за идентификация, анализ и договаряне на изискванията са *итеративни, interleaved processes*, които обикновено трябва да се повторят няколко пъти.
- Има различни техники за идентификация на изискванията, които могат да се използват, включително интервюиране, сценарии, soft systems methods, прототипиране и етнография.

Ключови моменти

- Прототипите са ефективни за идентификацията на изискванията, защото заинтересованите лица имат нещо, с което *могат да експериментират*, за да определят своите същински изисквания.
- *Checklists* са особено полезни като начин за организация на процеса по анализиране на изискванията. Те напомнят на анализаторите какво да търсят, когато четат предложените изисквания.
- Преговорите за изискванията винаги са необходими за *разрешаване* на конфликтите между изискванията и премахване на припокриванията. Преговорите включват обмяна на информация, обсъждане и разрешаване на несъгласия.

АНАЛИЗ НА СОФТУЕРНИТЕ ИЗИСКВАНИЯ

Специфициране на изискванията

Лекция 6

Съдържание

- Роля на метода на инженеринга на изискванията
- Документиране на изискванията на естествен език
- Специфициране на изискванията чрез аналитични модели:
 - Data-flow модел
 - Семантични модели на данните
 - Обектно ориентирани методи за анализ
 - Формални модели при ИИ

Методи за инженеринг на изискванията (ИИ)

- Процесът на инженеринг на изискванията обикновено се ръководи от *метод* на инженеринг на изискванията.
- **Методът на ИИ е систематичен подход за документиране и анализиране на изискванията на системата.**
- Нотацията, която осигурява средство за изразяване на изискванията, е свързана с метода.

Необходими характеристики на метода:

- **Точност на описанията** чрез избраната нотация с възможност за проверка за последователност и коректност.
- Подходящ запис на изискванията, удобен за *споразумение* с крайните потребители: например, **интегриране на формални и неформални описания**.
- Да подпомага **формулирането на изисквания** на **всички** етапи (запис, структуриране, анализ, решения, перспективи и връзки).
- Възможност да се **дефинира** (моделира) **вънния** свят.
- Възможност за **отразяване на промените** без да е необходимо да се преработи целия набор от изисквания.
- Възможности за **интегриране** на други (различни) методи (техники за моделиране) с цел пълнота на описанието.
- Да осигурява възможност за **комуникация** на хората (заради идеите и получаване на обратна връзка).
- Използване на **софтуерен инструмент** – начин да се управлява сложността.

Няма идеален метод !

- Методите използват различни *техники за моделиране*, за да формулират системните изисквания.
- Системният модел може да бъде обогатен чрез моделиране на *различни аспекти* чрез използваните на различни техники за моделиране.
- Примери за :
 - Модел на потока на данните (Data flow model)
 - Композиционен модел (Compositional model): *np.* (Entity relationship)
 - Модел на класовете (Classification model) Пр.: Object diagrams
 - Модел на поведението (Stimulus-response model)
 - Модел на процесите (Process model)

Специфициране на изискванията (Requirements specification)

Дефиниция: Процесът на записване на потребителските и системните изисквания в документ за изисквания.

- Изискванията на потребителите трябва да бъдат разбираеми от крайните потребители и клиентите, които нямат техническа подготовка.
- Системните изисквания са по-подробни изисквания и могат да включват повече техническа информация.
- Системните изискванията могат да бъдат част от договор за разработване на системата
 - Затова е важно те да са възможно най-пълни.

Аспекти при документирането 1

- Независимо от използвания език и метод на работа **три аспекта** трябва да бъдат документирани:
- **Функционалност: описва се с трите перспективи на системата**
 - **Данни:** използване и структура
 - **Функции:** условия, обработка, резултати
 - **Поведение:** Динамиката на проявление
- Нормалните случаи, както и изключенията трябва да бъдат представени.

Аспекти при документирането 2

- **Аспект на качеството**
- Начин на представяне на системата
 - Обем на данните
 - Време за реакция
 - Скорост
 - ...

Специфициране на стойности на метрики

- „**-ilities**“ като
 - Използваемост
 - Надеждност
 - Други

Аспекти при документирането 3

- **Ограниченията**

- Технически: протоколи, интерфейси, ..
- Правни: закони, стандарти, нормативи
- Културни
- Околна среда
- Физически
- Решения / ограничения, изисквани от ключови З.Л.

Как да документираме - начини

Представяне, ориентирано към решения (Documenting Solution-oriented Requirements):

Неформално

- Естествен език

Полуформално

- Структурни модели
- Модели на взаимодействието



Диаграми, обогатени с текст.

Формално

- Формални модели



Специфициране на естествен език

- Изискванията са написани като *естествени езикови изкази*. Могат да бъдат допълнени с диаграми и таблици.
- Изразителен, интуитивен и универсален начин.
- Разбирам за клиентите.

Указания за писане на изисквания

- Въведете *стандартен* формат и го използвайте за всички изисквания.
- Използвайте изказ по последователен начин.
 - за задължителни изисквания,
 - за желаните изисквания.
- Използвайте *маркиране* на текст, за да идентифицирате ключови части от изискването.
- *Избягвайте* използването на компютърен *жаргон*.
- Включете *обосновка* защо е необходимо изискването.

Документиране на естествен език: запис на изисквания, насочени към софтуерно решение

Пример:

R15: If a glass break detector at the window detects that the pane has been damaged, the system shall inform the security service.

Как трите перспективи са отразени в това изискване?

Проблеми при специфицирането на естествения език

- **Липса на яснота и на точност**
 - Точността се постига трудно без да се чете и проверява документа.
- **Смесване на изисквания**
 - Функционалните и нефункционалните изисквания са склонни да бъдат смесвани.
- **Сливане на изискванията**
 - Няколко различни изисквания могат да бъдат изразени заедно.

Phrase templates

[Rupp et al. 2009
ISO/IEC/IEEE 29148:2011]

Use **templates** for creating **well-formed** natural language requirements

Typical template:

[<Condition>] <Subject> <Action> <Objects> [<Restriction>]

Example:

When a valid card is sensed, the system shall send
the command 'unlock_for_a_single_turn' to the turnstile
within 100 ms.

Структуриран запис на естествен език - 1

Ip. A structured specification of a requirement for an insulin pump

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r_2); the previous two readings (r_0 and r_1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

Структуриран запис на естествен език - 2

Пр. A structured specification of a requirement for an insulin pump

Action

CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements

Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition

The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r0 is replaced by r1 then r1 is replaced by r2.

Side effects None.

Таблична спецификация - приложения

- за допълване на описание на естествен език.
- когато трябва да се дефинират няколко възможни начина на действие.
- при вградени системи

Пример:

The insulin pump system bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Пример. Tabular specification of computation for an insulin pump

| Condition | Action |
|--|---|
| Sugar level falling ($r_2 < r_1$) | CompDose = 0 |
| Sugar level stable ($r_2 = r_1$) | CompDose = 0 |
| Sugar level increasing and rate of increase decreasing $((r_2 - r_1) < (r_1 - r_0))$ | CompDose = 0 |
| Sugar level increasing and rate of increase stable or increasing $((r_2 - r_1) \geq (r_1 - r_0))$ | CompDose = = round $((r_2 - r_1)/4)$ If rounded result = 0 then CompDose = MinimumDose |

Документиране с използване на модели на системата

- Data and object modelling
- Behaviour modelling
- Function and process modelling
- User interaction modelling
- Goal modelling
- UML



- Моделите се използват главно да представят *функционалните изисквания*, докато нефункционалните се специфицират на естествен език.

Структурно описание на системата: Модел

Какво е модел?

Има ли общоприета дефиниция на модел?

All models are wrong – some are useful.

George E. P. Box

Модел

Моделът е абстракция, опростено изображение на обекта на изследване във форма, различна от реалното му съществуване.

Моделът отразява или възпроизвежда обекта на изследването в достатъчна степен, за да позволи получаване на информация за неговото изучаване.

Модели на софтуерна система: концептуални модели

Какво е модел?

Защо и кога се използват при ИИ?

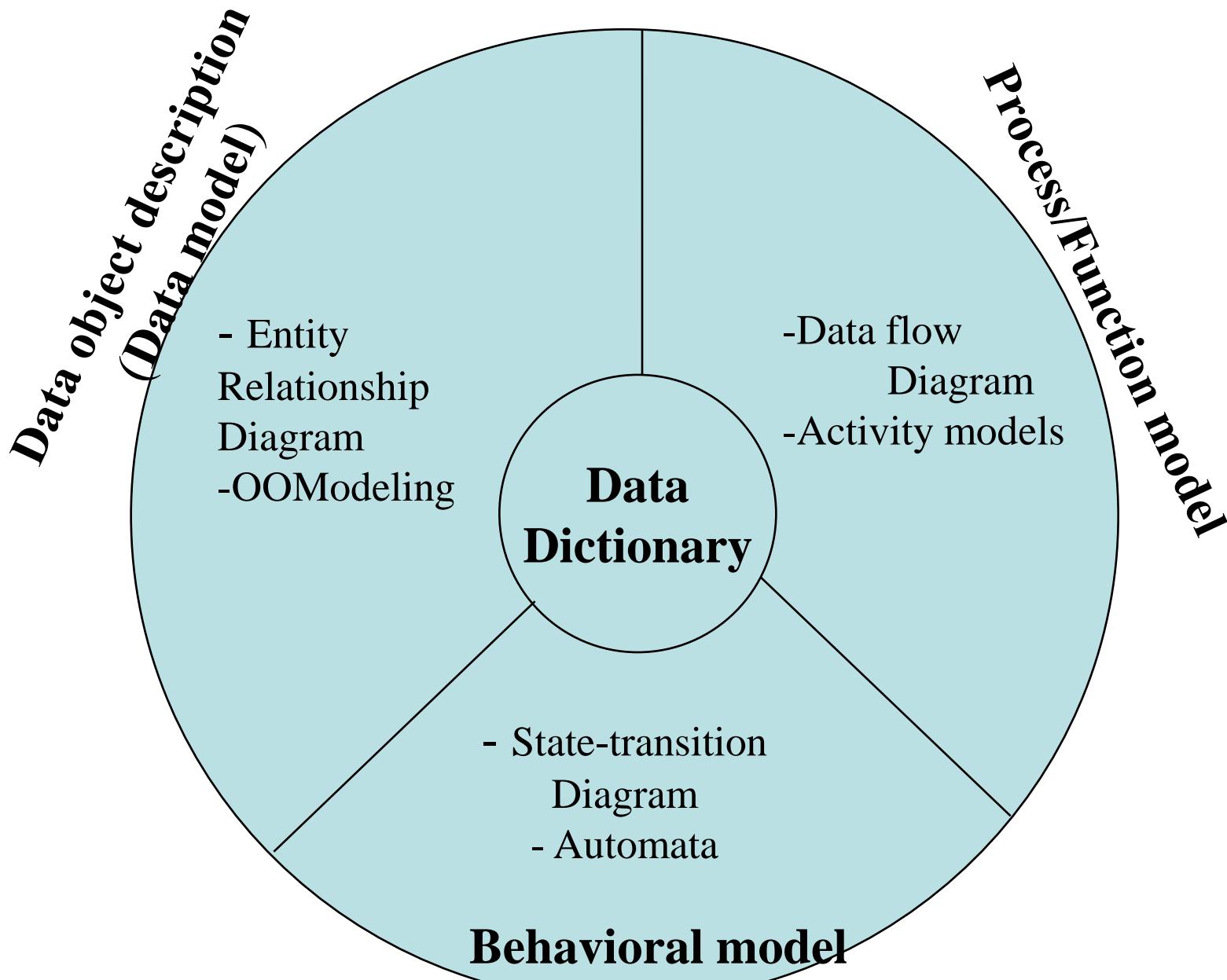
Моделите на системата са (важен) **мост** между анализа на дадена система и проектирането ѝ.

- Моделите са графични или математически представления.
- Моделите на системата се основават на (*изчислителни*) *концепции* (*computational concepts*) като обекти и функции.

Перспективи на софтуерната система:

- Перспективи на решението:
 - Функции
 - Поведение
 - Данни (статична структура)
- Връзка между трите перспективи

Модели на различните перспективи



Модели според определена перспектива на системата

1. Модели на функциите - как се обработват/преработват данните в системата

Пример: Модел на потока на данните (Data-flow models (DFM))

2. Модели на поведението (Stimulus-response/behavioral models) - модели на състоянията на системата

Примери: Диаграми на преходите (state machine models, state transition diagrams); Спецификация на управлението (control specification)

3. Модели на данните - как някои същности (entities) на системата се изграждат от други; Как същностите имат общи/еднакви атрибути и имат общи характеристики

Пример: Обектно описание и композиционни модели (Object description and composition models)

Какво друго може да се моделира? (виж Тема 4)

- **Взаимодействието на потребителя със системата**
 - Модели на контекста
 - Случаи на употреба, сценарии
 - Диаграма на последователностите
- **Описание на целите**
 - Дърво на целите

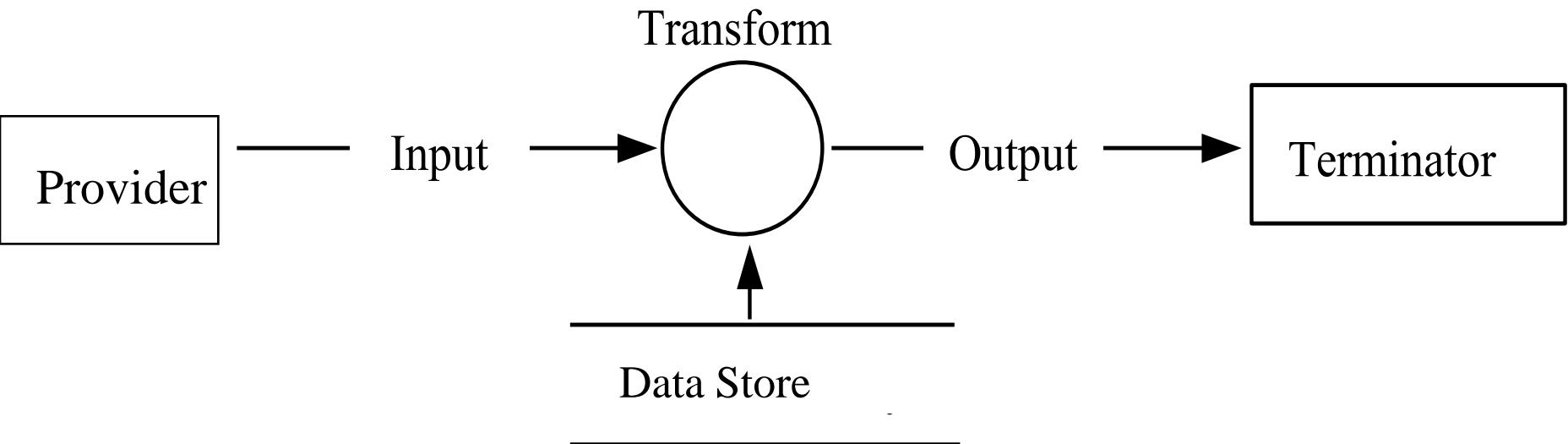
Перспектива на функциите:

Модели на функциите и на потока на действията

1. Модел на потока на данните Data Flow Models (DFM)

- DFM представя потока на информация в рамките на системата, както и между системата и средата.
Той *не показва* последователност на поведение и на управляваща информация.
- Диаграми (модел) на потока на данните е графично представяне на:
 - външни за процеса елементи
 - процеси (функции)
 - поток на данните
 - хранилища за данни
- Показва постепенното преобразуване на данните

Data Flow Diagram (Model): нотация



- *Прости и интуитивни*
- *Пример*

DFD елементи

External Entity (външни същности) - източник или получател на данни. Те обикновено са извън обхвата на изучаваната област.

Process - трансформира или манипулира данни

Data flow - пренася данни между източник и процес, процес и получател, или между два процеса

Data Store - хранилище на данни

Connector symbols (конекторни символи), ако диаграмата се представя на няколко страници.

DFD йерархия

- DFD могат да се използват на различни нива на абстракция (*levels of abstraction*).
- Най-високото ниво на DFD е Контекстаната диаграма.
- Процесът може да се покаже подробно на по-ниско ниво на DFD.
 - В такъв случай, нетните входящи и изходящи потоци трябва да бъдат балансирани на съответните нива на DFD.
 - По-ниското ниво на DFD може да има складове за данни, които са локални за него и не са показани на по-високо ниво на DFD.

Най-ниското ниво в DFD

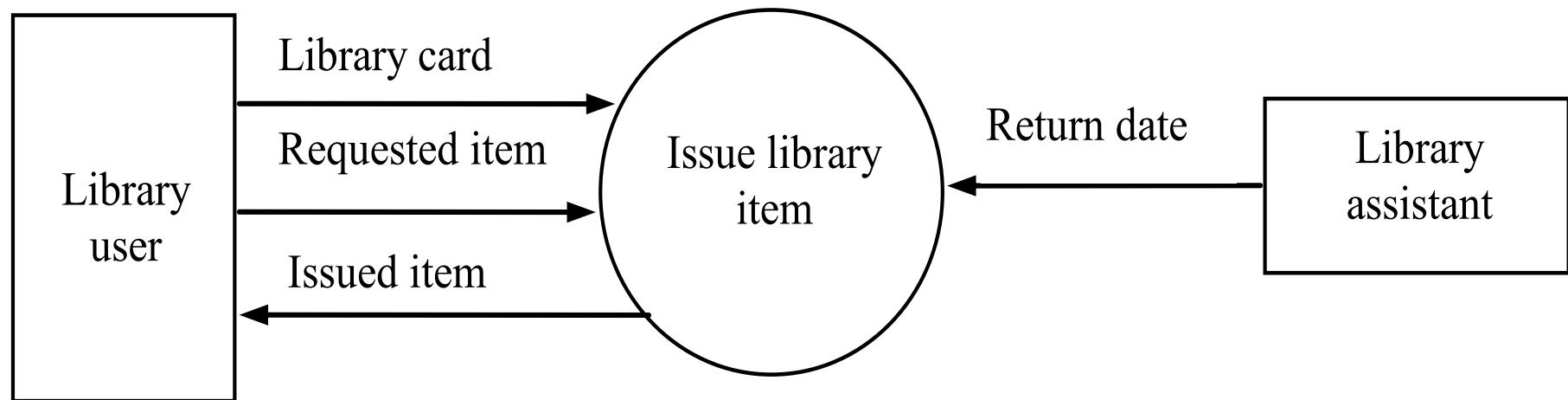
- Процес, който не може да бъде разгледан по-детайлно се свързва с мини спецификация.
- Мини спецификация:
 - Структурирано описание на естествен език
 - Диаграма на действия
 - Таблици (Decision Table, Decision Tree)

Пример: DFD на електронна библиотека

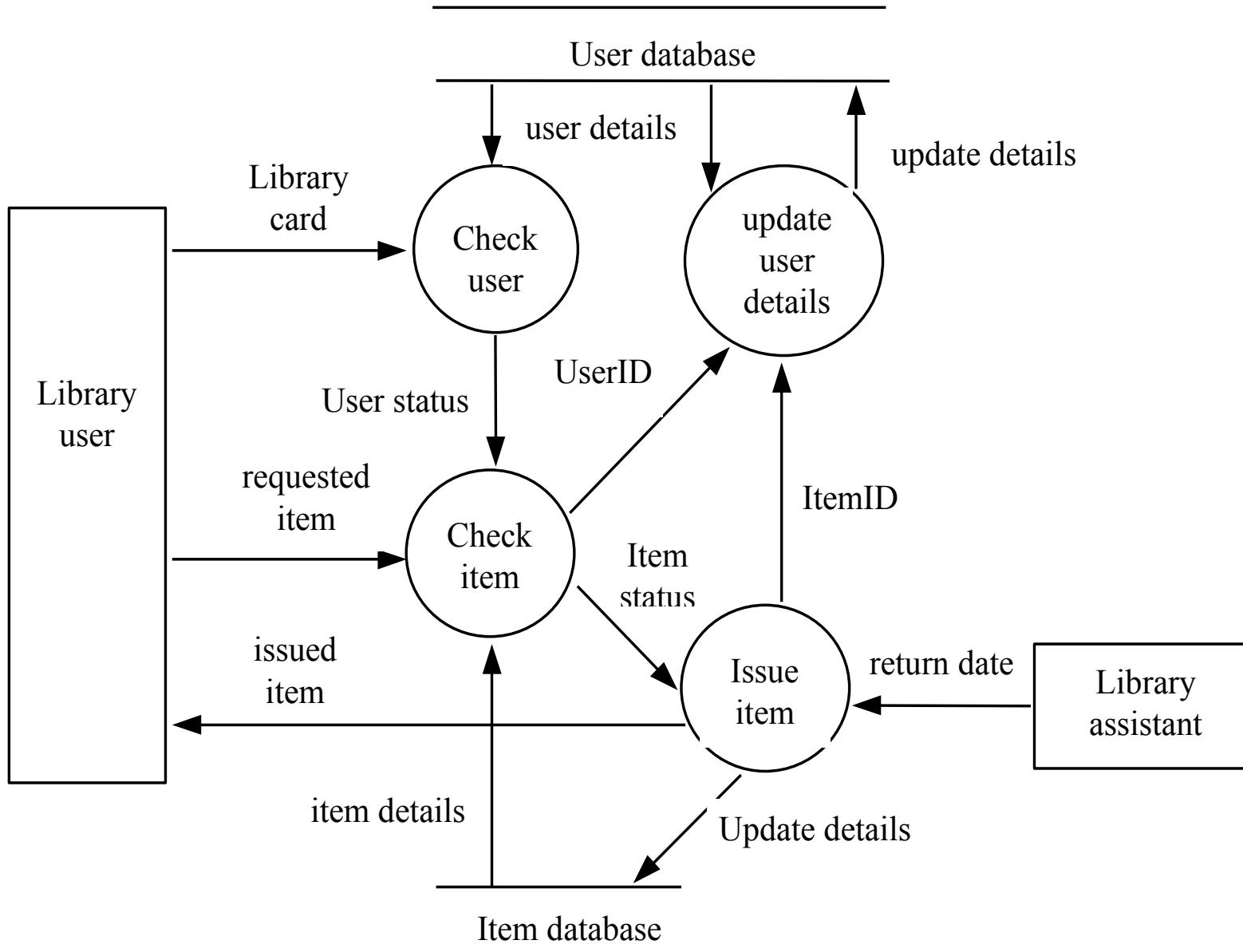
Библиотечна (елементарна) система за автоматизирано издаване (заемане) на библиотечни документи.

- **Първата диаграма** на потока на данни е контекстно ниво
- **Level 1** на DFD е конструирано чрез разлагане “балона” на библиотечната система в под-функции
- **Level 2** ...
- ...

Library example - Context level data flow diagram



Library example - Level 1 DFD



Означения - вариабилност

Няма единно означение на елементите на DFD

- Показаното означение е въведено от DeMarco
- Gane и Sarson използват *rounded rectangles* за процеси, *shadowed rectangles* за източници и получатели, and *squared off C's* за хранилища на данни
- Orr използва *rectangles* за процеси, *ellipses* за източници и получатели и за хранилища на данни

Допълнение: DFD Guidelines

- Choose *meaningful* names for processes, flows, stores, and terminators
 - use active unambiguous verbs for process names such as Create, Calculate, Compute, Produce, etc.
- *Number* the processes: a DFD at any level, should have no more than *half* a dozen processes and related stores
- Make sure that DFD is internally *consistent*
 - *avoid infinite* sinks
 - *avoid spontaneous* generation processes
 - *beware of unlabeled* flows and *unlabeled* processes
 - beware of *read-only* or *write-only* stores

Структурен анализ

- Подходът на описание на потока на данни е типичен за метода на структурен анализ (Structured Analysis (SA))
- Две основни стратегии доминират структурния анализ
 - "Old" метод популяризиран от DeMarco 1978 г.
 - Лесен за разбиране +
 - Позволява декомпозиция +
 - Описанието на данните е остаряло –

"Modern" подход от Yourdon (1984, 1990)

Подход на DeMarco

- Подход отгоре-надолу

Физическата система се описва с модел на логическия поток на данни чрез:

- Анализ на текущата физическа система
- Дефиниране на логически модел
- Разработване на логически модел
- Изграждане на нова система

Съвременен структурен анализ

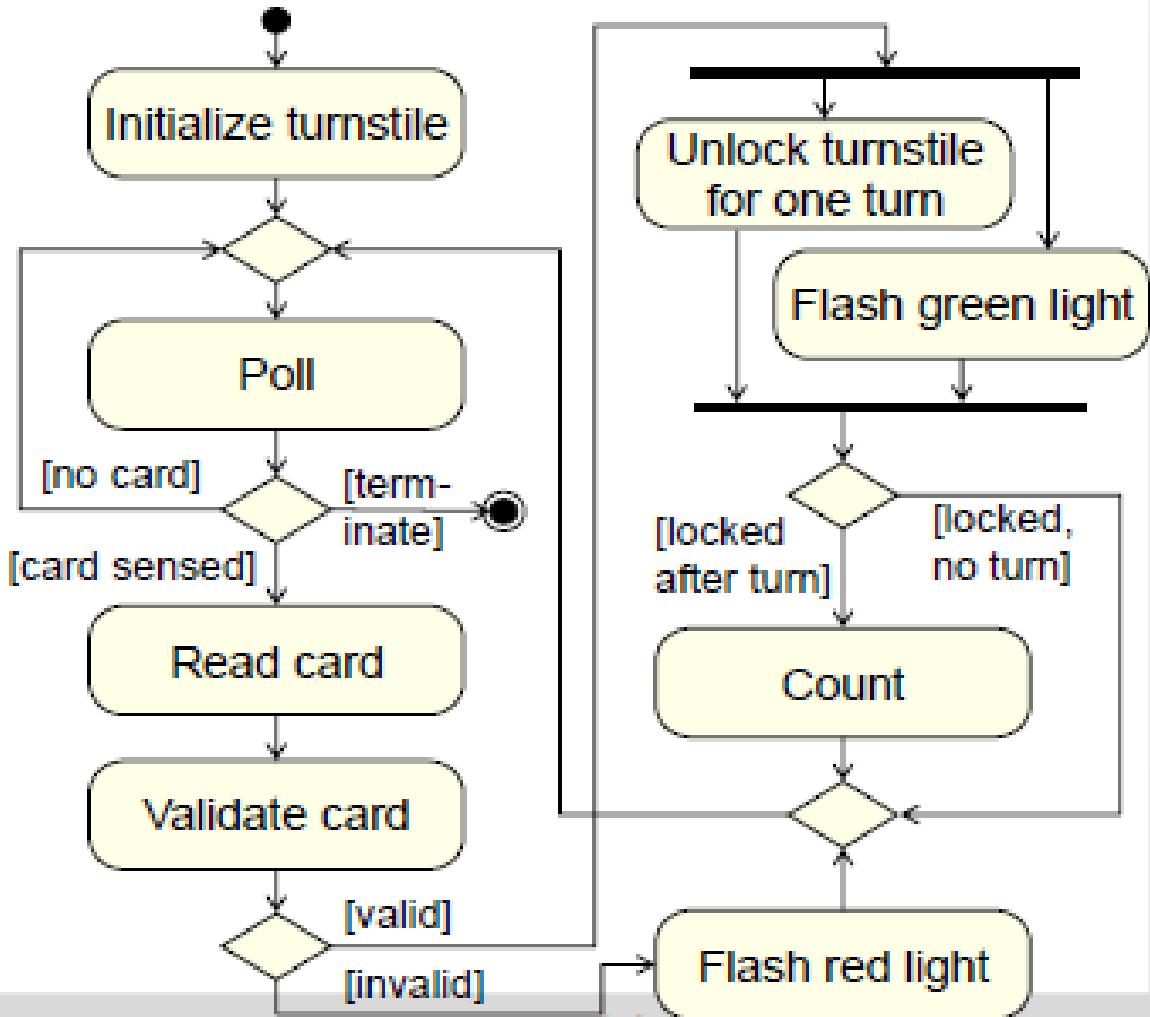
- Прави разлика между *реални* (съществени) нужди на потребителите и тези изисквания, които представляват външно поведение, което отговаря на тези нужди.
- Case tools (1990)
- Разширения за инженерни проблеми в *реално време* през 1985 г. и 1987 г. за описание на управление и поведение.
- Други подходи за структурен анализ:
 - Structured Analysis and Design Technique /Техника за структуриран анализ и дизайн (SADT) 1977
 - Структурирани системи за анализ и Методология на проектирането (SSADM), 1994

Недостатък! DFD не е ефикасен подход за сложни системи/интерфейси, защото се изиска отделна диаграма за всеки възможен вход.

2. Модел на дейностите:

Activity modelling: UML activity diagram

- Models process activities and control flow
- Can model data flow
- Semantics based on Petri nets



Документиране на изискванията от перспектива на поведението на системата

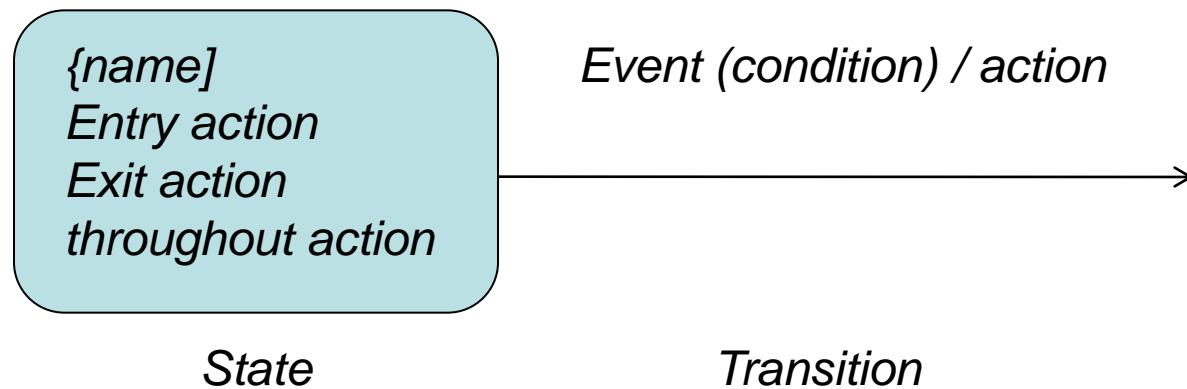
State Transition Diagram (STD)

- Показва **динамиката в поведението** на една система:
 - Как системата реагира на последователност от външни събития
 - Как независими компоненти координират работата
- Основни компоненти
 - Състояние (State)
 - Действие/събитие (Event):
 - Външно и/или вътрешно
 - Времево
 - Переходи между състоянията
- Полезно за описание на поведението на
 - системи в реално време (примери?)
 - интерактивни системи

Видове модели на поведението. Означения

Statecharts (Harel 1987)

- Улеснява дефиниране на действията, които системата ще изпълнява



- Условни преходи
- Йерархични зависимости (за сложни системи)
- Паралелност

Забележка!: Необходим е речник на означенията

3. Модели на данните

- Начин да представим логическата форма на данните
- Моделът на данните се ползва често заедно с DFD
- Видове модели на данните

3.1. Релационен модел (Codd, 1979; Date, 1990)

- Задава данните като набор от таблици с общи ключове
- Недостатъци на релационния модел
 - Неявно описание на данните
 - Неадекватното/неудобно и непълно моделиране на релациите
- Подобрение - чрез включване на информация за семантиката на данните

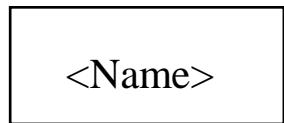
3.2 Семантичен модел

- Подходите на семантично моделиране на данни включват:
 - Entity-relationship model ERM (Chen, 1976) и вариантите му:
 - RM/ T (Codd, 1979)
 - SDM (Hammer and McLeod, 1989)
 - Hull and King, 1987
- Моделите представлят: *субектите* (entities) в база данни, техните *атрибути* и техните *връзки*.
- Използва графични нотации, софтуерни инструменти
 - Приложение на UML: същностите като обектен клас

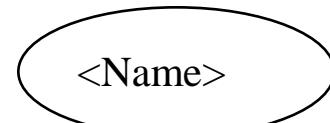
Entity Relationship Diagram (ERD) / Model

- Представя логическия вид на данните и връзките между тях
- Основни компоненти:
 - *Същност (Entity)*: обект или дейност
 - *Атрибути (Attributes)*
 - *Връзки(Relationships) - елементи:*
 - кардиналност (cardinality): (1:1), (1:n), (n:1), (n:m)
 - опционалност: задължителна или незадължителна
 - зависимост:
 - а) рекурсивна връзка
 - б) под/над тип (supertype/subtype), осъществени чрез наследствена връзка

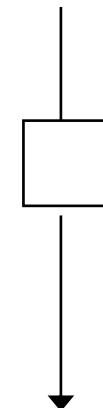
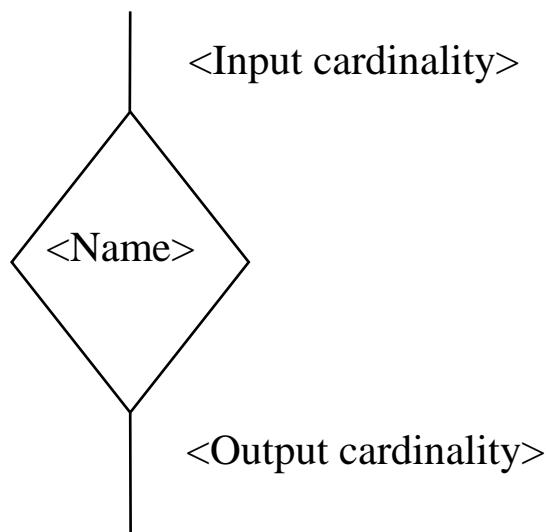
ERM notation



An Entity

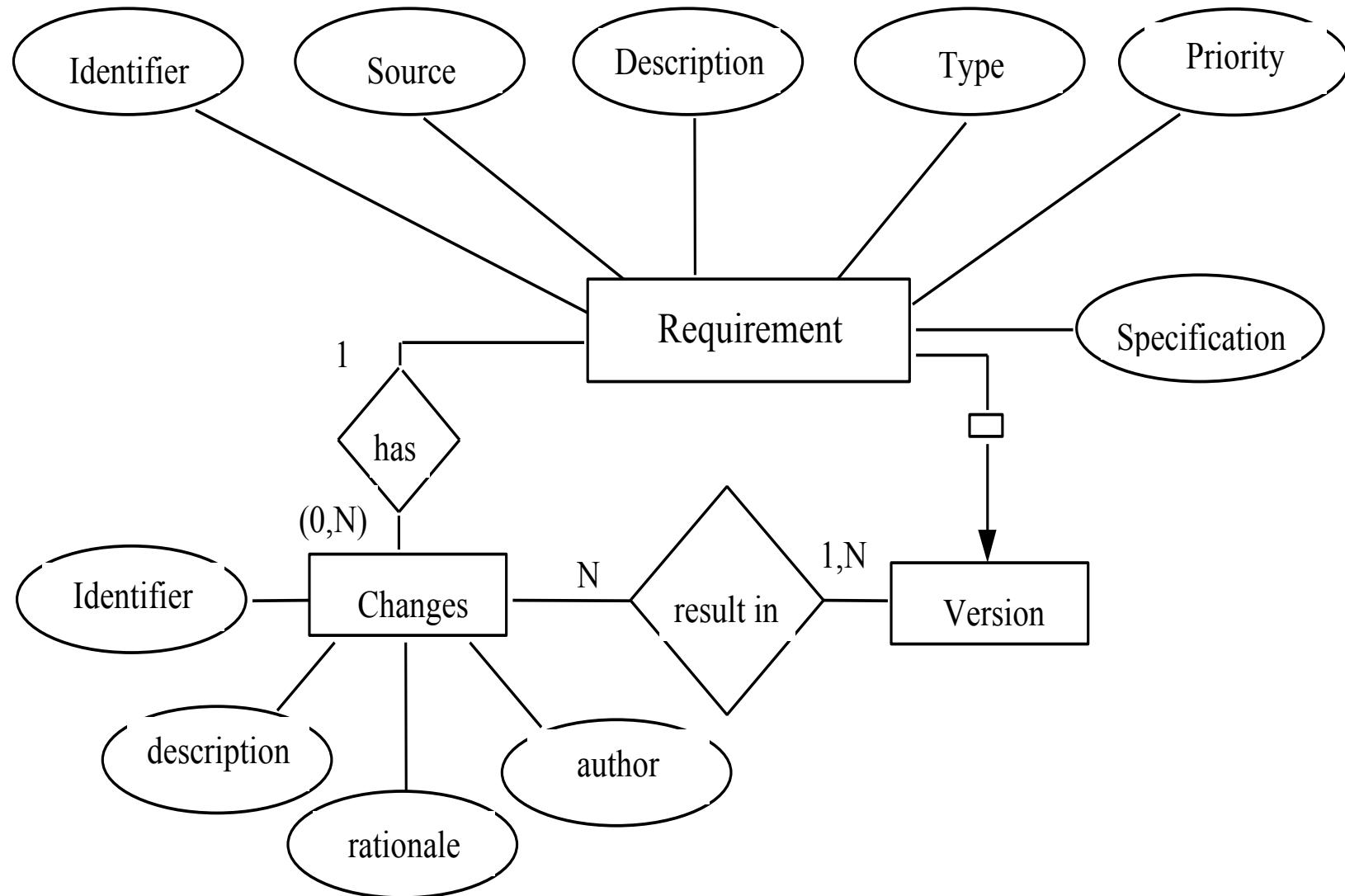


An entity/relation attribute



An inheritance relation

Пример: ERD на променящо се (evolving) изискване



3.3 Обектни модели (ОМ)

Обектно-ориентиран подход за моделиране

- ERM е най-близкият предшественик на **обектно-ориентирания модел**
- ОМ представят *едновременно* данните и тяхна обработка, както и класификация и връзки на субектите на системата
- Основната идея е идеята за **ОБЕКТ**
- Обектно-ориентираният анализ, проектиране и разработка е общоприет.

История:

- Shlaer and Mellor (1988) – first published material
- Colbert (1989)
- Coad and Yourdon (1990)
- Wirf-Brock (1990)
- **Rumbaugh** (1991)
- **Jacobson** (1992)
- Martin-Odell (1992)
- **Booch**, 1994

Забел.! Означенията за различните методи са семантично сходни.

- **UML (1999) е обобщаващ метод и ефективен стандарт за ОМ**

Основни концепции на обектно-ориентираното моделиране

- **Обекти и класове (Objects and classes)**
- **Капсулиране (Encapsulation) на данни за обект**
- **Наследство (Inheritance)**
- **Съобщения (Messages)**
- **Методи (Methods)**

Обектно-ориентираната парадигма се използва последователно от анализа до реализацията за целия процес на софтуерното инженерство.

Но! Понякога крайните потребители го намират за неразбираем и предпочитат и функционален модел като DFD.

Обектен модел: елементи

Обект:

Нешто реално или абстрактно, за което запазваме данни и описваме операции за манипулиране на тези данни.

- Обектите са концепции за *капсулиране, клас и наследство*
- Обектите поддържат абстракцията на компонент чрез разлагането на класовете на отделни обекти.

Обекти

- Типични обекти:
 - Физически места, сгради ...
 - Устройствата, с които системата взаимодейства
 - Системите, които си взаимодействат с проучваната система
 - Организационни единици
 - Нещо, което трябва да бъде запомнено с течение на времето
 - Специфични роли на хората, които използват системата.

примери: сензор, кола, акаунт, софтуерен дизайн, отдел, ...

- Обектите са *същности с атрибути на данните и операции* на обектния клас
- **Обектите може да бъдат композитни.**

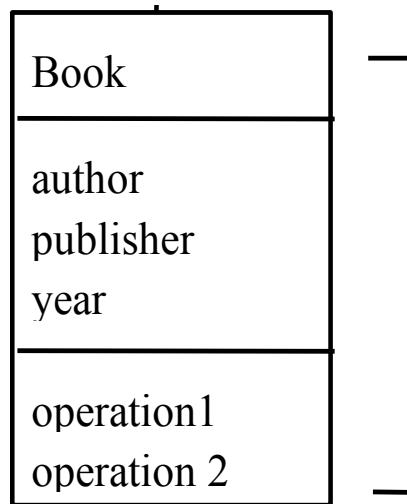
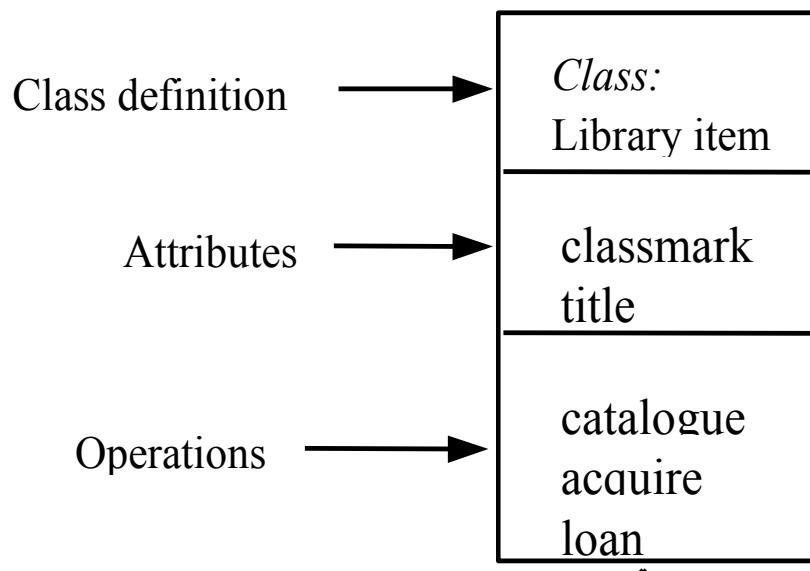
(Обектен) Клас – основен елемент на ОМ

- Описание на множество от обекти, които споделят общи атрибути и операции (услуги)
 - Обектът е *инстанция* от класа
- Класът реализира обекти от даден тип
 - Пример: The object type *Bank Customer* refers to a class of bank customers.
If “John Smith” is a bank customer, then **bank customer** is the *class* and “John Smith” is an *instance* of the bank customer.
- Трудности при идентифицирането

Операции и методи

- **Операция е действие за четене и манипулиране данните на обект.**
 - Отнасят се *само* до структурите на данните от този тип обект.
 - За достъп до структурите за данни на друг обект, обектите трябва да изпращат *съобщения* до този обект.
- **Метод е реализацията на дадена операция** – специфицира се пътя, по който операциите се кодират в софтуера.

Концепция за обектно-ориентиран подход



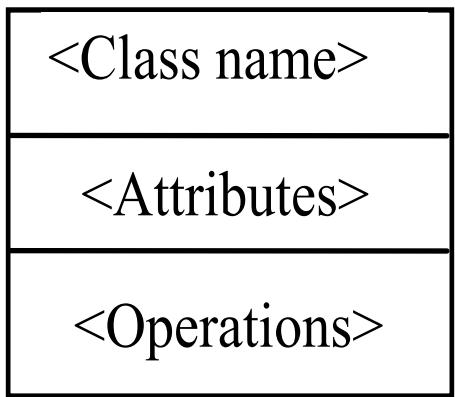
Капсулиране

- **Заедно пакетиране на данни и операции**, което
- **Предпазва** данните на обекта от **неоторизиран достъп**
- Детайли за това, как операциите се реализират остават **скрити** от клиентите.

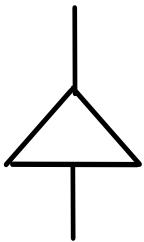
Наследство

- Таксономия на класовете
- Обекти на по-ниско ниво в **йерархията** на класа наследяват атрибутите и операциите на техните родители.
- Обектите са в състояние да включат данни и/или операции, които са специфични за тях.
- Наследяването на данни от повече от един родител се нарича **множествено наследяване**. Наследените атрибути и услуги са **конюнкция** на родителските.

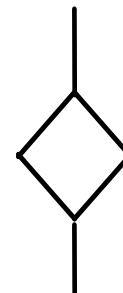
Object-oriented notation



(i) Class

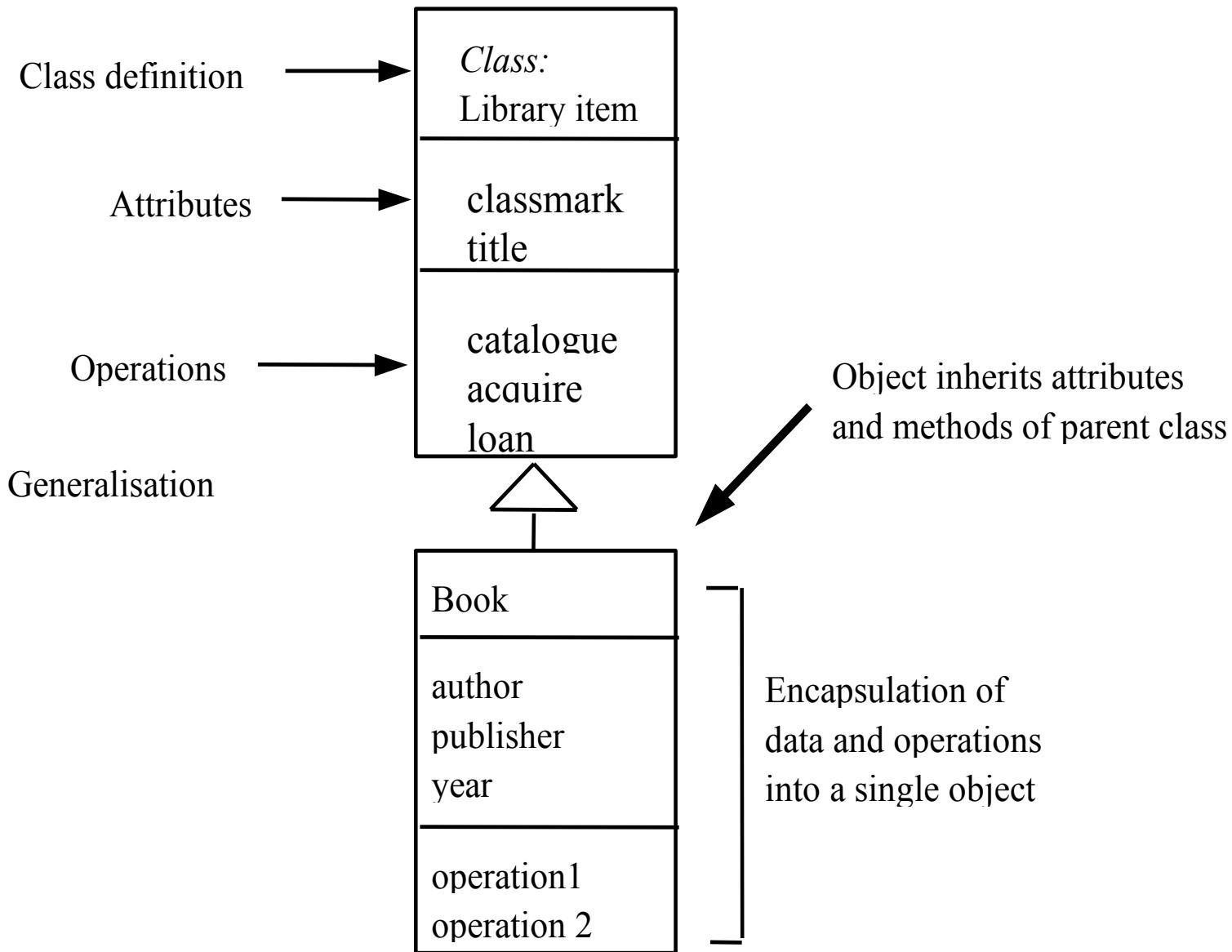


(ii) Generalisation



(iii) Aggregation

Концепция за обектно-ориентиран подход - пример



Агрегиране (Object aggregation)

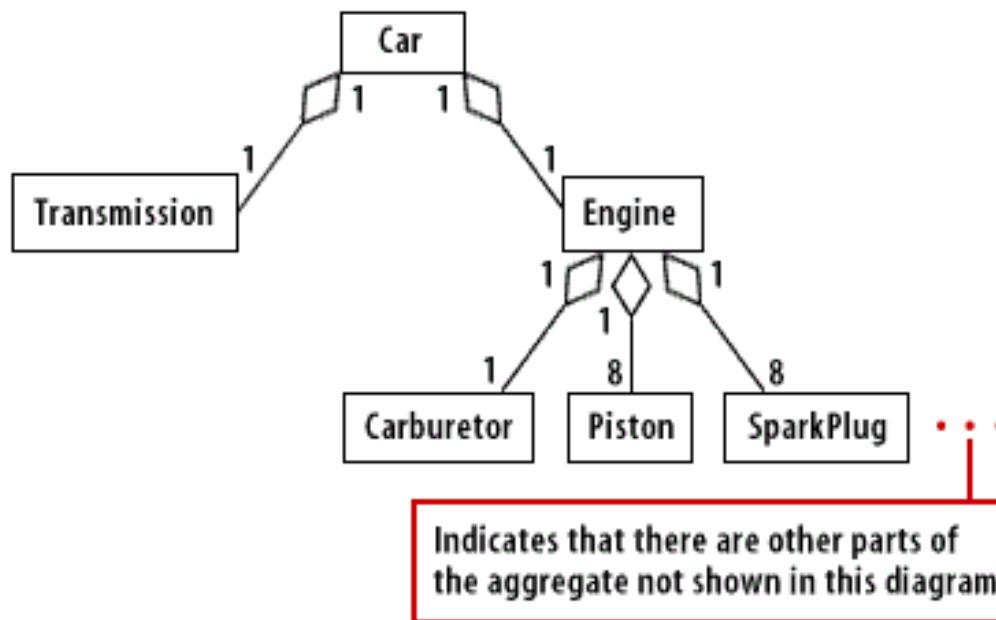
Обекти са съставени от други обекти

- Пример 1:

„A study pack (of a course) is composed of one or more assignments, slide packages, lection notes and video types.”

- Пример 2:

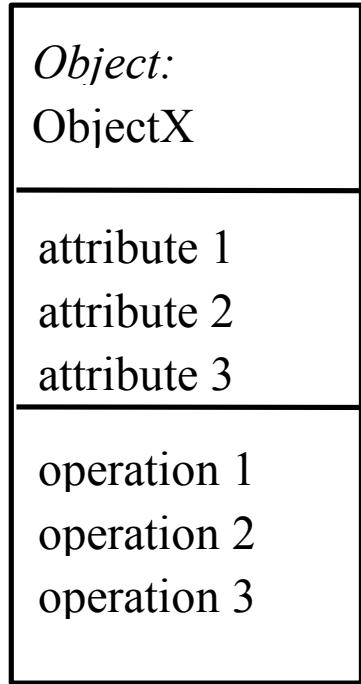
This example shows multiple levels of aggregation within the same object.



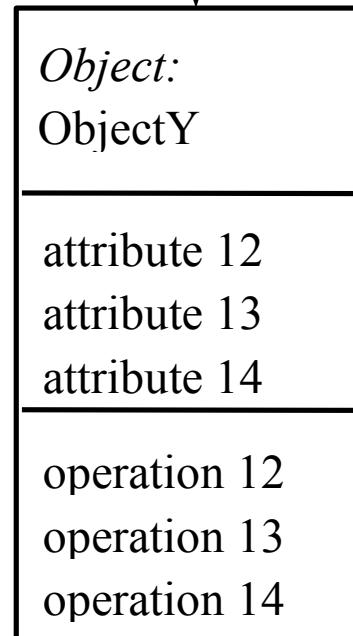
Съобщения

- Обектите **комуникират** чрез изпращане на съобщения
- Съобщенията обхватват:
 - Името на обекта получател
 - Операцията, която трябва да се извика
 - Опционално - множество от параметри
- Когато обект получи съобщение, то той **извиква операция**
- Операцията **изпълнява** подходящ метод

Message passing



Message 1:
to:ObjectY
operation: 12
parameters: a.b



Стъпки в обектно-ориентирания метод за моделиране

Повечето методи, базирани на обектно-ориентиран модел споделят общи/еднакви стъпки на анализ:

- 1) Идентифициране на основните обекти
- 2) Конструиране на структури на обектите, определяйки асоциациите на обекти в класовете
- 3) Дефиниране на атриутите, свързани с всеки обект,
- 4) Определяне на съответните операции за всеки обект
- 5) Дефиниране на съобщенията, които могат да бъдат предавани между обектите

Notes!: Object class identification is recognised as a difficult process requiring a deep understanding of the application domain.

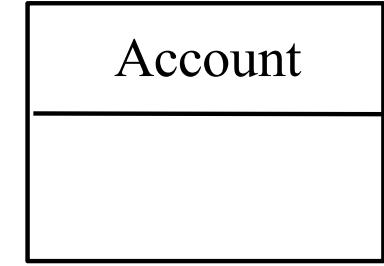
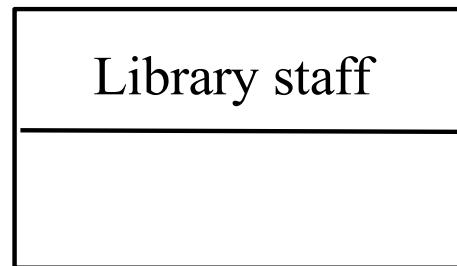
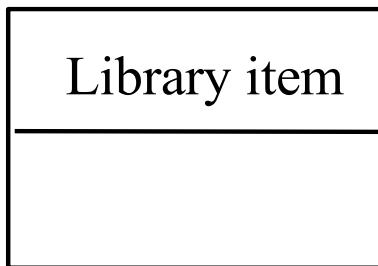
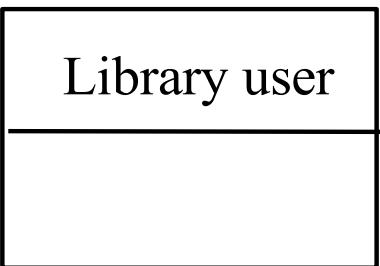
Object classes reflecting domain entities are reusable across systems.

Пример за обектно моделиране: Library example

Изисквания:

- A library system is intended to provide its **users** with the ability to automate the process of:
 - Acquiring library items
 - Cataloguing library items
 - Browsing library items
 - Loaning library items
- **Library items** comprise published and recorded material
- The system will be administered by a member of the **library staff**
- **Users** must *register* with the system administrator before they can borrow library items

Step 1 - Initial classes identified



Library example (cont.)

Изисквания:

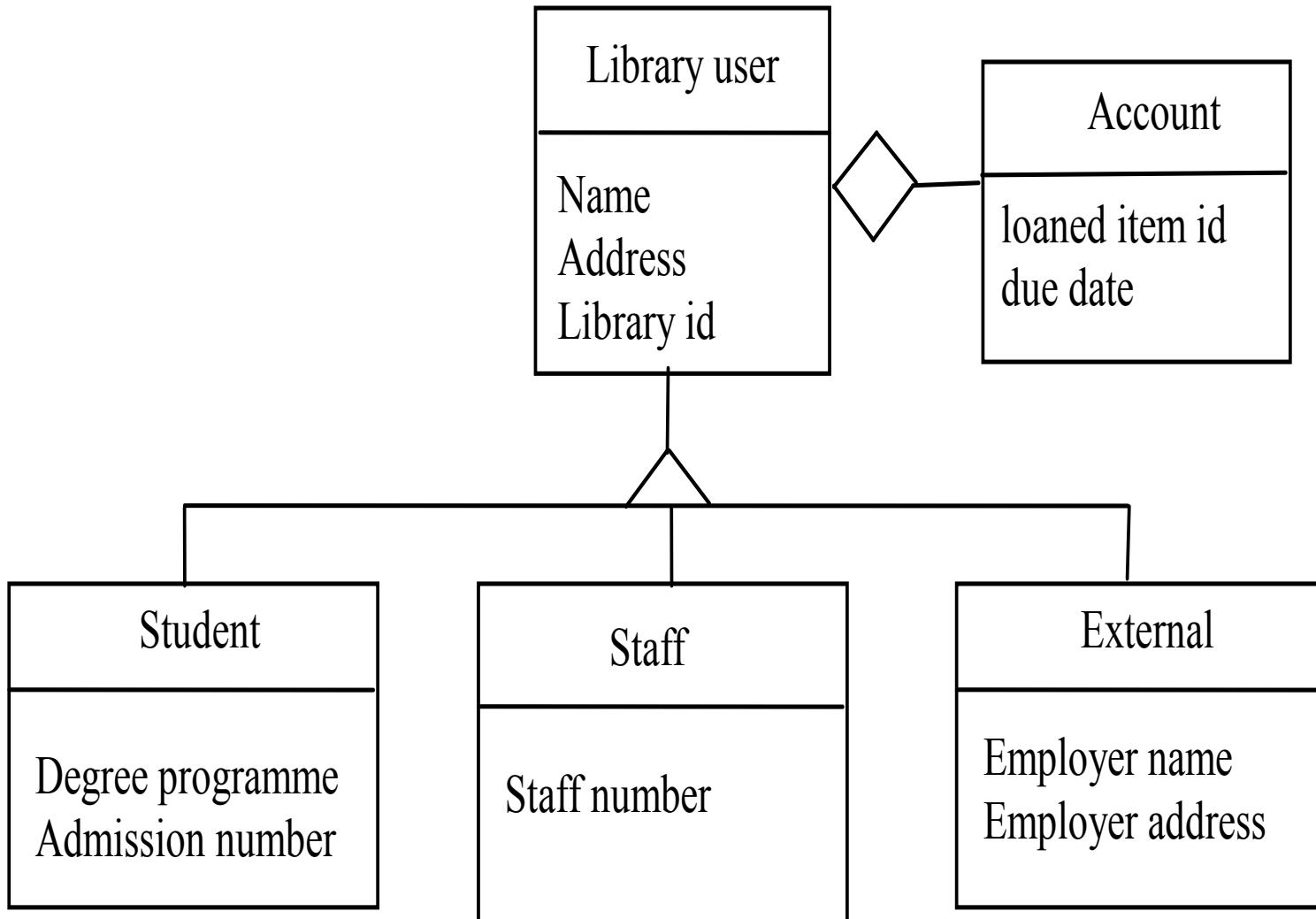
- **Library users** are drawn from three primary groups:
Students, Members of staff and External users
- **All library users** have as part of their registration:
Name, Library number, Address, Account
- **In addition** the following information also required for registration:
Students - Degree programme and admission number
Staff - Staff number
External users - Employer details

Step 3 - Identifying the attributes

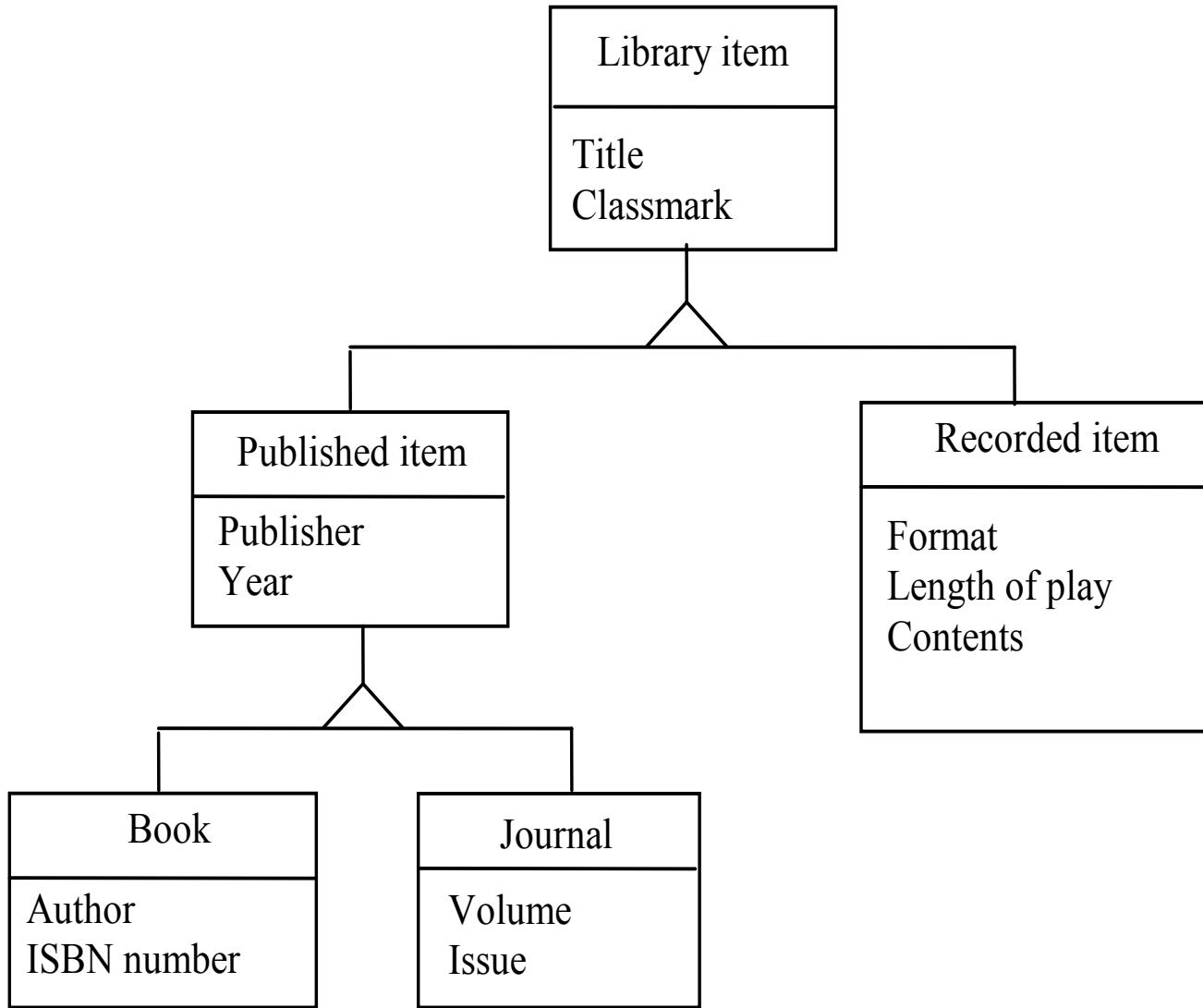
Attributes can be revealed by analysis of the system requirements

- For example, it is a requirement that **all library users must be registered** before they can use the library
 - This means that we need to keep registration data about library users
 - Library users may also be provided with an **account** to keep track of the items loaned to them
- Library item has the attributes; title, description and classmark
- The library user class has the attributes; name, address and library ID

Step 2, 3 - Inheritance for Library user



Step 2,3 - Inheritance for library item



Step 4 - Relationships between classes

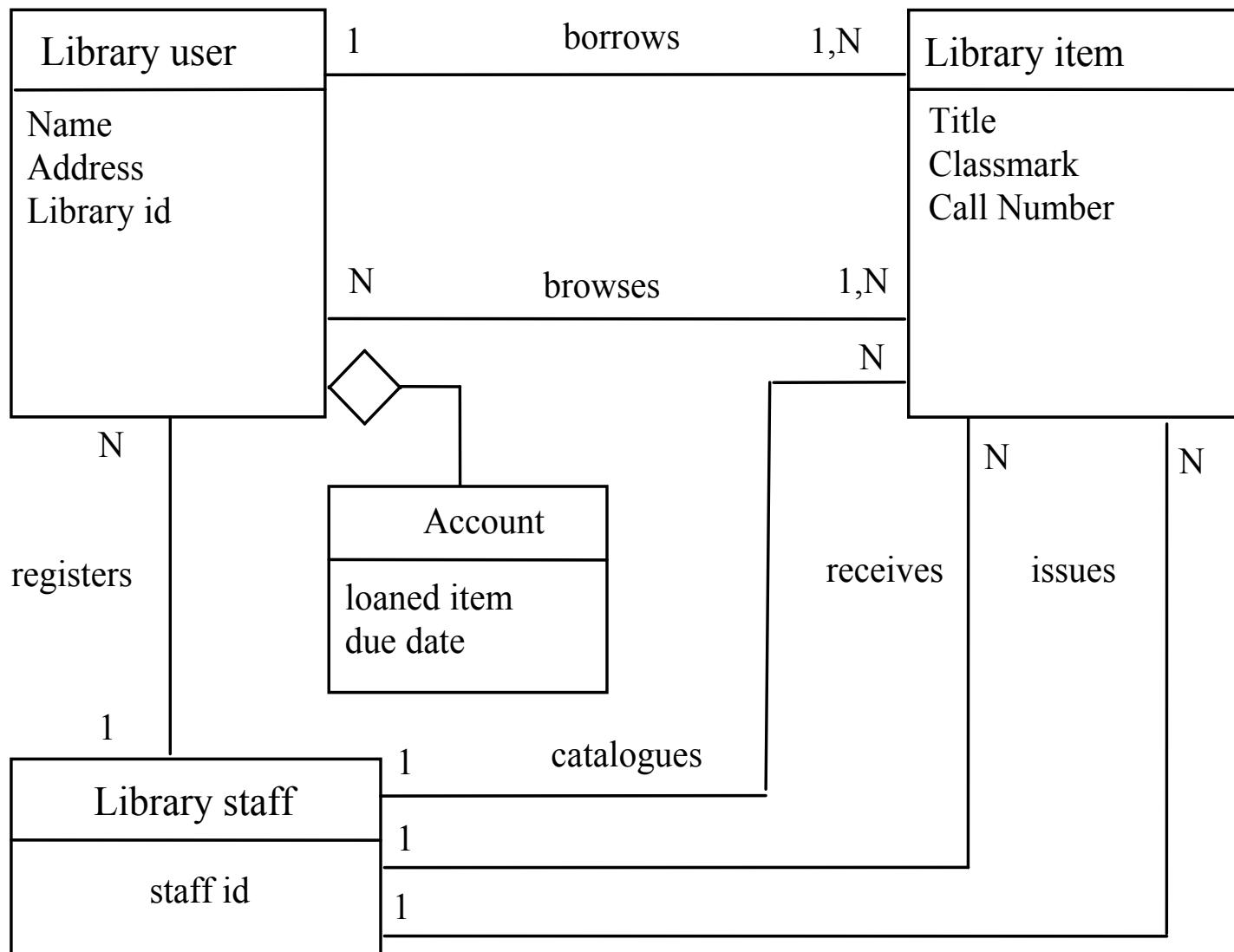
Изисквания:

We can identify the following relationships from the partial requirements:

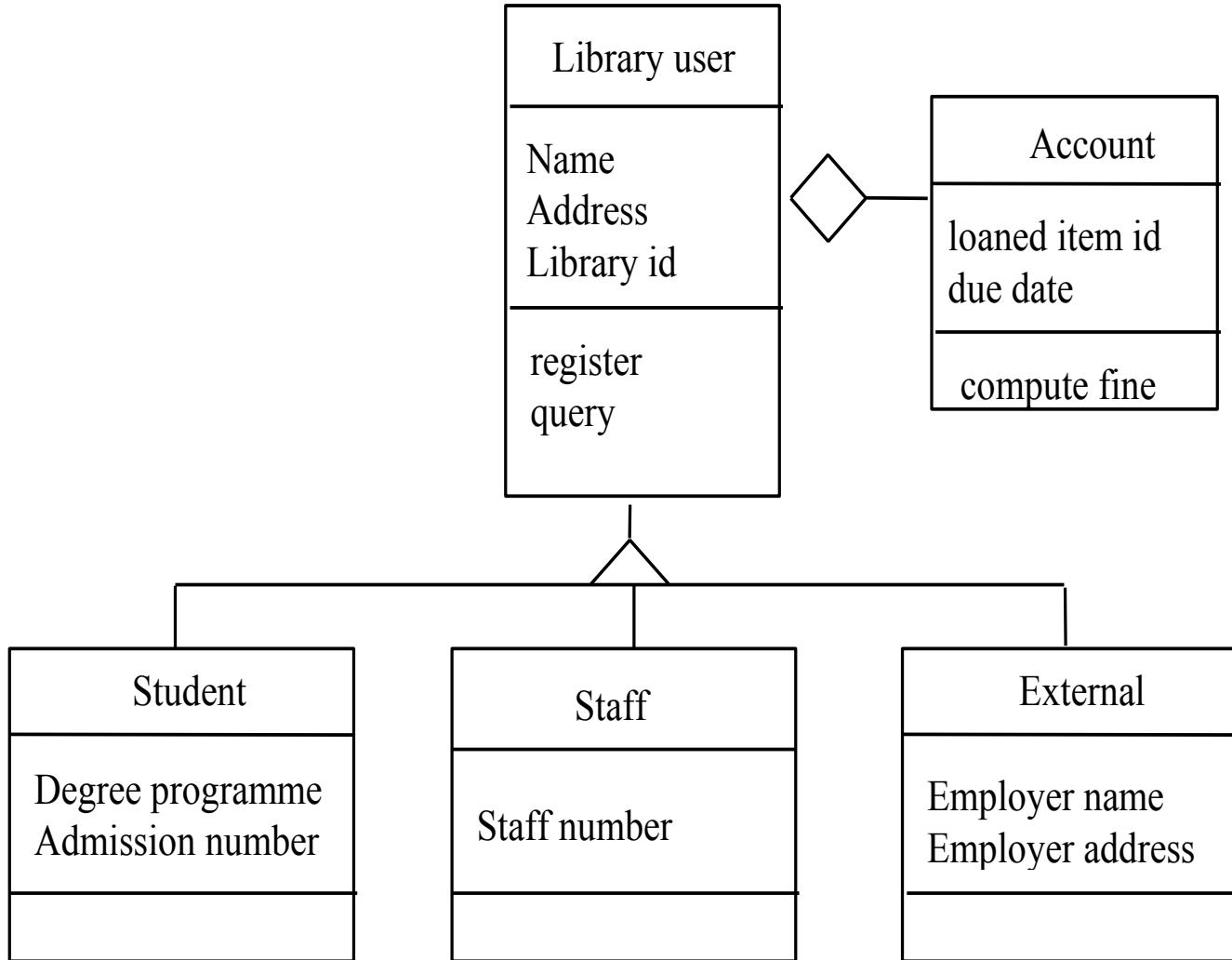
- (i) A library user **borrow**s a library item
- (ii) A library item **is recorded** or **published**
- (iii) The system administrator **registers** the library user
- (iv) Library users **are** students, staff and external users
- (v) The system administrator **catalogues** the library items
- (vi) The library assistant **issues** the library items

....

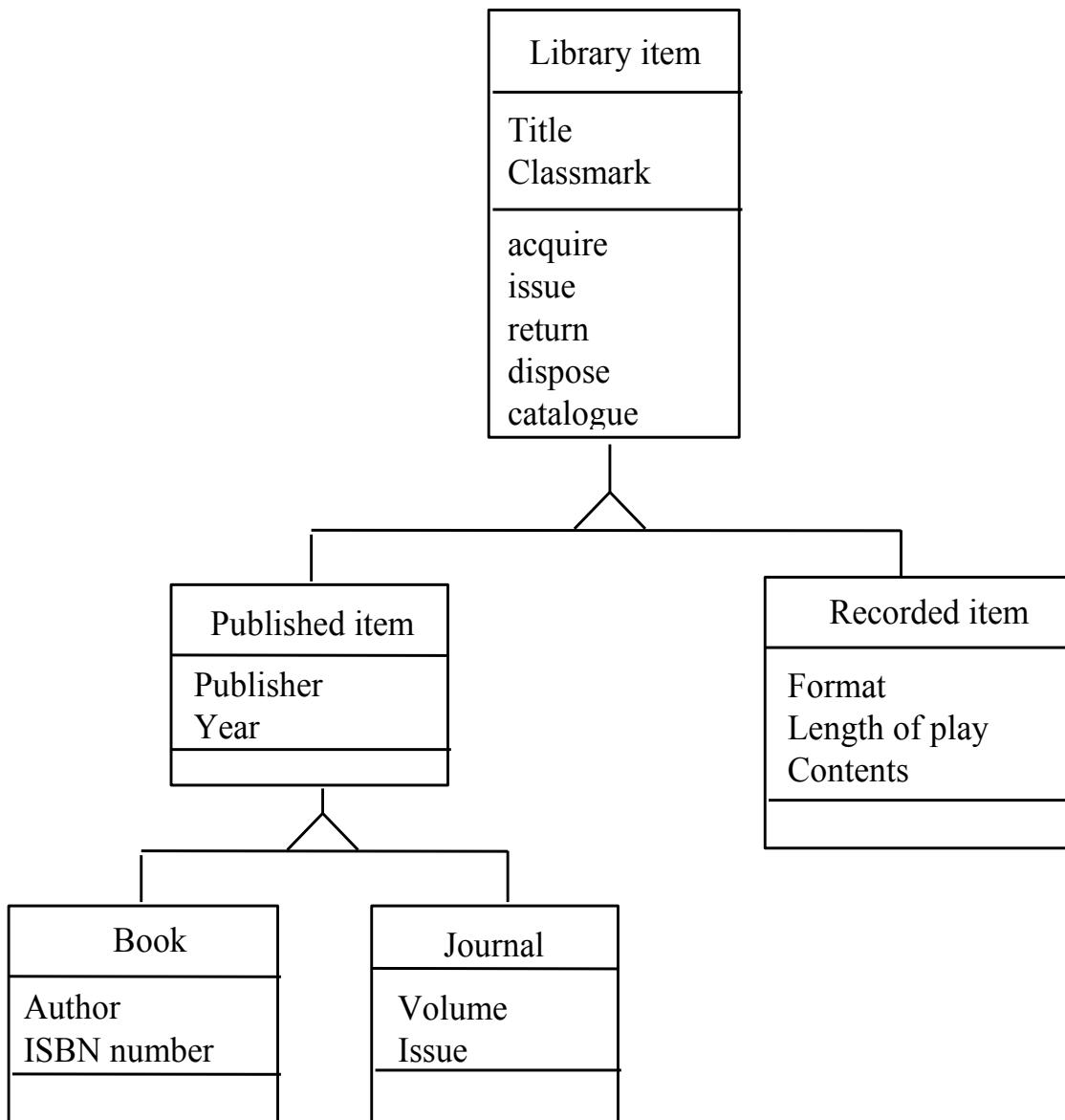
Basic object operations



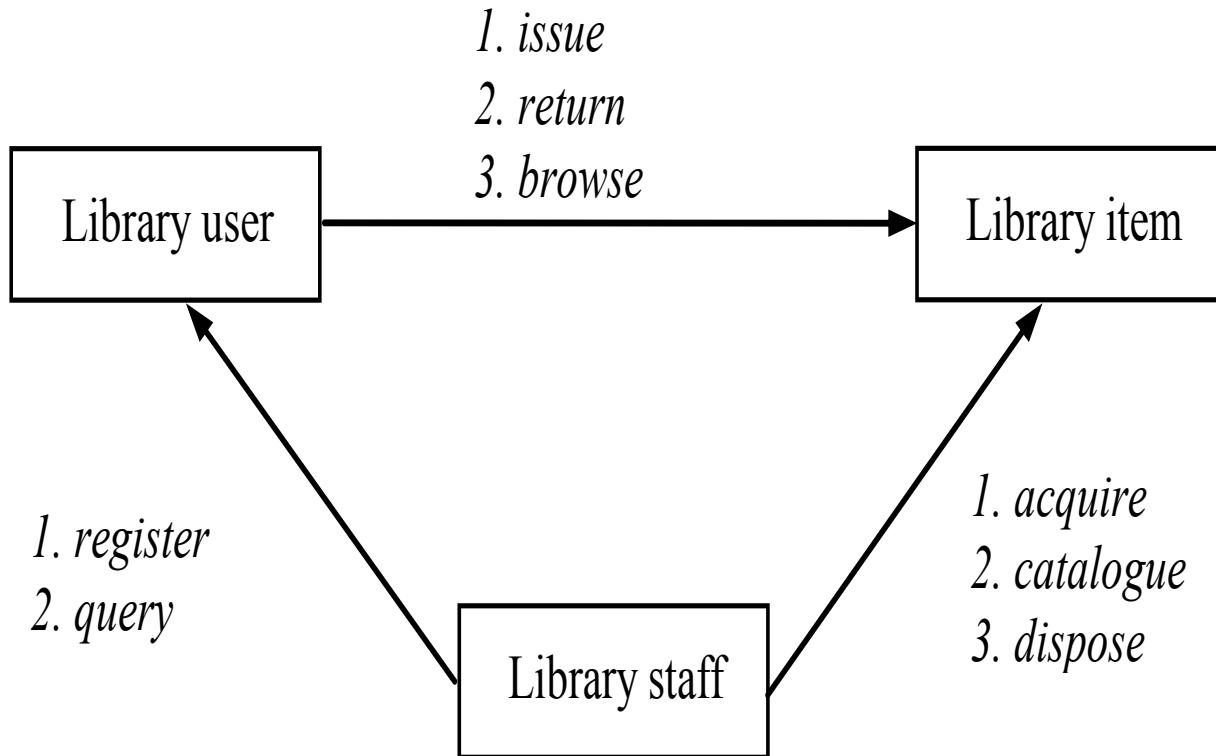
Step 4 - Operations for library user and library staff



Step 4 - Operations for library item



Step 5 - Messages between objects



Речник на данни

Речникът на данните е **организиран списък** на всички елементи, отнасящи се до системата с точни строги дефиниции, за да се има общо разбиране за всички елементи на системата.

(например) речникът описва:

- примитивни елементи на данните като *име, размер, обхват*
- *значение и обхват* на потоците от данни, на хранилища за данни
- подробности за *връзките*, които са дефинирани в моделите

Речник на данни: предимства

- Начин да осигурим *консистентност* на спецификацията.
- Механизъм за *управление/организиране на информация* чрез имена (на обекти, елементи, състояния, действия,...).
- Запазва *организационна информация*, която може да свързва анализа, проектирането, разработването, използването, поддържането и разширяването.

Речник на данни: примери

1. customer-name = title + first-name +
(middle-name) + last-name

title = [Mr. | Miss | Ms. | Mrs.| Dr. | Prof.]

first-name = {legal-character}

order = customer-name + shipping-addr
+1:10 {item}
* comments*

2. Детайлно описание на състоянията на системата

Формални методи за описание на софтуерна система- класификация

- Категоризация в спектъра на формалното описание (“formality” spectrum) (Pressman, 1992)
- Полуформални методи (Semi-formal and informal methods)
 - Use natural language, Diagrams, Tables and simple notation
 - Structured analysis; Object-oriented analysis
- Формалните методи включват:
 - Математически синтаксис и семантика
 - Примери: Z, B, VDM, LOTOS, CSP ...

ФМ за целите на разработването на софтуер

- Осигуряват високо ниво на точност на системата, която да покрие напълно спецификацията, но

Зашо!

Не могат да гарантират абсолютна точност!

- Не могат да решат всички въпроси и проблеми на анализа и разработката:

Разкриват само/част от функционалните изисквания.

Отнася се до всички модели - формални и полуформални.

Зашо!

Предимства на ФМ за целите на разработването на софтуер

Формалните методи са ниво на абстракция, която позволява да се създаде качествен (удобен, разбираем, надежден) софтуер.

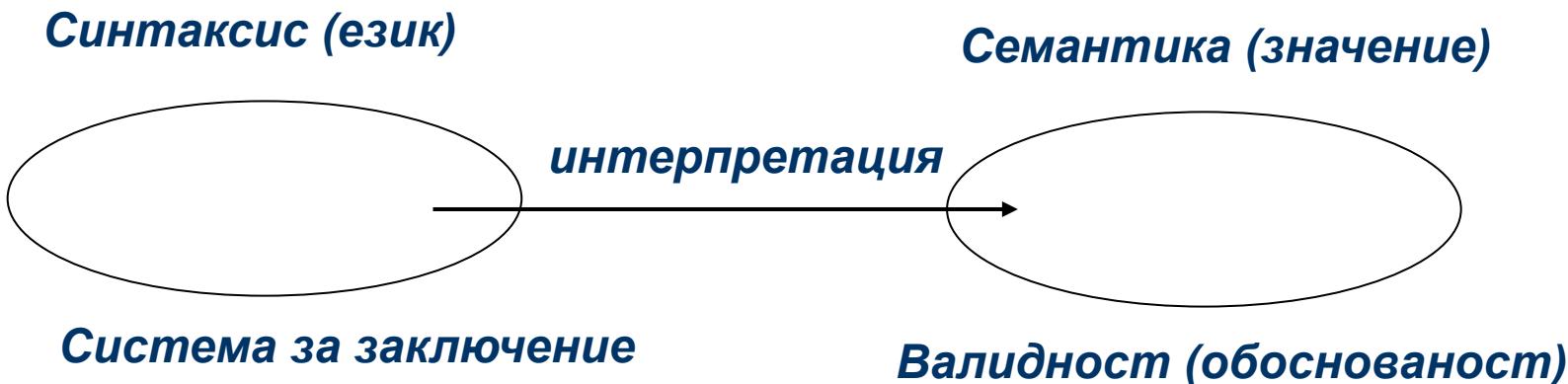
- Намаляват неточността
- Осигуряват строгост в описанията на ранни етапи от разработката
- Могат да проверят/гарантират коректността и точността, непълнотата и неконсистентността на спецификацията.

Разпространение на ФМ

- ФМ не са широко разпространени
(FM are slowly emerging from academic research and now being applied in industrial software developments where *safety or security is critical*.)
- Причини:
 - Трудности да бъде разбрана спецификацията
 - Трудна формализация на отделни аспекти на системата
 - Трудна възвръщаемост на инвестицията +
+ (консерватизъм и нежелание).

Компоненти на формалната спецификация

- **Синтаксис (Syntax):** специфична нотация за представяне на спецификацията. Базира се на синтаксиса на теорията на множествата и предикатните изчисления.
- **Семантика (Semantics):** дефинира съвкупността от обекти, които ще се използват, за да се опише системата.
- **Връзки (Relations):** дефинират правилата, които показват кои обекти правилно удовлетворяват спецификацията.



Формални методи - примери

При формалните методи информацията е структурирана и представена на подходящо ниво на абстракция и се ползват при:

- > проектиране,
- > разработване,
- > тестване,
- > експлоатация и поддръжка,
- > трансформиране на софтуера.
- > документация/спецификация

Примери:

- А) **CICS** (*Customer Information Control System*) е фамилия от продукти за банкови транзакции, произведена от IBM UK Laboratories at Hursley Park. Осигурява достъп до данни, комуникация, интегритет, сигурност т.е. за управление на информация...
- Б) В **method** – Meteor line 14 на Парижкото метро. 110000 реда на B model, от които се генерира 86000 реда код на Ada без грешка. (Oct. 1998).)

Z нотация: въведение

Една формална спецификация трябва да може да представи голямо количество проза. Тя съпоставя математическите обекти към особеностите на проектираната система: състояния на системата, структури от данни, техни свойства и операции с тях.

Как се постига ясна и точна спецификация?

Съществуват два елемента, които изграждат формалната спецификация на **Z нотацията**:

– **математическият:** теорията на множествата и математическата логика;

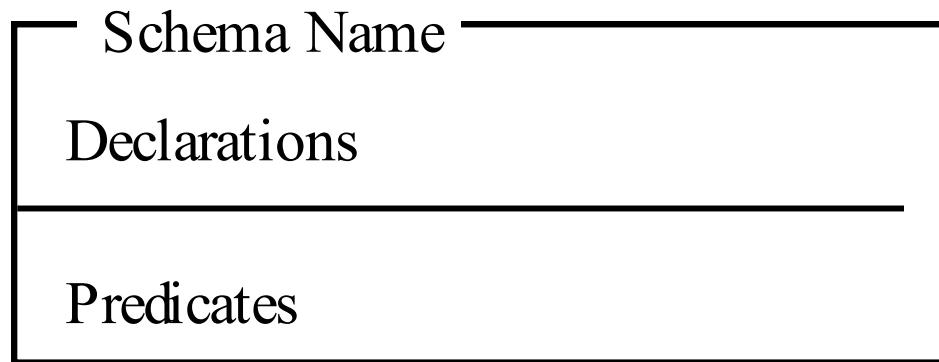
- **езикът на схемите** се използва за *структурни и композиционни* описания: събиране на части от информацията, формулиране на общи описания и доказателства, необходими при следващо приложение.

Z нотация: Схеми

- използва **Теория на множествата и логиката**
 - множества, релации, функции
 - предикатна логика от първи ред
- **Схемите** са именувани записи, които съдържат:
 - име
 - декларация
 - предикат
- **Изчисления със схеми (Schema calculus)**
 - Математически оператори, чрез които се изграждат по-големи схеми чрез по-малки
 - **Z описанието се представя като колекция от схеми**
- **Синтактични събирания (Syntactic conventions)**
 - за да се направят описанията лесни за възприемане;
 - за да улесним програмирането.

Z schema

Z схемата обхваща 3 части: Име, Декларация и Предикат



*Декларациите на схемата съдържат имената и техния тип във вида:
идентификатор: тип*

*Предикатите на схемата дефинират връзките между обектите в
декларацията, за да представят специфични свойства.*

Използване на Z схема

- За описание на:
 - *Тип*
 - *Състояние*: Да опише състояние на системата, декларирайки променливите и предикатите за инвариантните характеристики на това състояние.
 - *Операция*: Декларацията съдържа компонентите на изходното състояние, компонентите на крайното състояние, входната и изходната информация за операцията. Предикатната част описва релацията между входната и изходната информация и началните и крайните състояния.

Пример на описание със Z нотация: Library example

- **Basic sets** for this system are [Book, Borrower]
- The **state space** of the lending library can be defined using the following schema:

```
Library
stock: P Book
onLoan: Book → Borrower

dom onLoan ⊆ stock
```

Schema for borrow operation

Borrow

$\Delta \text{Library}$

$\text{book?} : \text{Book}$

$\text{reader?} : \text{Borrower}$

$\text{book?} \in \text{stock}$

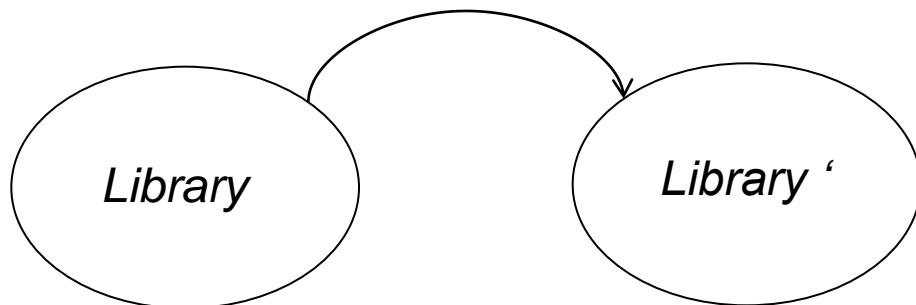
$\text{book?} \notin \text{dom onLoan}$

$\text{onLoan}' = \text{onLoan} \cup \{\text{(book?, reader?)}\}$

$\text{stock}' = \text{stock}$

$book?$

$reader?$



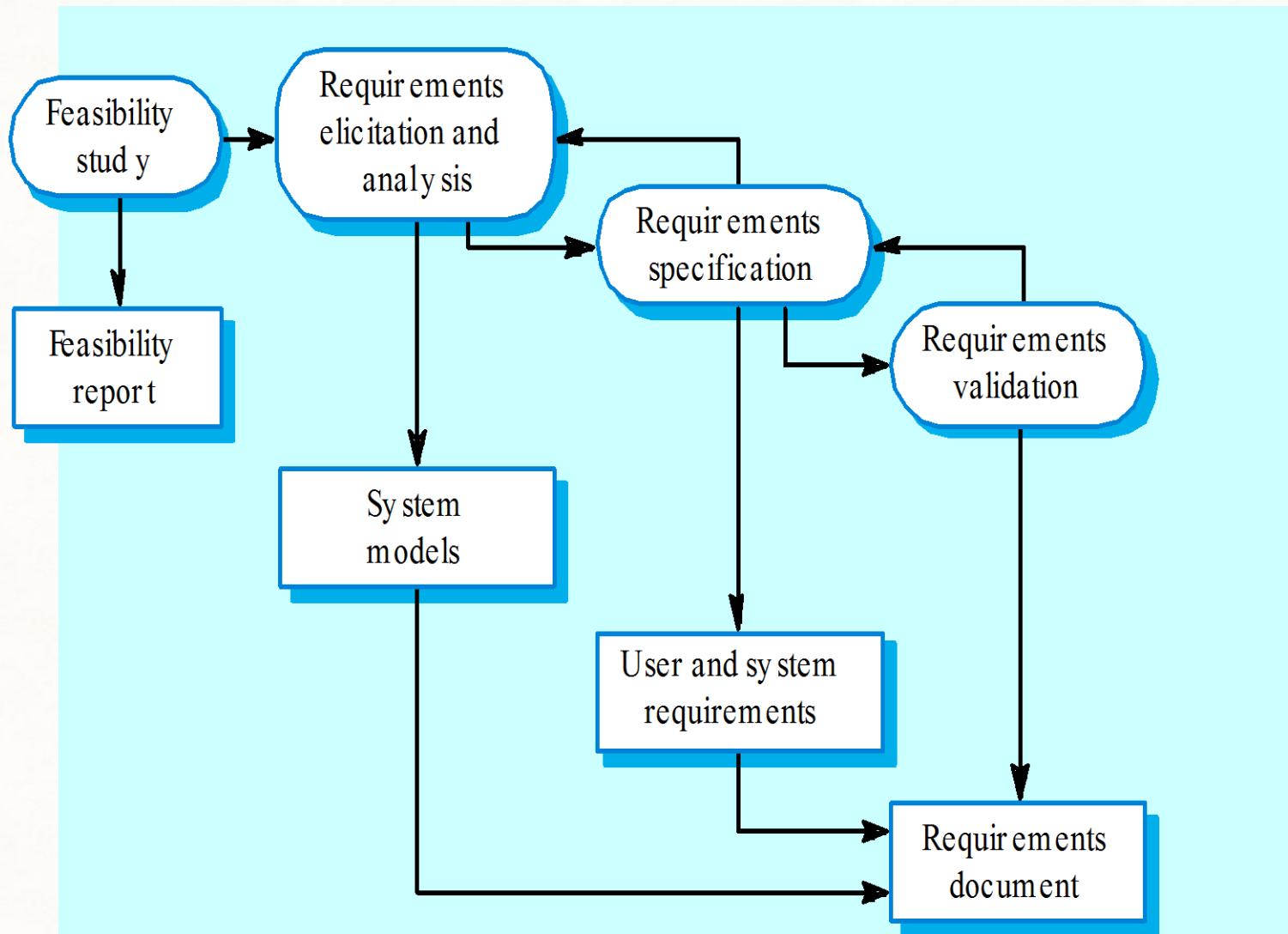
Валидиране на изискванията

Лекция 7

Съдържание

1. Валидиране на изискванията – същност и особености
2. Техники за валидиране на изискванията:
 - Преглед/рецензия на изискванията
 - Прототипиране
 - Валидиране на модела
 - Тестване изискванията

Процес на инженеринг на изискванията



Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
 - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Sommerville

Анализ и валидиране – общото и различното

- **Анализът** работи със сировите изисквания, директно извлечени от заинтересованите лица.
 - “Дефинирахме ли правилните изисквания /Have we got the right requirements?”
- **Валидирането** работи с краиния чернови вариант на формулираните изисквания т.е. с вече договаряни и съгласувани изисквания.
 - “Дефинирахме ли правилно изискванията / Have we got the requirements right?”

Валидиране - същност

to validate - 1. потвърждавам, узаконявам, ратифицирам

- Удостоверява, че документът, описващ изискванията, е приемливо описание на системата.
- Документът на изискванията се проверява за:
 - Пълнота и последователност (completeness and consistency);
 - Конфликт в изискванията;
 - Неясни, двусмислени изисквания;
 - Съответствие със стандартите;
 - Грешки в моделите на системата;
 - Технически грешки
 - ...
- Участници в процеса на валидиране

Основен проблем на валидирането на изискванията

- *Не съществува документ, който да е база за сравнение за оценка на коректността;*
- Критерият е: 1) ясно описание и
- 2) пълен отговор на всички изисквания на заинтересованите лица.

Валидиране на изискванията – основни проверки

- **Валидност:**

Does the system provide the functions which best support the customer's needs?

- **Консистентност:**

Are there any requirements conflicts?

- **Пълнота:**

Are all functions required by the customer included?

- **Реалистичност:**

Can the requirements be implemented given available budget and technology

- **Проверимост:**

Can the requirements be checked?

Входна и изходна информация за процеса на валидране



Входна информация за процеса на валидиране

- **Документ на изискванията**
 - Би трябвало да е цялостна завършена версия на документа. Изготвянето и форматът на документа е съобразен със стандартите на организацията.
- **Знание за организацията**
 - Знание (често което се подразбира) за организацията – използвана терминология, умения на работещите, култура, структура ...
- **Организационни стандарти**
 - Трябва да бъде проверено спазването на локални стандарти.

Изходна информация за процеса на валидиране

- **Списък на проблемите**, открити в описаните изисквания:
 - *Идеален вариант*: класификация на проблемите по тип.
- **Действия в отговор на съгласуваните изисквания**

Списък от действия, които да отговорят на откритите проблеми в дефинираните изисквания.

Забел.: Не е задължително 1:1 съответствие между проблеми и действия!

Техники за валидиране на изискванията

- **Преглед на изискванията (Requirements reviews)**
 - Систематичен ръчен анализ на изискванията.
- **Прототип**
 - Използване на изпълним модел на системата за проверка на изискванията.
- **Валидиране на модели**
 - Проверка за точността на моделите, представени в спецификацията
- **Генериране на тестови случаи**
 - Разработване на тестове за изискванията (to check testability).

Техники за валидиране на изискванията - 1

1. Преразглеждане / преглед / рецензиране (review) на изискванията

- **Най-често използваната техника** за валидиране на изискванията:

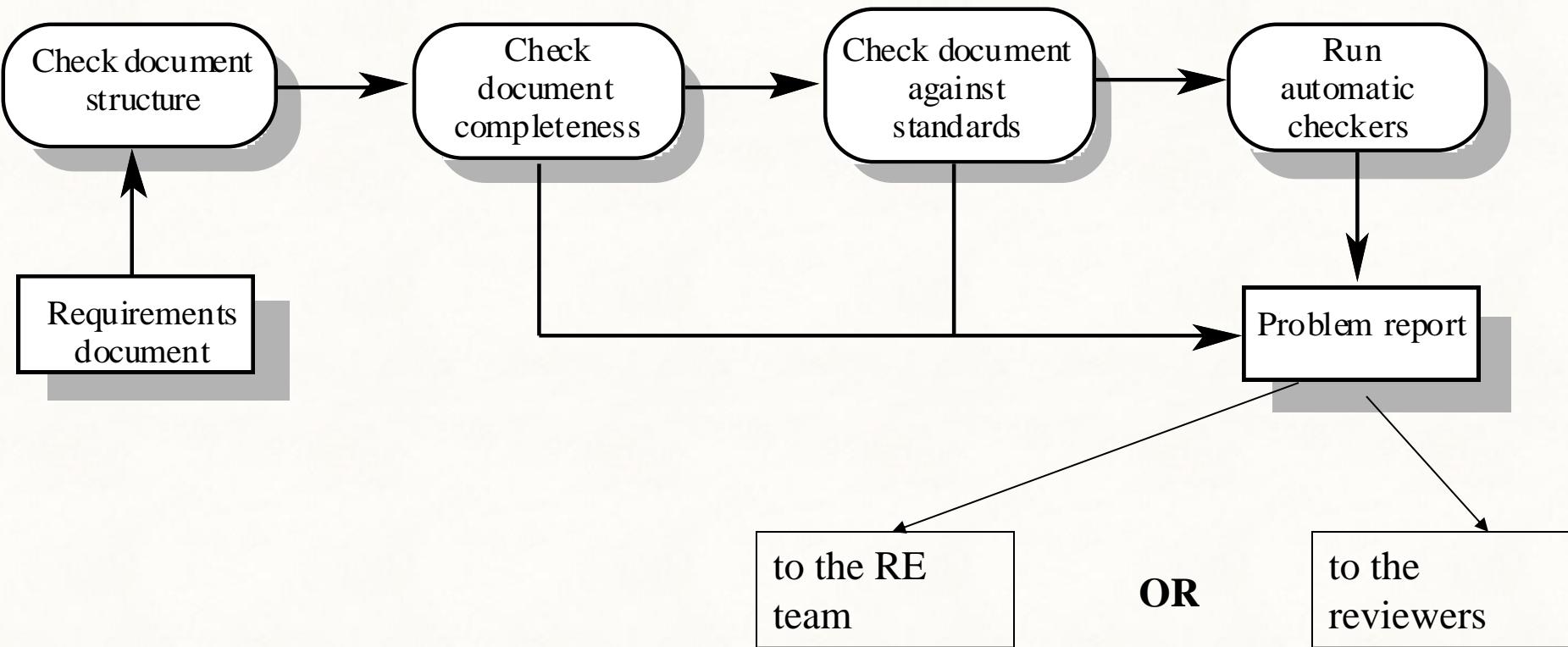
Формира се специална група, която да се

- 1) запознае с изискванията и да ги анализира;
- 2) формулира забелязани проблеми,
- 3) дискутира възможните действия за справяне с проблемите.

Предварителен преглед

- Разходът за прегледа на изискванията би могъл да бъде намален, ако се направи предварителна проверка от един човек. Търсят се явни пропуски - пълнота на документа, съгласуваност със стандартите, печатни грешки ...
 - Автоматични проверки
 - Техническо оформление
- Документът може да бъде върнат за коригиране или списък от проблеми да бъде разпространен между участниците в екипа.
- Кой осъществява предварителния преглед?
- Продължителност на предварителния преглед.

Процесът на предварителен преглед може да бъде схематично представен



Дейности на прегледа на изискванията (1)

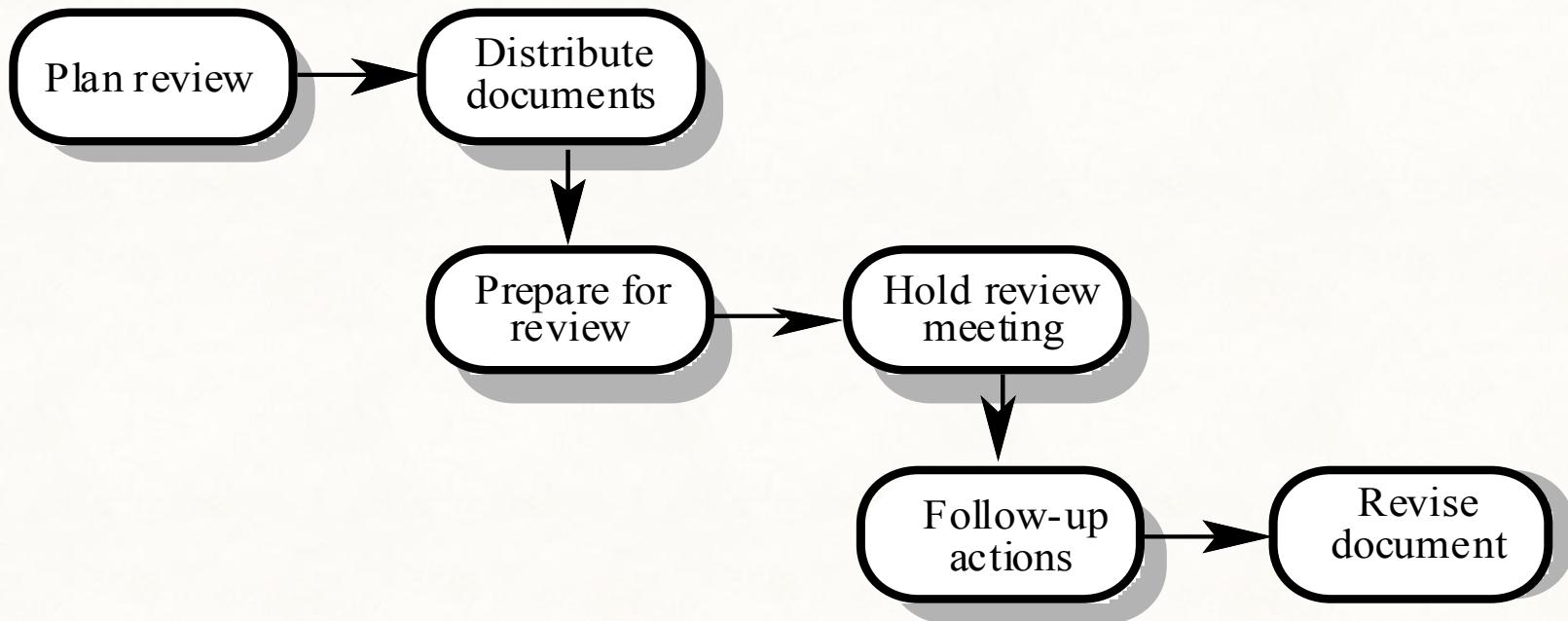
- **Планиране**
 - Определя се екип, ръководител и протоколчик; избира се време и място за срещите.
- **Разпространение** на документите за преглед до участниците в екипа.
- **Подготовка** за прегледа:
 - Всеки рецензент да прочете документа за изискванията и да набележи открити конфликти, пропуски, непоследователности, отклонения от стандартите, др. проблеми.

Дейности на прегледа на изискванията (2)

- **Среща по прегледа**
 - Дискутират се коментари и проблеми от отделни участници от екипа и се дефинира набора от възможни действия за справяне с проблемите.
- **Проследяване на действията**
 - Ръководителят на екипа проверява дали формулираните действия са изпълнени.
- **Ревизиране на документа**
 - Документът на изискванията се редактира, за да отрази съгласуваните действия. На този етап той може да бъде предаден като окончателен вариант или върнат за нов преглед.

Кой може да бъде ръководител на екипа по преглед на изискванията?

Модел на процеса на преглед на изискванията



(Възможни) Действия за решаване на всеки отделен проблем

- **Изясняване на изискването**
 - лошо представено или да има случайно изтървана информация.
Презаписване на изискването.
- **Липсваща информация**
 - Отговорност на инженерите по изискванията е да *открият* тази информация от съответен източник.
- **Конфликт в изискванията**
 - Ако съществува значим конфликт в изискванията, то трябва да се *преговаря*, за да достигне до съгласие.
- **Нерелистични изисквания**
 - Изискването не би могло да бъде реализирано, защото има технологични и/или друго ограничение върху системата. Необходимо е ползвателите да се консултират, как да направят изискването *по-реалистично*.

Време за преглед: 2-4 изисквания за час!

Каква е разликата от справяне с програмни грешки?

....

Участници в екипа на прегледа

- Част от рецензентите трябва да включват **ползватели** с различни специалности, за да се
 - използват различни умения и знания;
 - разбере по-добре необходимостта от промяна на изискванията;
- В екипа трябва задължително да се включи:
 - *експерт от специфичната област* на създаваната система, както и поне един *краен потребител на системата*.
- В екипа е добре да бъдат включени:
 - *Клиент*
 - *Разработчик на системата*
- Брой на експертите, формиращи екипа - от 3-10 человека;
 - Организиране на работата на екипа: компромис с екипа, но по-малко с времето.

Преглед/ревю на изискванията - характеристики

- Ревютата трябва да се организират регулярно
- Екипът на клиента и екипът на изпълнителя трябва да участват в ревютата.
- Ревютата могат да бъдат:
 - формални (with completed documents) or
 - неформални.

Въпроси на прегледа

| | |
|--|---|
| <ul style="list-style-type: none">• Единствено формулиране / идентифициране на дадено изискване? | Проследимост; Съответствие със стандартите |
| <ul style="list-style-type: none">• Даден ли е речник на специализираните термини? | Разбираемост |
| <ul style="list-style-type: none">• Разбираемо ли е само за себе си изискването или е необходимо да бъде и разгледано и друго, за да бъде разбрано даденото? | Разбираемост; Пълнота |
| <ul style="list-style-type: none">• Използван ли е последователно един и същи термин в описанието на изискването? | Неяснота |
| <ul style="list-style-type: none">• Изиска ли се една и съща услуга при на различни изисквания? | Консистентност; излишество |
| <ul style="list-style-type: none">• Има ли противоречия в информацията при тези изисквания? | |
| <ul style="list-style-type: none">• Ако изискване се отнася и до други характеристики, те описани ли са в документа? | Пълнота |
| <ul style="list-style-type: none">• Групирани ли са заедно изисквания, които са свързани? Ако не - взаимно отнасят/насочват ли се? | Организираност; Проследимост |

Проблеми в описаните изисквания - пример

- “4. EDDIS will be configurable so that it will comply with the requirements of all UK and (where relevant) international copyright legislation. Minimally, this means that EDDIS must provide a form for the user to sign the Copyright Declaration statement. It also means that EDDIS must keep track of Copyright Declaration statements, which have been signed/not-signed. Under no circumstances must an order be sent to the supplier if the copyright statement has not been signed.”

Открити проблеми:

- Не пълно (**Incompleteness**)
 - What international copyright legislation is relevant?

Recommendation:

....

- Не ясно (**Ambiguity**)

– Signature?

- *Recommendation:*

...

- Съответствие със стандартите (**Standards**)

...

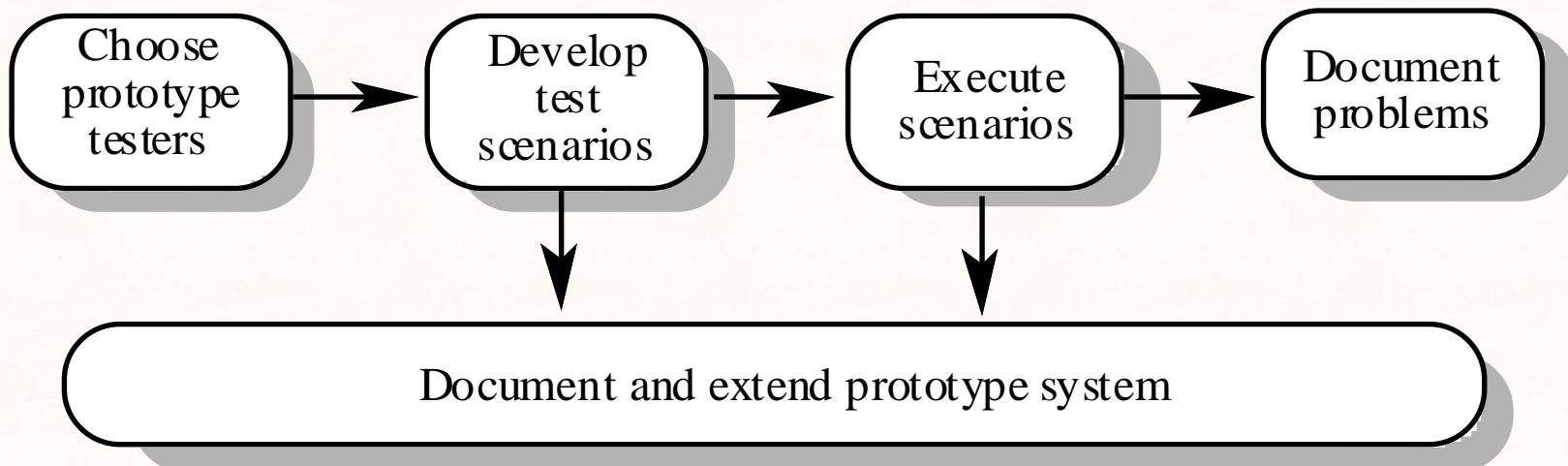
2. Прототипиране

Важно! Техниката на прототипиране в процеса на валидиране заслужава да се прилага **само** в случай, че вече е разработен прототип за целите на извличане на изискванията.

- Прототипите, предназначени за валидиране на изискванията, имат за цел да **демонстрират изискванията и да помогнат на** заинтересованите лица да **открият проблеми**.
- Трябва да бъдат достатъчно **пълни, ефикасни и робастни**, за да се оцени използването на исканата система.
- Валидирането може да започне с непълен прототип, но задължително системата трябва да бъде допълнена и довършена по време на валидирането.
- **Документация** и ръководство за ползване *трябва* да бъдат предоставени.

Прототипиране за целите на валидирането

Когато прототипите, създадени в процеса на извличане, са с непълна функционалност, то прототипът се доработва по време на валидирането:



Действия при прототипирането: успоредни процеси

- **Избор на подходящ екип за тестване на прототипа**
 - Потребители с *умерен* опит и които са “отворени” към използването на нови системи.
 - Крайни потребители с *различен характер на извършваната дейност*
- **Разработване на тестови сценарии**
 - Внимателното планиране предполага формулиране на достатъчно *пълен набор* от тестови сценарии.
- **Изпълнение на сценариите**
 - При изпълнение на сценариите потребителите е добре да работят *самостоятелно*, като по този начин симулират и реалната ситуация на използването на системата.
- **Документиране на проблемите**
 - Съставяне на електронен или хартиен *документ* с откритите проблеми.

2.1. Разработване на ръководство на потребителя

- Писането на ръководство стимулира детайлния анализ и допринася за изясняване на изискванията. (effect of requirements rewriting)
- Информация, която трябва да се съдържа в ръководството
 - Описание на функционалността и как тя е реализирана;
 - Коя част на системата не е изпълнена
 - Как да се излезе/възстанови нормалния статус при наличие на конфликт или проблем.
 - Как да се инсталира и стартира системата
- Ръководство на потребителя
 - Чернови вариант (дори и без наличие на прототип)
 - Чернова за създаване на крайния вариант на документацията на системата.
 - Цената за писане на ръководството да се включи към разходите за 28 валидиране.

3. Валидиране на модел

Проверката на точността на създадените модели на системата е съществено в процеса на валидиране.

- **Цел на валидирането** на модела
 - *моделът е логичен, съдържа цялата необходима информация, няма конфликти* (self consistent)
 - *моделите за една система са вътрешно и външно съвместими* (internally and externally consistent)
 - *всеки модел точно представя реалните изисквания* на клиента
- Някои проверки са възможни с **автоматични** софтуерни инструменти
 - CASE tools support. (Какво може да бъде проверено?)
- **Перифразирането** на модела е ефективна техника за проверка

Перифразиране

A) Шаблони

Пример: DFD -> перифразирано описание

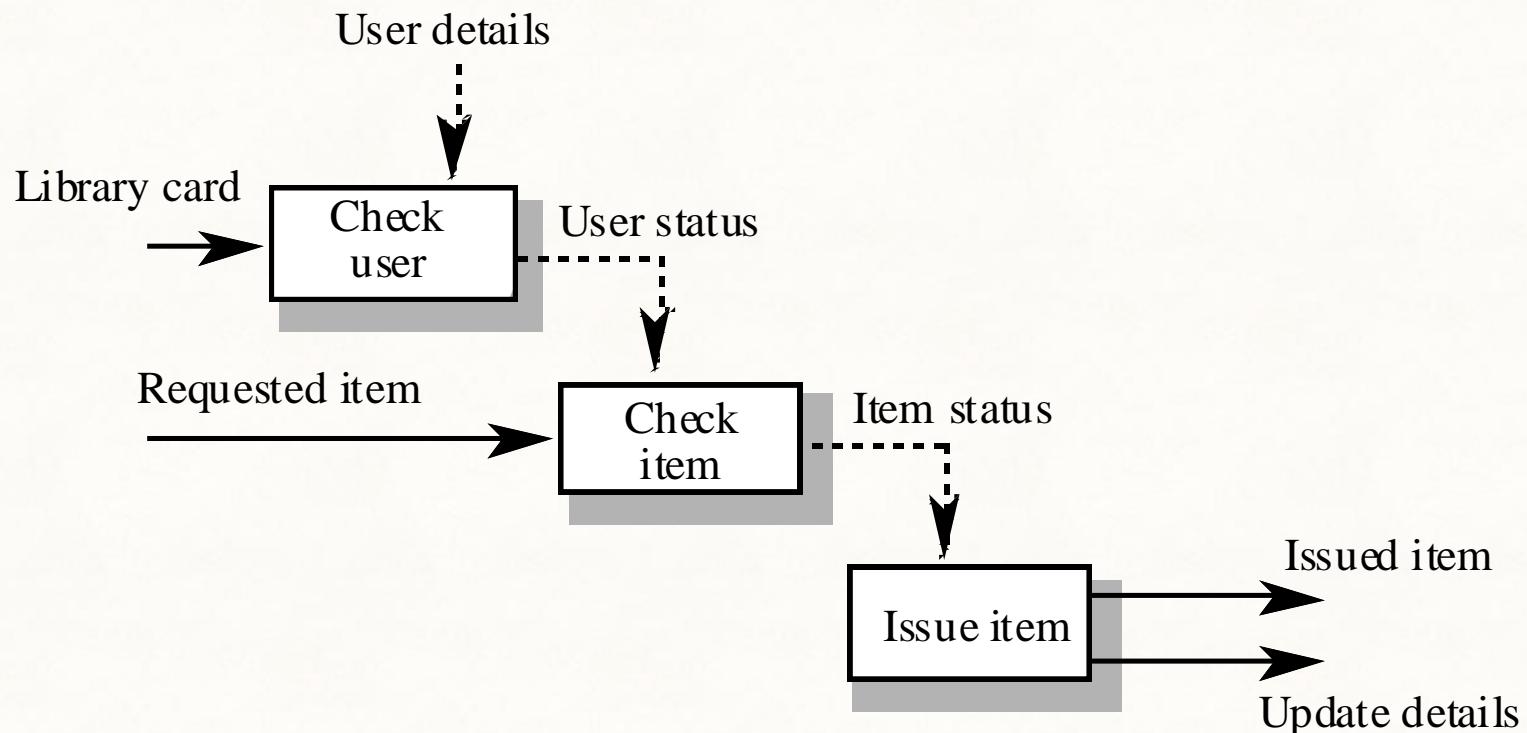
1. Име на осъществяваната трансформация (функция)
2. Източник и входна информация
3. Функционалност на трансформацията
4. Изходна информация и за къде е предназначена
5. Изключения или управляваща информация

Позволява да се открие лесно липсваща информация.

Б) “Автоматизирани” шаблони – отчети в Case tools - проблеми.

В) Математическо доказателство при използване на формални спецификации и модели.

Пример: DFD за услугата заемане на библиотечно издание (item) от читател на библиотека



Пример: Перифразиране на Data-flow диаграма за заемане на библиотечно издание

Check user

| | |
|-------------------------|--|
| Inputs and sources | User's library card from end-user |
| Transformation function | Checks that the user is a valid library user |
| Transformation outputs | The user's status |
| Control information | User details from the database |

Check item

| | |
|-------------------------|--|
| Inputs and sources | The user's status from Check user |
| Transformation function | Checks if an item is available for issue |
| Transformation outputs | The item's status |
| Control information | The availability of the item |

Issue item

| | |
|-------------------------|---|
| Inputs and sources | None |
| Transformation function | Issues an item to the library user. Items are stamped with a return date. |
| Transformation outputs | The item issued to the end user |
| Control information | Database update details |
| | Item status - items only issued if available |

4. Тестване на изискванията

- За всяко изискване трябва да може да бъде *съставен тест/ове за проверка* дали изискването е дефинирано добре.
- Невъзможността да се създаде тест говори за липсваща или неясна информация в описанието на изискванията.
- **Всяко функционално** изискване в документа на изискванията трябва да бъде *анализирано с подходящ тест. (requirements testability)*
- *Тест за изискването или тест на системата ?*

Тестови случаи за изследване на описаните изисквания

Въпроси, подпомагащи съставянето на тест за изискванията:

- Каква *употреба* ще се тества? (*контекст на сценария*)
- Съдържа ли изискването *достатъчна информация*, за да позволи разработване на тест? (други изисквания, зависимости между тях?)
- *Един* или повече тестове са нужни, за да се направи изследването?
(Какво евентуално може да показва необходимостта от повече тестове?)
- Може ли изискването да бъде *преформулирано*, за да бъдат тестовите случаи по-ясно дефинирани?

Проектиране и попълване на формуляр на тест за всяко изискване (Test record form)

- **Идентификатор** на изискването
- **Свързани изисквания**
- **Описание на теста**
 - Кратко описание на теста и обяснение защо е обективен за даденото изискване; описание на входната и изходна информация на системата.
- **Проблеми**
 - Описание на проблемите, които правят теста труден или невъзможен.
- **Коментари и препоръки**
 - Съвети за евентуални решения на откритите проблеми.

Пример: Формуляр на тест за изискване 10.iv на EDDIS

10.(iv) When users access EDDIS, they will be presented with web pages and all the services available for them

Requirements tested: 10.(iv)

Related requirements: 10.(i), 10.(ii), 10.(iii), 10.(vi), 10. (vii)

Test applied: For each class of user, prepare a login script and identify the services expected for that class of user.

The results of the login should be a web page with a menu of available services.

Requirements problems: We don't know the different classes of EDDIS user and the services which are available to each user class. Apart from the administrator, are all other EDDIS users in the same class?

Recommendations: Explicitly list all user classes and the services which they can access.

Трудности при проектиране на тест за някои типове изисквания

- **Изисквания за системата**

- Това са изисквания, прилагани за системата като *цяло*. Те са най-трудните за валидиране независимо от използвания метод. Тестове, които *не се изпълняват*, не могат да тестват нефункционални, общи системни характеристики като например използваемост.

- **Изключения**

- Изисквания, които *изключват* специфично поведение. Например: изискването формулира, че системните повреди *никога не трябва* да допускат повреда на базата данни. Такова изискване *не би могло да бъде тествано изчерпателно*.

- **Някои нефункционални изисквания**

- Някои нефункционални изисквания като надеждност могат да бъдат тествани само с *голям брой тестове*.

Управление на изискванията

Лекция 8

Управление на изискванията

- Приоритизиране на изискванията
- Цели на управлението на изискванията
- Постоянни и променливи изисквания
- Управление на промените
- Проследимост и проследяване на изискванията

Приоритизиране на изискванията

- **Приоритет на изискванията**

Приоритетът на едно изискване показва *важността* на изискването спрямо останалите изисквания според един или няколко критерия за *приоритизиране*:

- отделно за *всяко* изискване или
- чрез сравняване на изискванията *по двойки*.

- **Приоритизирането е и въпрос на “управлението”:**

Мениджърът на проекта трябва да уравновесява обхвата на проекта според ограниченията на плана, бюджета, персонала и целите за качество.

- **Критерии за приоритизиране**

- **Най-съществени функции**
- Или на база на стойност (ресурс), цена и риск (K. Wiegers)

- **Техники за приоритизиране**

Приоритизиране на изискванията с MoSCoW

- MoSCoW е *техника за приоритизиране*, която се използва, за да се постигне споразумение със заинтересованите страни за важността, която те поставят на изпълнението на всяко изискване (**using a set of words that have meaning**).
- Разработена от Dai Clegg от Oracle UK Consulting; в CASE Method Fast-Track (RAD)

- **M - MUST** (задължително трябва да го има).
- **S - SHOULD** (необходимо е да го има, ако е възможно).
- **C - COULD** (може да го има, ако не влияе на нещо друго).
- **W - WON'T** (няма да го има, но е желателно (**WOULD**) в бъдеще).

Приоритизиране на изискванията с използване на RFC 2119 Harvard University

- В много стандарти на документи няколко думи, често са с главни букви, се използват, за да обозначат изискванията в спецификацията.

Авторите, които следват тези указания *трябва да включат тази фраза в началото на документа:*

Ключовите думи:

"MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",
"OPTIONAL"

RFC-Request for Comments

Управление на изискванията

Дефиниция. Процес на управление на изискванията е *процес на управление на промените на изискванията на системата.*

Управлението на изискванията се осъществява *паралелно* с другите дейности от ИИ и *продължава* след предоставяне на първата версия на документа с изискванията, както и по време на разработването на системата.

Управление на изискванията

Главните задачи на управлението на изискванията са:

- Управление на промените в *уговорените* изисквания
 - Управление на връзките *между* изискванията
 - Управление на *зависимостите* между документа на изискванията и *други* документи, създадени в процеса на инженеринг на изискванията.
-
- Изискванията *не могат да се управляват* ефективно без да има **проследимост** на изискванията

Едно изискване е **проследимо** ако е ясно *кой* е предложил изискването, *защо* съществува изискването, *кои* изисквания са *свързани* с него и *как* това изискване е свързано с *друга* информация като дизайн на системата, реализация и потребителска документация.

Управление на изискванията

Главните задачи на управлението на изискванията са:

- Управление на промените в *уговорените* изисквания
 - Управление на връзките *между* изискванията
 - Управление на *зависимостите* между документа на изискванията и *други* документи, създадени в процеса на инженеринг на изискванията.
-
- Изискванията *не могат да се управляват* ефективно без да има **проследимост** на изискванията

Едно изискване е **проследимо** ако е ясно *кой* е предложил изискването, *защо* съществува изискването, *кои* изисквания са *свързани* с него и *как* това изискване е свързано с *друга* информация като проект, реализация и потребителска документация.

Управление на изискванията

- Добрата практика изисква да “очакваме” промените
 - ❖ повече от 50% от изискванията се променят преди и по време на разработката (Summerville)
- *Произход* на промяната на системата
- *Документиране*
- *Приоритет* на промените

Фактори за промяна на изискванията (1)

- *Грешки и неразбиране в изискванията*
 - могат да бъдат открити по време на анализа и валидирането на изискванията или по-късно в процеса на разработката.
- *Знанията (=> искаанията) на клиента/ крайния потребител за системата се развиват и увеличават*
- *Технически проблеми, разминаване с графика или с разходите.*

Кое от изброените е най-честата причина за необходимост от промяната?

Фактори за промяна на изискванията (2)

- Променящи се *приоритети на клиентите*
 - Приоритетите на клиентите се променят като резултат от *променящата се бизнес среда, появата на нови конкуренти, смяна на персонала* и др.
- Изменения във *външни обстоятелства*
 - Средата, в която ще бъде инсталирана системата, може да се промени, така че да се наложи промяна на системните изисквания. (пр. закони, нов информационен поток, технологии...)
- *Организационни промени*
 - Организацията, която ще използва системата, може да измени своята структура и бизнес процеси.

Кой от факторите най-често е причина за промяната?

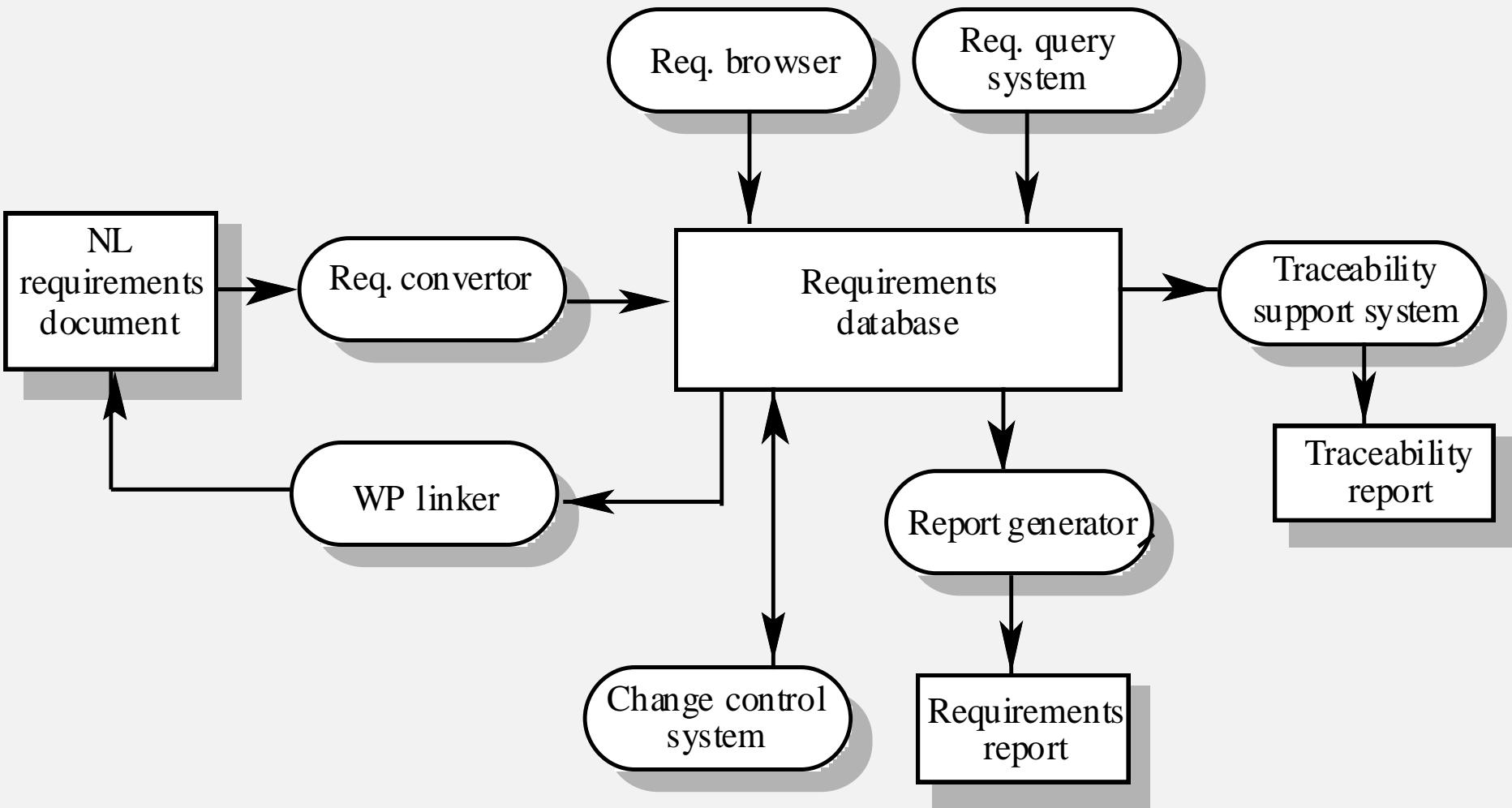
CASE средства за управление на изискванията

- Управлението на изискванията включва *събиране, съхранение и поддържане на големи количества информация*
- Съществува набор от CASE средства, които са *специално създадени*, за да подпомагат управлението на изискванията: DOORS, RML, RDD-100
- Други CASE средства като системите за управление на конфигурациите могат да бъдат *приспособени* към инженеринга на изискванията

Инструментите за подпомагане на управлението на изискванията могат да предоставят следните средства:

- *База от данни за съхраняване на изискванията.*
- *Средства за анализ и генериране на документи за създаване на документи с изискванията.*
- *Средства за управление на изискванията, с които се проверява дали промените са точно оценени и остойностени.*
- *Средства за проследяване, които помагат да се открият зависимости между системните изисквания.*

Система за управление на изискванията



Постоянни и променливи изисквания

Промяната на изискванията е неизбежна,
но !

Някои изисквания са обект на промяна повече от други

- *Постоянните изисквания* се отнасят до *същността на системата* и нейната *приложна област*.
- *Променливите изисквания* са специфични за *екземпляра на системата в определена среда* и за *определен клиент*.
 - как може да бъде използвано знанието за видовете изисквания?

Фактори за промяна на изискванията: Пример

Пример: Система за управление на документите на студентите (протоколи/информация за студентите, курсовете, оценките и др.) в университет.

Дефинирайте:

- А) постоянни изисквания:

- В) променливи изисквания:

Видове променливи изисквания

- **Непостоянни изисквания**
 - Промените се дължат на средата, в която работи системата (*pr.* Tax deduction system)
- **Неочеквани (emergent) изисквания**
 - Това са изисквания, които *не могат да бъдат дефинирани напълно* при специфицирането на системата, но се появяват докато системата се проектира и реализира(*pr.* интерфейс, ...)
- **Consequential requirements**
 - Базирани на предположения за това, как ще бъде използвана системата. Когато системата *влезе в употреба*, някои от тези предположения се оказват погрешни.
- **Изисквания за съвместимост (Compatibility requirements)**
 - Това са изисквания, които *зависят от друго оборудване* или процеси (*pr.* система за управление).

Идентифициране и съхраняване на изискванията

- **Важно е всяко изискване да има уникална идентификация**
- Изискванията трябва да се съхраняват така, че **достъпът** до тях и за *свързаните* с тях изисквания да е **лесен**.
- Да се осигури възможност за работа на **повече хора**.

Съхранение на изискванията

- Най-разпространеният подход е **номериране на изискванията** според *секцията* и подсекциите в документа с изискванията.

Проблемите:

- Номерата *не могат* да бъдат поставени преди завършването на документа.
- *неявна* класификация (и близост) на изискванията, което внася *неточност*.

Възможни начини за съхранение:

- В един или повече текстови файлове.
 - ✓ Динамично преномериране: автоматично преномериране на параграфите и включване на препратки (cross-references).
 - ✓ Contents management systems

Текстови документи на изискванията

- **Предимства**

- Всички изисквания се съхраняват на едно място
- Достъпът до изискванията е възможен за всеки, който има подходящо приложение за текстообработка
- Лесно е да се състави крайният документ с изисквания

- **Недостатъци**

- *Зависимостите* между изискванията трябва да се поддържат *външно*
- Възможностите за търсене са ограничени
- Не е възможно *свързването* на изискванията с предложените промени в изискванията
- Не е възможно да има *контрол* на версията за *отделните* изисквания
- *Няма автоматична навигация* от едно изискване до друго

решение?

Други техники за идентификация на изискванията, записани чрез текстов софтуер (2)

- **Символна идентификация**
 - Изискванията се означават със символно име, което е свързано със самото изискване (*Пр.*: EFF-1, EFF-2, EFF-3 за системна ефективност (**efficiency**)). *Проблеми?*
- **Идентификация според записите в базата от данни**
 - Изискването се въвежда като се задава идентификатор на съответния запис.
 - Основно копие, което да се ползва от повече хора.

База от данни за изискванията

- Всяко изискване се представя като елемент на базата данни.
- За достъп до изискванията се описват *заявки*.
- **Предимства**
 - Добри възможности за заявки, търсене и навигация
 - Поддръжка на управлението на промените и на версии.
- **Недостатъци**
 - Необходим е *софтуер* и *умения* за работа с базата от данни.
 - Трябва да се *поддържа връзка* между базата от данни и документа с изискванията.

Какъв софтуер за БД да се използва?

Класове за БД с изисквания:

- а) релационни
- б) обектно-ориентирани

| SYS_MODELS | REQUIREMENT | SOURCE_LIST |
|---|--|---|
| Model: MODEL Description: TEXT Next: MODEL NULL | Identifier: TEXT Statement: TEXT GRAPHIC Date_entered: DATE Date_changed: DATE Sources: SOURCE_LIST Rationale: REQ_RATIONALE Status: STATUS Dependents: REQ_LIST Is_dependent_on REQ_LIST Model_links: SYS_MODELS Comments: TEXT | People: TEXT Documents: TEXT Reqs: REQ_LIST |
| REQ_LIST | REQ_RATIONALE | |
| Req: REQUIREMENT Description: TEXT Next: REQUIREMENT NULL | Rationale: TEXT Diagrams: GRAPHIC Photos: PICTURE | |

**Обектно-ориентирани БД: кои са техните предимства?
Кога са рентабилни?**

БД за изисквания – фактори за избор (1)

- **Изложението (statement) на изискванията**
 - Ако е необходимо съхранението на други данни освен текст, може да е необходимо да се използва база от данни с *мултимедийни възможности*.
- **Брой на изискванията**
 - По-големите системи обикновено имат нужда от база от данни, която е предназначена да управлява много голямо количество информация и да работи на специализиран софтуер.
- **Работа в екип, разпределение на екипа и компютърна поддръжка**
 - Ако изискванията се разработват от разпределен екип от хора и от различни организации, тогава е необходима база от данни с възможност за отдалечен достъп от много места.

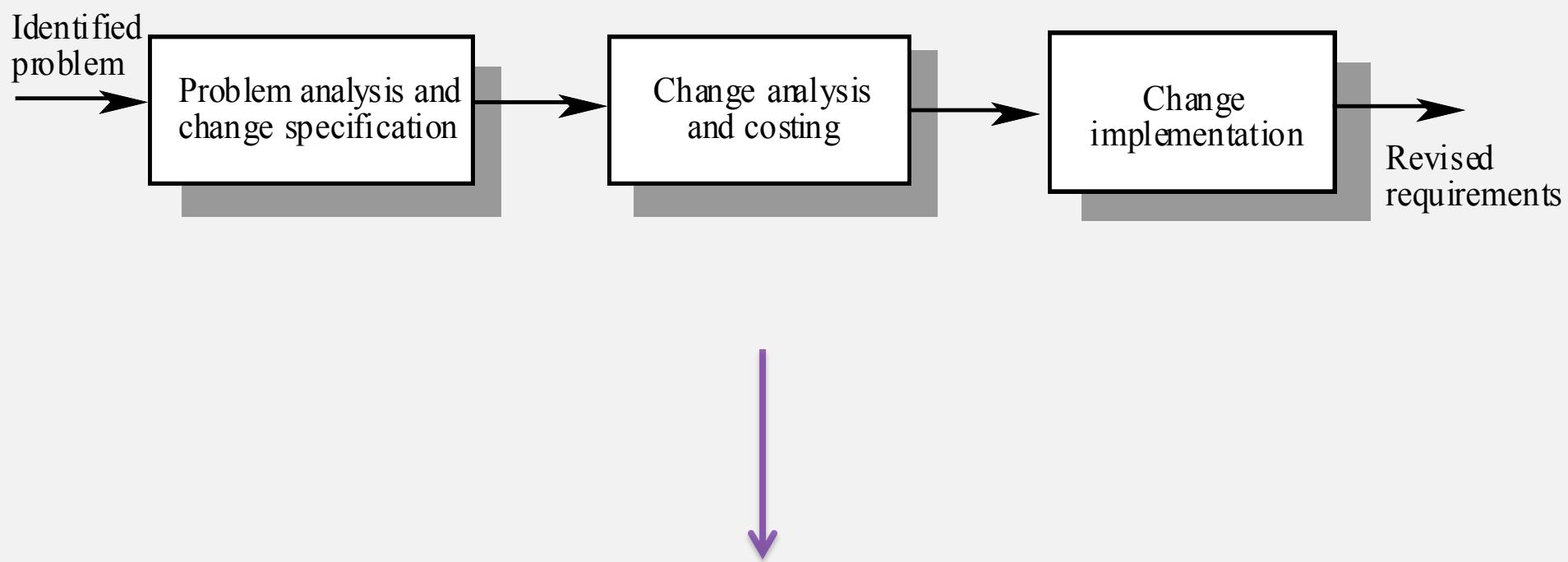
БД за изисквания – фактори за избор (2)

- Използване на CASE (Computer Aided SE) средства
 - Базата от данни трябва да е *същата като или съвместима* с базите от данни в CASE средствата. Това обаче може да бъде проблем при някои CASE средства, които използват своя собствена proprietary (патентована) база от данни
- Използване на съществуваща база от данни
 - Ако има база от данни, която вече се използва за поддръжка, то тя трябва да се използва за управление на изискванията.

Управление на промените (Change management)

- Управлението на промените включва *процедурите, процесите и стандартите*, които се използват при промяна в изискванията.
- **Политики за управление на промените:**
 - Процес на заявка за промяна и *информацията*, необходима за обработката на всяка заявка за промяна.
 - Процес за *анализ на въздействието и разходите за промяната* и съответната информация за проследимост.
 - Определяне на (независими) *група*, която да разглежда заявките за промяна.
 - *Софтуерна поддръжка* (ако я има) за процеса на управление на промените.

Процес на управление на промените - 1



Процес на управление на промените - 2

- **Открива се** проблем в изискванията

Проблемът може да се прояви при анализа на изискванията, *нови нужди на клиентите или проблеми в работата на системата.*

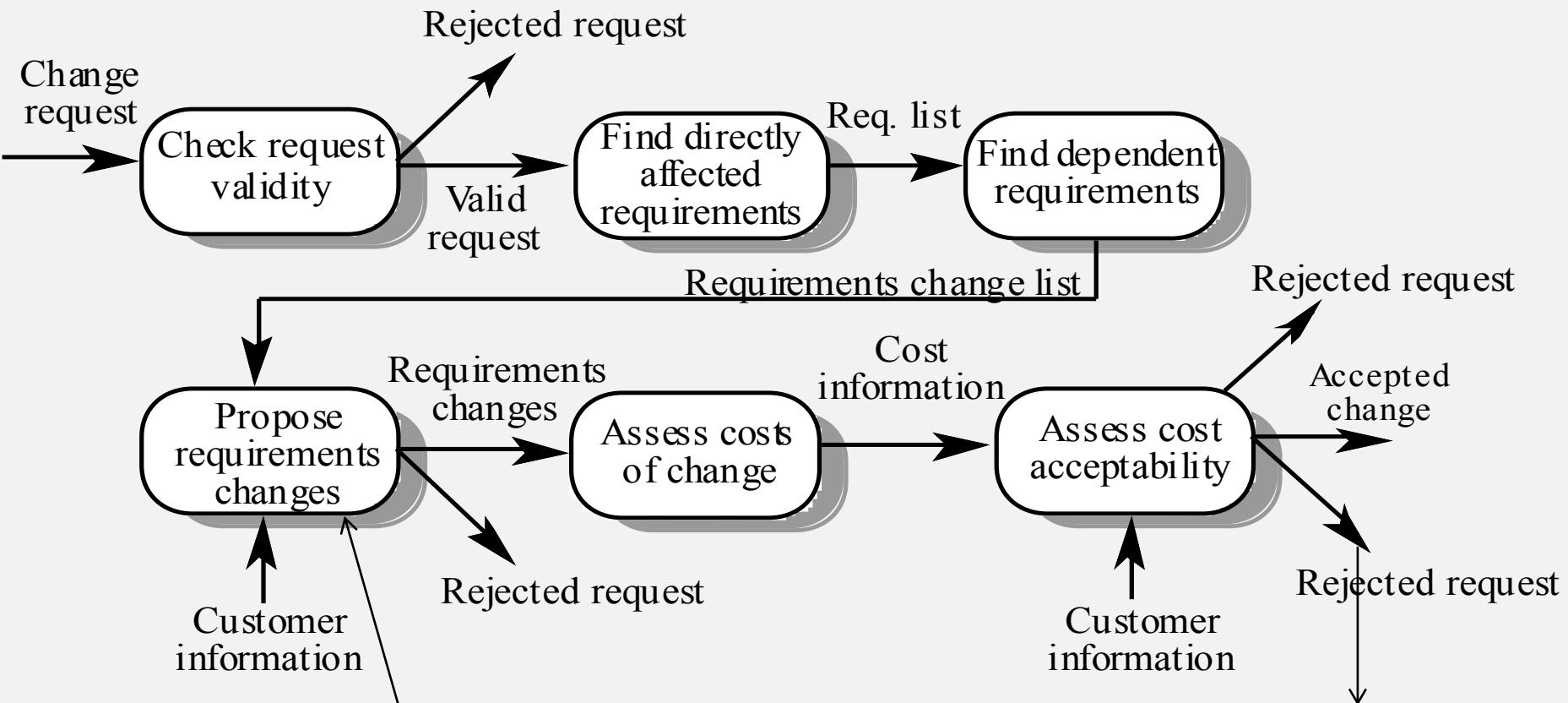
- **Предложените промени се анализират**

Проверява се колко изисквания са засегнати от промяната и колко е приблизителната *цена във време и пари*, за да се направи промяната.

- **Промяната се реализира**

Извършват се *поправки* в документа с изискванията или се съставя *нова версия на документа*. Валидира се с използване на *процедури за проверка на качеството*.

Анализ и оценка на промените



Колко са основните дейности?

Кои са етапите , в които е възможно отхвърляне на промените и какви са съответните причини за това?

Обработка на промените - документи

- Предложените промени – във *формуляр за заявка за промяна*, който се предава и обновява от всички участващи в анализа на промените.
- Формулярите за заявка за промяна:
 - Документиране на резултата от всеки етап от анализа на промените
 - Дати
 - Хората, отговорни за промените
 - Статус (“отхвърлена”, “приета”, ...)
 - Коментари
 - Запис в БД (кога?)

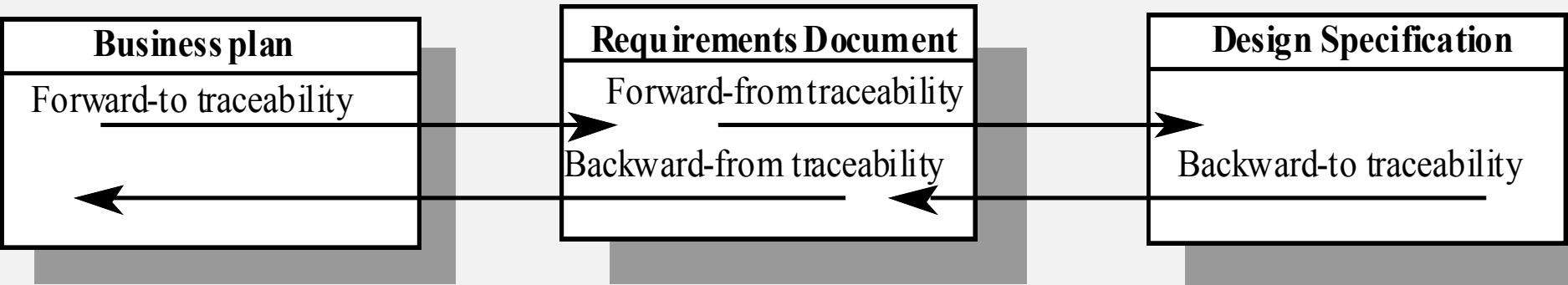
Софтуерни инструменти за управление на промените

- Специализирани *инструменти* за управление на изискванията – обективна необходимост
- **Инструментите могат да имат следните възможности:**
 - Електронни формуляри за заявяване на промени, които се попълват от различни участници в процеса.
 - База от данни за съхранение и управление на тези формуляри.
 - Модел за промяна, който да информира *така че хората, които отговарят* за даден етап от процеса, да знаят кой отговаря за следващата дейност от процеса.
 - Електронен трансфер на формулярите и за информиране между хората с различни отговорности.
 - Директни връзки към база от данни с изисквания (само при най-сложните инструменти).
- *Проблеми:* налагат собствен модел на промяната, цена:
 - За големи организации + големи проекти
 - Алтернатива за по-малки проекти: Intranet и Internet базирани системи

Проследимост и проследяване

- **Информацията за проследимост (traceability information)** е информация за зависимостта, рационалността и реализацията на изискванията, която подпомага *оценката на въздействието* на промяната на изискванията.
- **Видове информация за проследимост** (Davis, 1993)
 - *Backward-from проследимост* Свързва изискванията с техните източници от други документи или хора
 - *Forward-from проследимост* Свързва изискванията с компонентите на дизайна и реализацията
 - *Backward-to проследимост* Свързва компонентите на дизайна и реализацията обратно към изискванията
 - *Forward-to проследимост* Свързва други документи, които са предшествали документа с изискванията, със съответните изисквания.

Проследимост backward/forward



**Информацията за проследимост трябва да включва
и документа с изискванията - защо?**

Кой трябва да определи политиките за проследимост?

Коя проследимост най-често се поддържа/използва?

Видове проследимост – примери 1

- Проследимост изисквания-източници
 - Свързва изискването с хората или документите, които инициират изискването
- Проследимост изисквания-рационалност (обосновка)
 - Свързва изискването с описание *защо* е зададено това изискване.
- Проследимост изисквания-изисквания
 - Свързва изисквания с *други изисквания*.
 - Двупосочна връзка.

Видове проследимост - примери 2

- Проследимост изисквания-архитектура
 - Свързва изискванията с *подсистемите*, където са реализирани тези изисквания. (особено важно в случаите на *различни подизпълнители*)
- Проследимост изисквания-дизайн
 - Свързва изискванията със *специфични хардуерни и софтуерни компоненти*, нужни за реализацията
- Проследимост изисквания-интерфейси
 - Свързва изискванията с *интерфейсите на външни системи*, които се използват

Таблици за проследяване на изискванията

- Показва *връзките между изискванията* (група от изисквания) или между изискванията и *компонентите* на дизайна
- Изискванията (или документите) се подреждат по хоризонталната и вертикалната ос, а връзките между изискванията се *отбелязват* в *клетките на таблицата*
- Таблиците за проследимост трява да се дефинират с номерирани изисквания за обозначаване на редовете и колоните на таблицата

Таблица за проследимост

Depends-on

| | R1 | R2 | R3 | R4 | R5 | R6 |
|----|----|----|----|----|----|----|
| R1 | | | * | * | | |
| R2 | | | | | * | * |
| R3 | | | | * | * | |
| R4 | | * | | | | |
| R5 | | | | | | * |
| R6 | | | | | | |

Кога и как се използва таблица за проследимост, ако са идентифицирани голям брой изисквания?

Списъци за проследяване

- Таблициите за проследяване могат да се реализират чрез *spreadsheet* в случай, че броят на изискванията е сравнително малък (няколкостотин)
- Друга възможност е използването на опростена форма на таблица за проследяване, в която заедно с описанието на всяко изискване се поддържат един или повече списъци на идентификаторите на съответните *групови* изисквания.
- Списъците за проследимост са *обикновени списъци с връзки*, които могат да се представят като текст или таблично.

Списък за проследимост

| Requirement | Depends-on |
|-------------|------------|
| R1 | R3, R4 |
| R2 | R5, R6 |
| R3 | R4, R5 |
| R4 | R2 |
| R5 | R6 |

Какъв е недостатъкът на списъка за проследимост?

Политики за проследяване на изискванията

- Основен проблем е високата цена за събиране, поддържане и анализа на информацията
- Политиките за проследяване дефинират *каква* информация за проследяване и *как* трябва да се поддържа.

Политики за проследяване

- *Информация* за съществуващите видове проследимост, която трябва да се поддържа.
- *Техники* като матрици за проследяване, които трябва да се използват.
- Указание *кога* трябва да бъде събрана информацията за проследяването по време на ИИ и разработването на системата.
- *Ролите* на хората, които са отговорни за поддръжката на информацията за проследимост.
- Указание за справяне и документиране на *изключенията* в политиката за проследимост (*напр.* при липса на време)
- Процес на *обновяване* на информацията за проследимост

Фактори, влияещи на специализацията на политиките за проследяване - 1

- **Брой на изискванията**
 - Колкото по-голям е броят на изискванията, толкова по-голяма е нуждата за *формални* политики за проследяване.
- **Продължителност на живота на системата**
 - За системите с *дълъг живот* трябва да се дефинират по-изчерпателни политики за проследяване.
- **Ниво на зрялост на организацията**
 - За подробните политики за проследяване е най-вероятно да бъдат *cost-effective* (рентабилни) в организации, които имат *по-високи нива на зрялост на процесите*.

Фактори, влияещи на специализацията на политиките за проследяване - 2

- **Големина и състав на екипа по проекта**
 - С малък екип е възможно да се оцени стойността на промените без структурирана информация за проследимост, но за големите екипи са необходими по-формални политики за проследяване.
- **Тип на системата**
 - Критичните системи като real-time control systems или системите с критично значение за безопасността изискват по изчерпателни политики за проследяване отколкото некритичните системи.
- **Специфични клиентски изисквания**
 - Предоставяне на специфична информация за проследимост като част от документацията на системата.

Възможност: Към спецификацията на изискванията да се добави и ръководство за проследимост на изискванията.

АНАЛИЗ НА СОФТУЕРНИТЕ ИЗИСКВАНИЯ

**Методи за извличане на изискванията,
основаващи се на методи на различните
гледни точки**

(Viewpoint-oriented requirements methods)

Лекция 9

Съдържание

- Понятие за гледни точки за ИИ
- Понятие за гледни точки в структурния анализ
- Подходи на гледните точки за ИИ
 - SADT
 - CORE
 - VOSE
 - VORD
- Валидиране на изискванията чрез методи на гледните точки

Анализ на изискванията, основаващ се на гледни точки

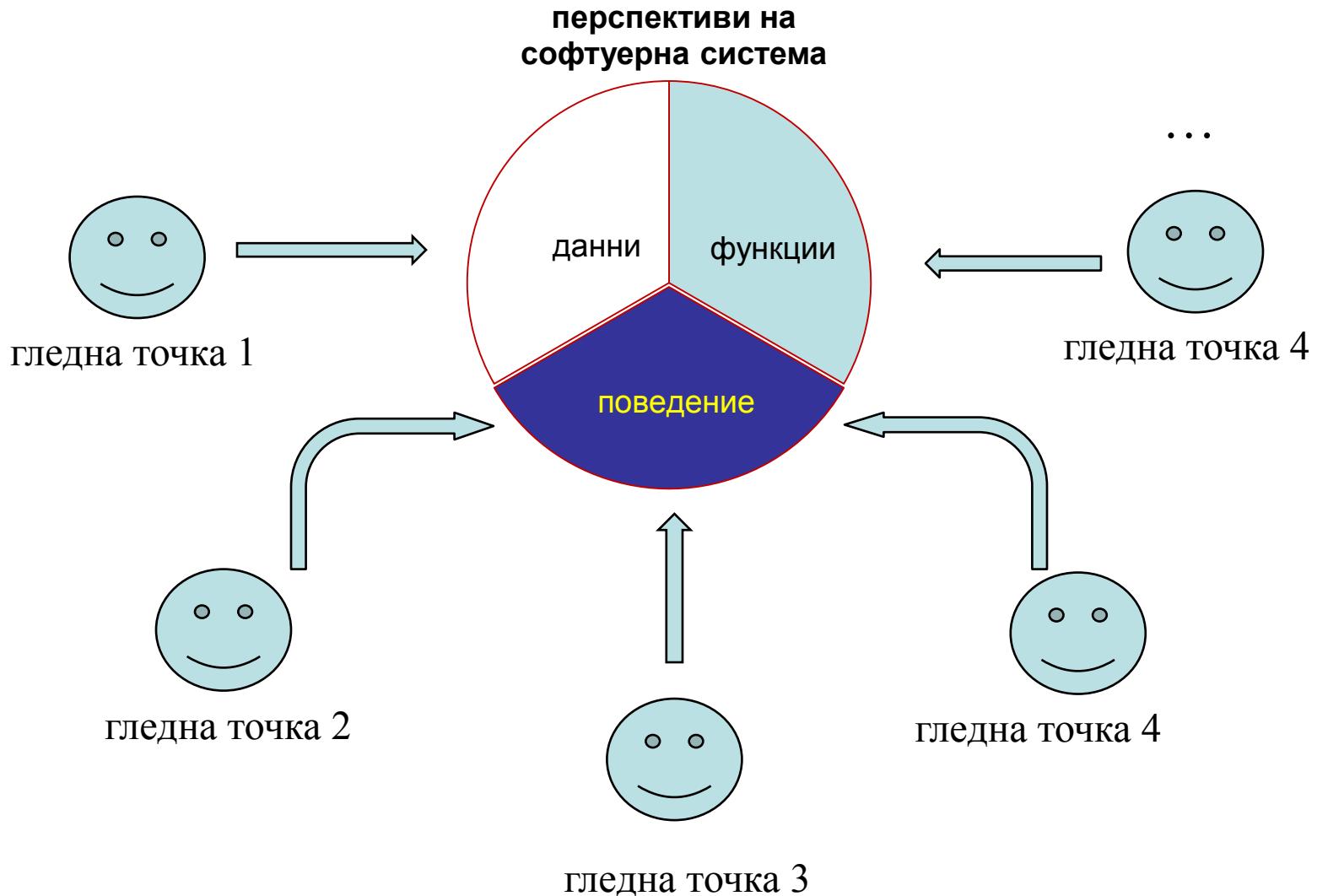
- Извличането на изискванията включва идентифициране, анализ и (раз)решаване на *различни идеи, перспективи и взаимовръзки* на отделни нива на детайлност.
- Методите за анализ, които са базирани на строги общи схеми, *не могат да отчетат всички съществуващи различни знания*, които са съществени за анализа.
- *Решение:* Методи, базирани на идеята за **гледните точки**

Гледна точка (в контекста на ИИ)

Дефиниция: Гледна точка е съвкупност от информация за системата или свързан проблем, среда или област, която е събрана от специфичното разбиране (виждане) на краен потребител или на друга система, или на разработчик, или друго заинтересовано лице.

Въпроси:

- Интегриране на информацията на отделните гледни точки;
- Решение на съществуващи конфликти



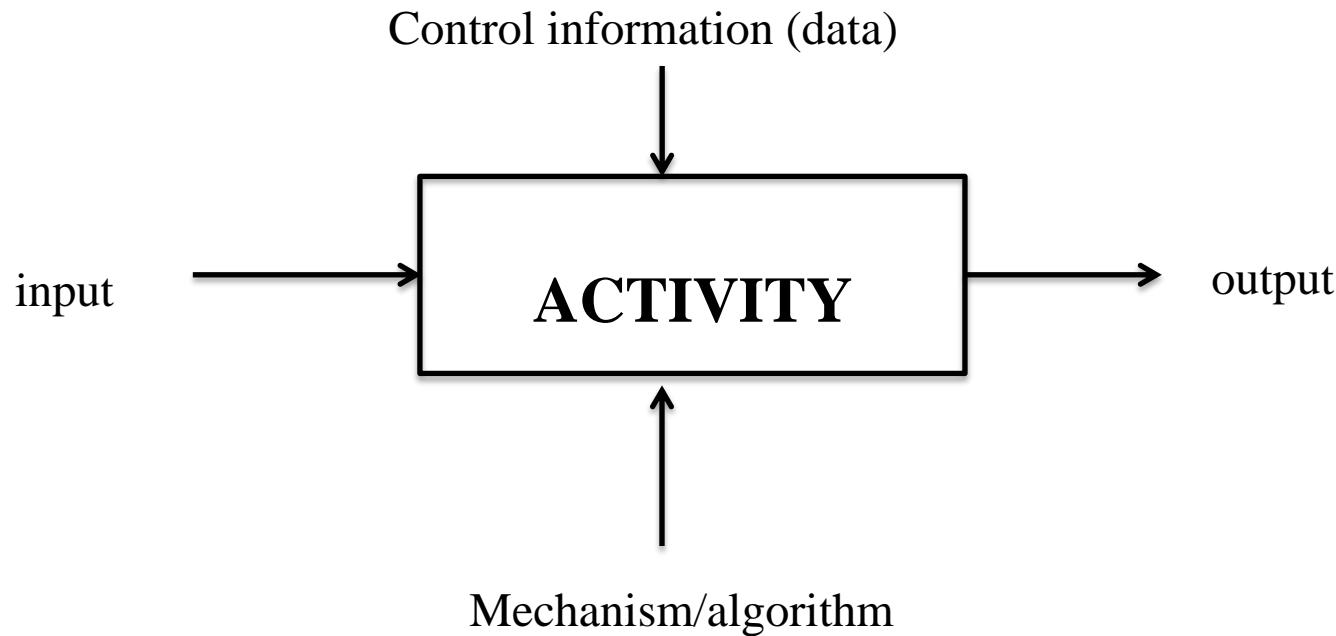
Предимства на подходите, базирани на гледни точки

- Разпознават *явно* разнообразието на източниците на изискванията.
- Осигуряват *механизъм* за организация и структуриране на тази *разнообразна* информация.
- Осигуряват *цялостност*.
- Осигуряват средство на източниците на изискванията или на заинтересованите страни *да открият и да проверят* своя принос към формулиране на изискванията.
- *Ефективност* на процеса на ИИ.

Подход на гледни точки чрез SADT (1)

- Техниката за **структурен анализ** и **техника за проектиране** (**SADT**) е разработена в края на 80-те от Ross и е имала широко приложение.
- Използва нотацията на структурния анализ (напр. модела на потока на данните (DFD)) за описание на множество от взаимодействащи си дейности.

SADT нотация

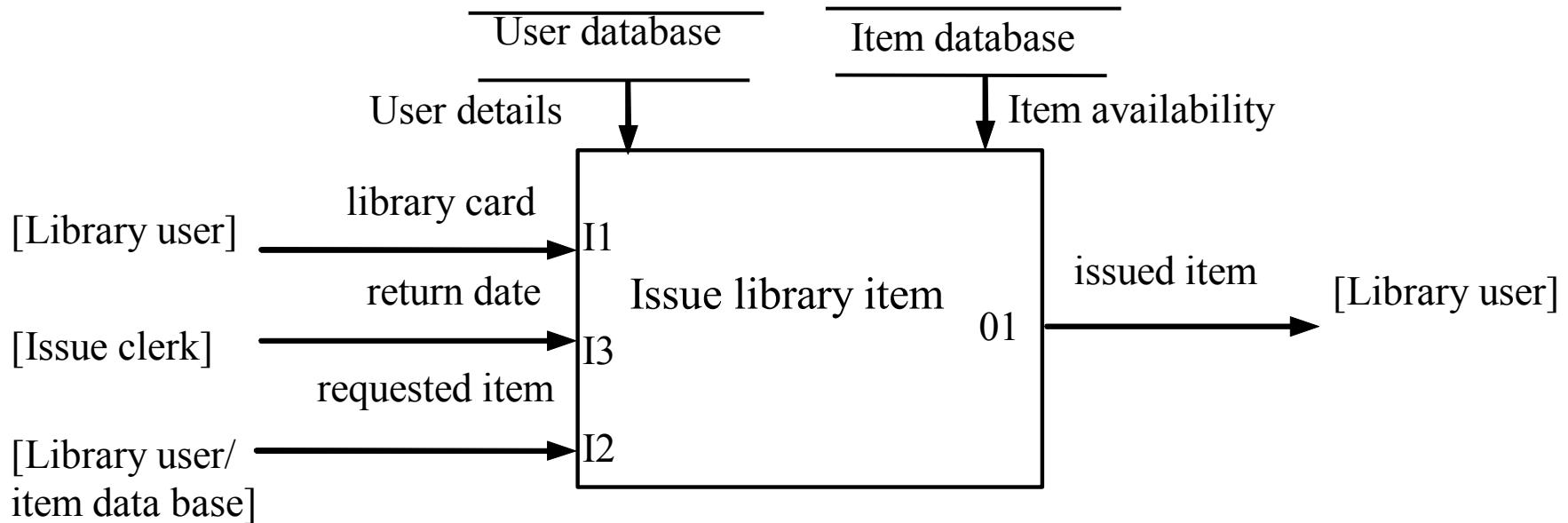


The next level of decomposition needs as a starting point for functional requirements.

Подход на гледни точки чрез SADT (2)

- SADT **декомпозира** проблема в множество от **йерархично свързани** диаграми;
- SADT **не дефинира** явно гледните точки
Гледните точки са **интуитивно разширение** на използваната от SADT техника за моделиране.
- “*Гледните точки*” на SADT са различните *източници* и *приемници (sinks)* на данни.

Пример за библиотека



“Гледните точки” са показани в квадратни скоби

Следващото ниво на декомпозиция свързва с входовете, управлението и изходите на по-детайлното ниво. Декомпозицията се повтаря до пълно изясняване на детайлите.

Controlled Requirements Expression (CORE)

- CORE е разработен за British Aerospace в края на 70-те от system designers (Mullery, 1979). Използва се и в Европейската космическа програма след 1980 г., също European Fighter Aircraft.
- Методът CORE е базиран на подход *за функционалната декомпозиция*
- CORE е базиран на **явно** дефиниране на гледни точки за формулиране на изискванията

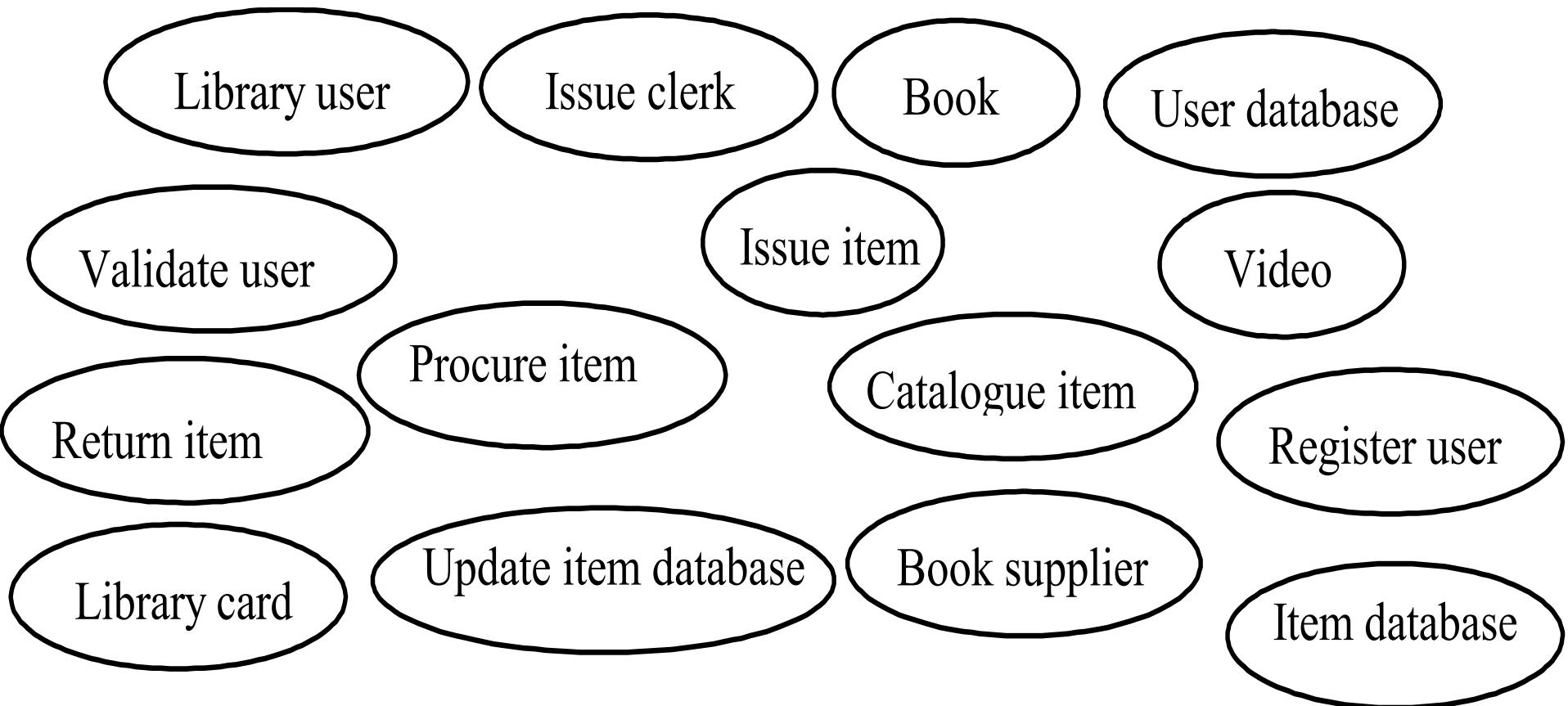
Стъпки на CORE метода

- Методът CORE се състои от 7 итеративни стъпки:
 - Определяне на гледните точки
 - Структуриране на гледните точки
 - Таблично обединяване (Tabular collection)
 - Структуриране на данните
 - Моделиране на гледните точки поотделно
 - Комбинирано моделиране на гледните точки
 - Анализ на ограниченията

Стъпка 1 – Определяне на гледните точки (1)

- *Определяне* на възможните гледни точки на **две нива**:
 1. Обхваща *всички обекти*, които взаимодействат със системата за идентифициране на функционални и нефункционални изисквания.
 - *Няма* строги правила за идентифициране на гледните точки;
 - „*Brainstorming*“ сравнение на мненията м-у отделни аналитици
 2. Прави разграничение между *defining (определящи)* и *bounding (границни)* гледни точки

Пример за библиотека – „първоначални“ гледни точки

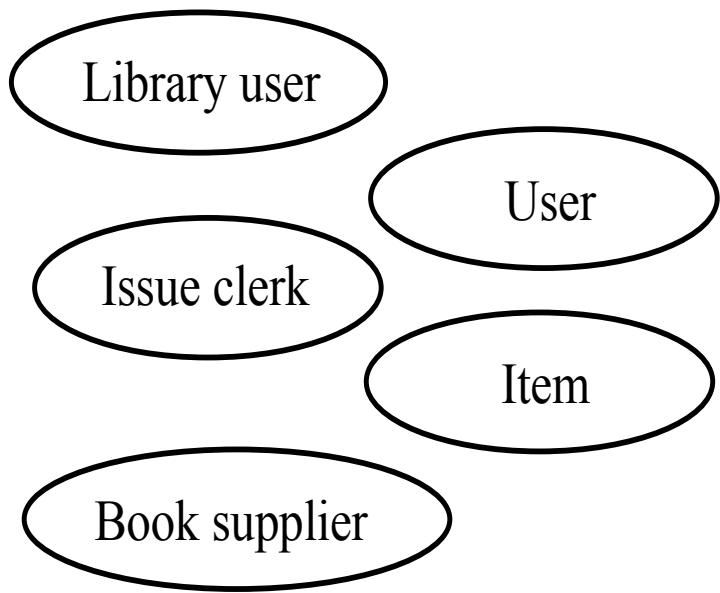


Стъпка 1 – Съкращаване на гледните точки (2)

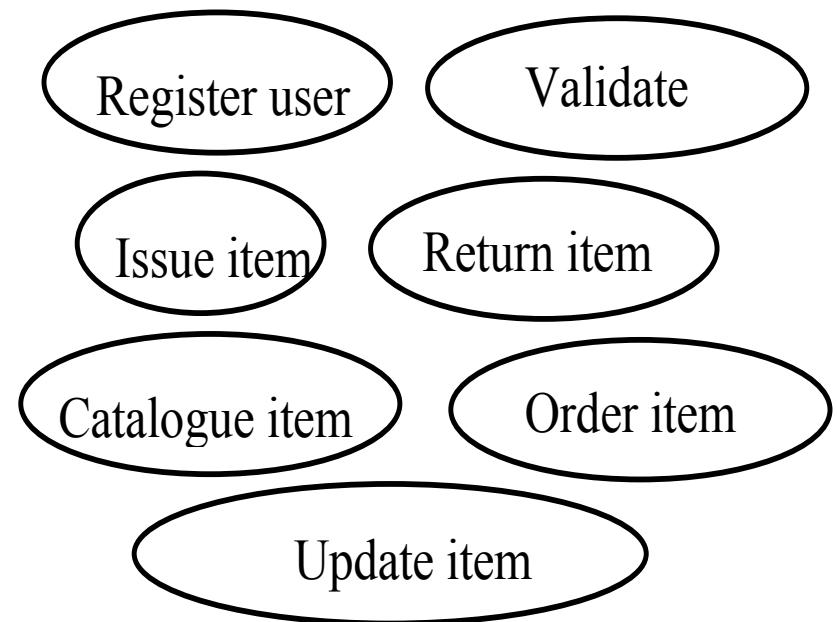
- Всяка елипса представя **най-абстрактната** форма на гледните точки
- Последният етап на идентификация на гледните точки включва съкращаване на идентифицираните гледни точки до *набори* от:
 - *Bounding (Границни) гледни точки:* Обекти, които взаимодействат със системата.
 - *Defining (Определящи) гледни точки:* (Под)процеси на системата, разглеждани низходящо.

Пример: Гранични и определящи гледни точки

Bounding Viewpoints



Defining Viewpoints

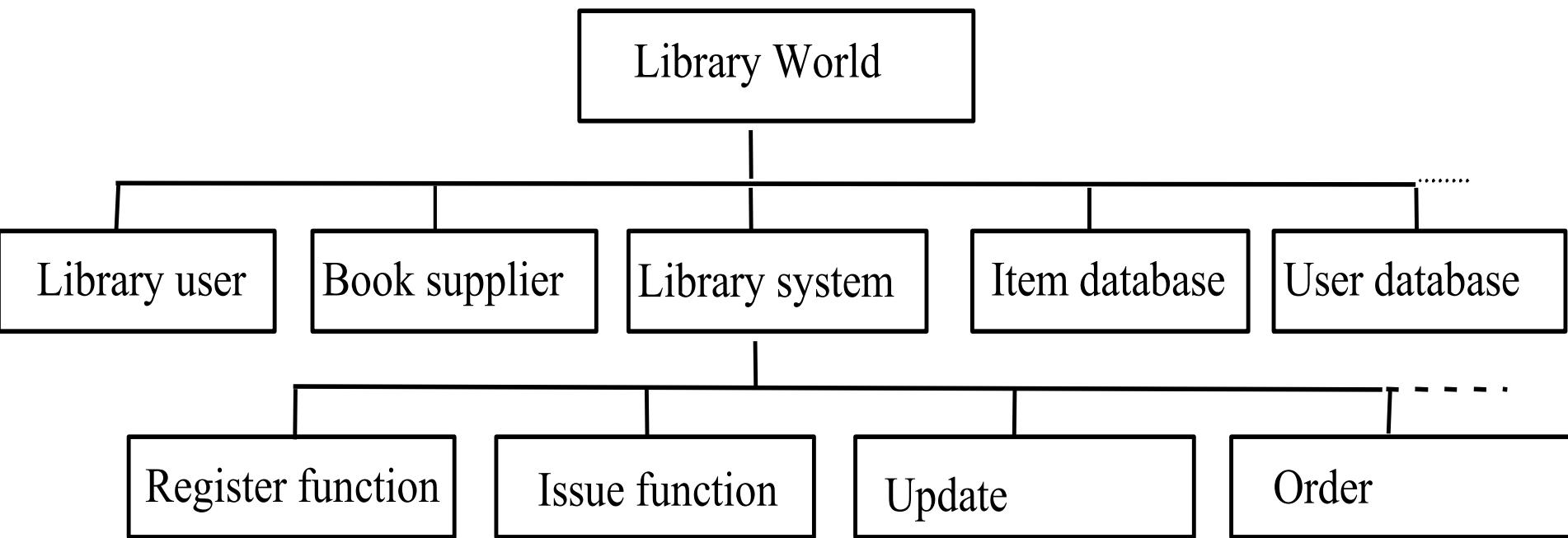


Стъпка 2 – Структуриране на гледните точки

Рамка за записване и за анализиране на гледните точки.

- *Итеративна* (top-down) декомпозиция на системата в йерархия от функционални подсистеми
- Структурно *bounding* гледни точки се поставят на *същото* ниво като целевата система
- *Всяка функционална подсистема представлява отделна гледна точка*

Пр. Библиотечна система – структуриране на гледните точки



Стъпка 3 – Таблична сбирка (Tabular collection)

- Механизъм за *събиране* на информация за дадена гледна точка
- Всяка гледна точка се разглежда последователно по отношение на *действието, което извършива*:
Данните, използвани за тези действия, получените изходни данни, източникът на данните и дестинацията на данните
- *Табличните сбирки имат за цел и да разкрият пропуските и конфликтите* в потока на информацията между отделните гледни точки и така да осигурят консистентност.

Библиотечна система - tabular collection

| Source | Input | Action | Output | Destination |
|--------------|----------------|---------------|------------------------------|-----------------------------|
| Library user | requested item | check item | issued item error message | Library user Issue clerk |
| Library user | library card | validate user | loan default message | Issue clerk |

Стъпки 4-7

- Стъпката (4) на *структурiranе* на данните включва декомпозиция на елементите от данните до *съставните им части* и създаване на *речник на данните*.
- *Моделиране на дейностите на отделните (стъпка 5) и комбинираните (стъпка 6) гледните точки* чрез използване на диаграми на дейностите с нотация като в метода SADT.
- Последната стъпка в CORE включва извършване на *анализ на ограниченията* върху системата като цяло.

Недостатъци на CORE

- Понятието за гледна точка е *слабо* дефинирано
- Анализът се съсредоточава върху вътрешните перспективи – *defining* (определящите) гледни точки
Bounding (границните) гледни точки не се анализират повече от това *да се разглеждат като приемници и източници на данни.*
- Заради вградената структура CORE трудно се интегрира с други методи за изискванията.

Системно инженерство, базирано на гледни точки

/Viewpoint-oriented system engineering (VOSE)/

- Разработен в Imperial College, Лондон в началото на 90-те
- Системното инженерство, базирано на гледните точки, е *рамка за интегриране* на методи за разработка на система.
- VOSE използва гледните точки, за да *разделя и да разпределя дейностите* и знанието на участниците в разработването на софтуера.
- Гледните точки (*in* VOSE) *обхващат* ролята и отговорността на даден участник в определен етап от разработката.

Шаблон за гледните точки

- Една **гледна точка** във **VOSE** може да се счита за шаблон, който описва какво вижда участника:
 - *Стил* - представителна (representative) схема на това, което вижда
 - Област (problem domain)
 - Спецификация (partial)
 - Работен план
 - Работен запис (development history)

Стандартни слотове на шаблон за гледните точки

Style

Definition of representation

Work Plan

Development actions and rules

Domain

Problem domain described by
ViewPoint

Specification

actual partial
specification

Work Record

Development
history

Viewpoint конфигурации

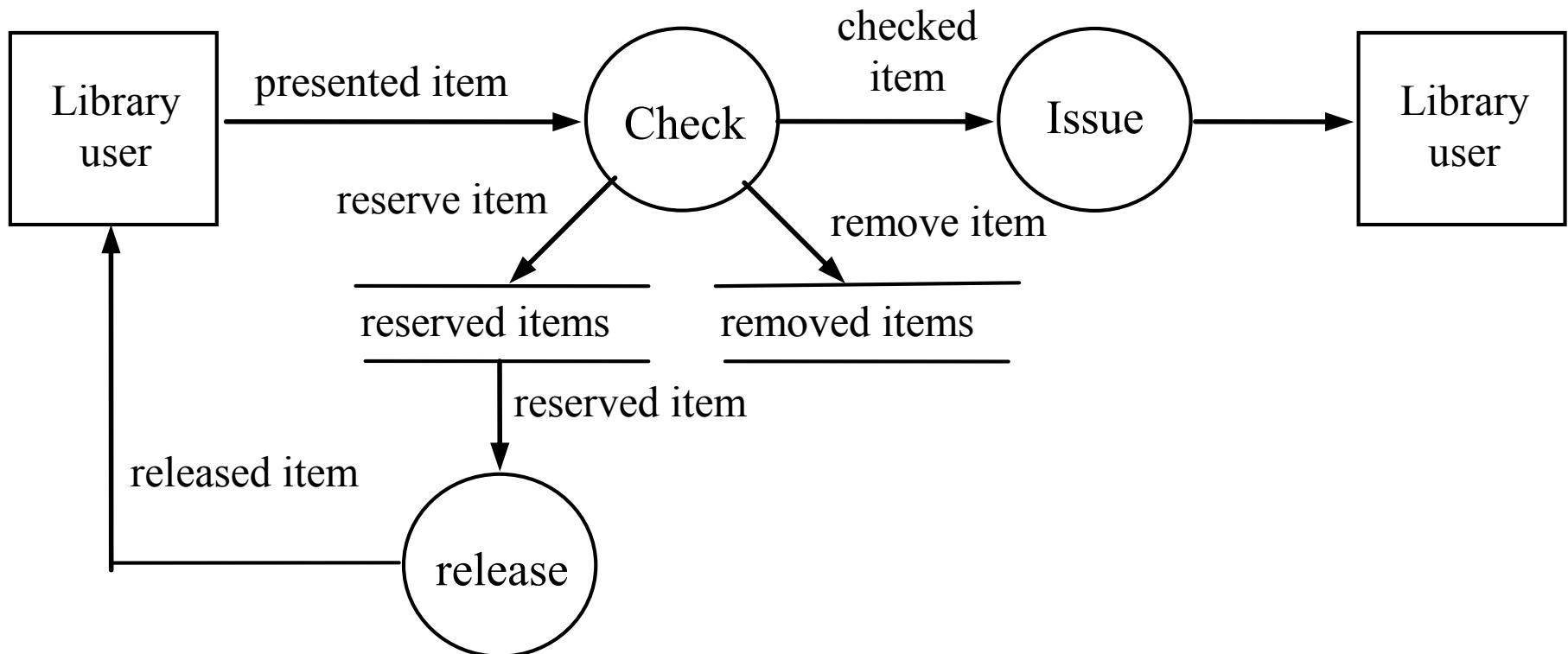
- Гледните точки могат да се организират в *конфигурации* като колекции от свързани гледни точки.
- Една конфигурация може да се състои от
 - Шаблони с *различни стилове*, разглеждащи един и същ дял на проблемната област, или
 - Шаблони с *един и същ стил*, разглеждащи различни дялове на проблемната област
 - *Крайната система е комбинация от конфигурациите*, на които всички конфликти са решени.

Пример: Библиотека

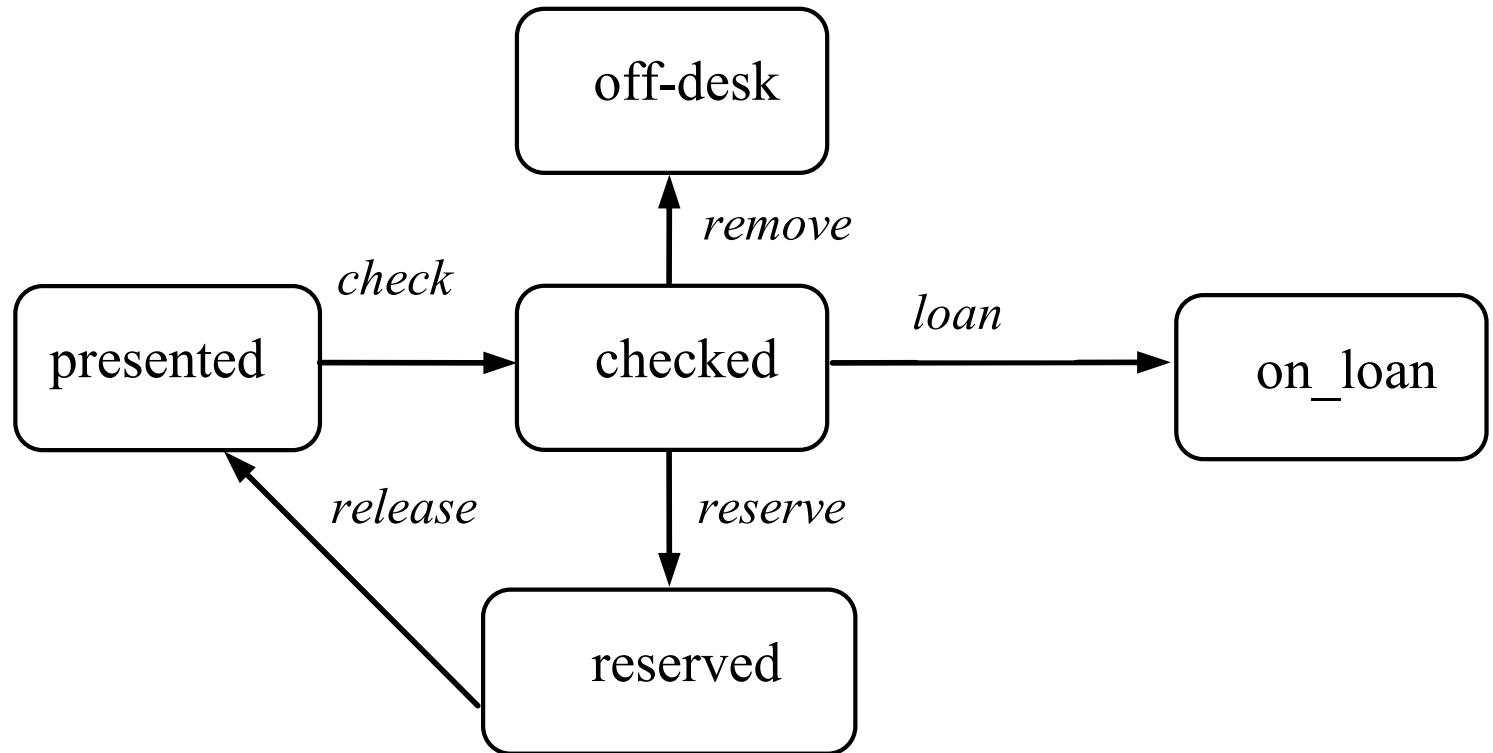
Нека имаме артикул от библиотеката, който потребителят представя за *заемане*, *връщане* или *резервиране*.

- „Библиотечният свят“ може да бъде разделен на области (гледни точки) на *issue desk* и на *library user* (читател).
- Използват се модели на *потока на данните* и *прехода на състоянията*, с които се моделира библиотечният артикул от гледна точка на всяка област.

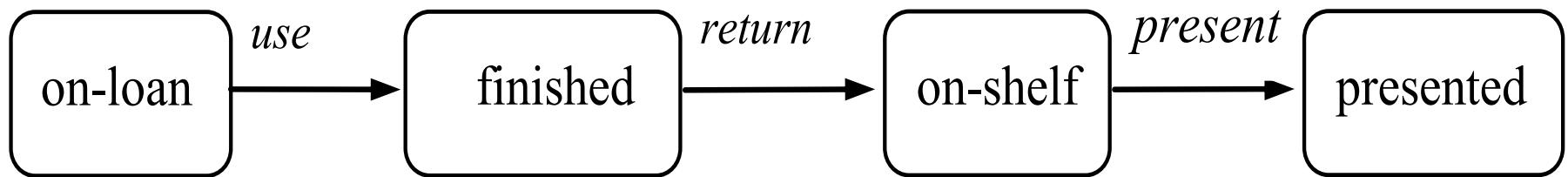
Модел на потока на данните на област на Issue desk с 3 процеса



Модел на прехода на състоянията на област на Issue Desk



Модел на състоянията на библиотечната единица от гл. точка на Library user domain

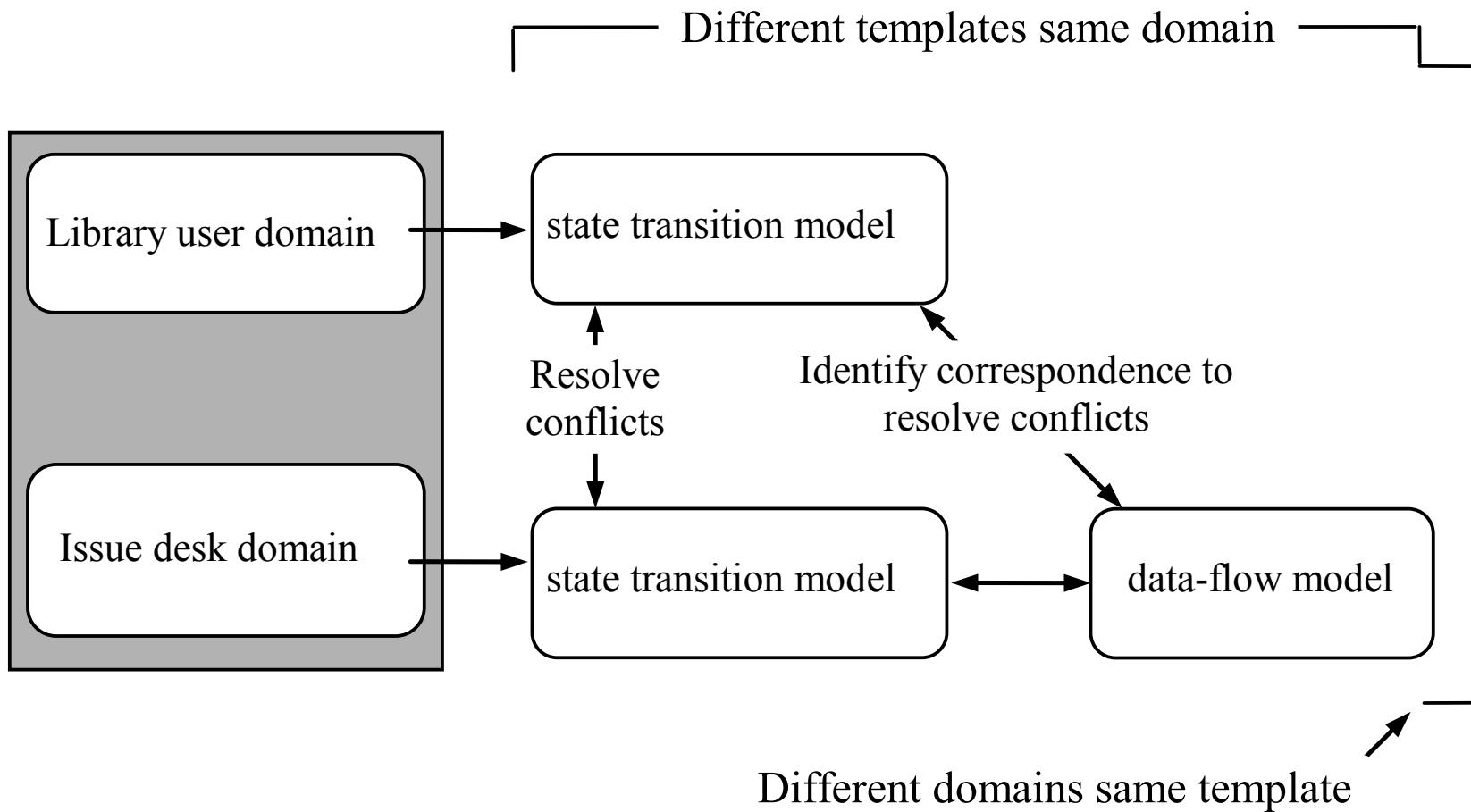


Свързване на изискванията с използване на подхода на гледните точки

(Разрешаване на конфликти)

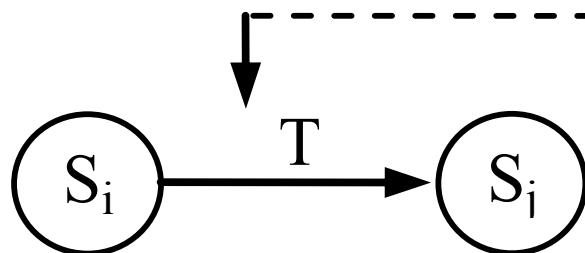
- Важно е, за да се осигури *съгласуваност* между различни представяния на областите
- За еднаквите стилове, виждащи различни домейни, конфликтите се разрешават чрез проверка за *последователност между моделите* т.е. да осигурим последователност в информационните потоци между отделните части на описанията.
- За различните стилове, виждащи еднакви домейни, трябва да се открият *съответствията между схемите за представяне*, за да се улесни проверката за съгласуваност.

Проверка за съгласуваност



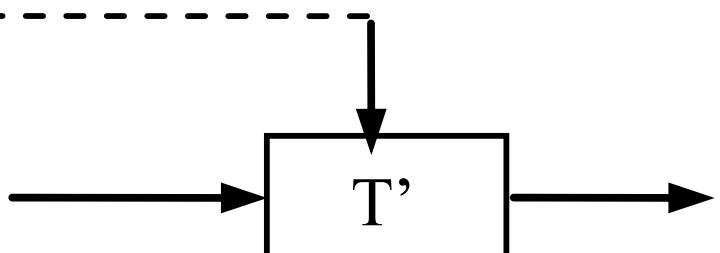
Съответствие между переход и функция

*State transition analysis
analysis*



Transition

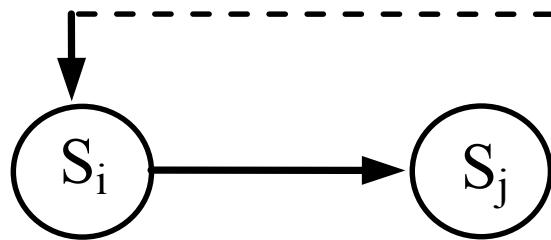
*Data-flow
analysis*



Function

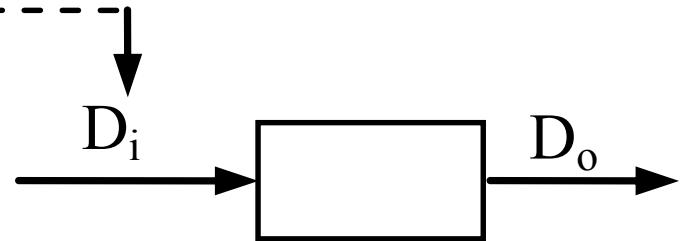
Съответствие между състояние и данни

*State transition analysis
analysis*



State

Data-flow



Data-flow

Mapping different templates, same domains

| Issue desk DFD | Issue desk ST |
|----------------|---------------|
| check | check |
| issue | loan |
| release | release |

Mapping on different domains, same template

Issue desk ST

presented
on-loan

Library user ST

presented
on-loan

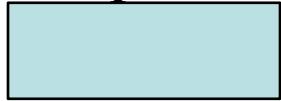
Дефиниране на изискванията, базирано на гледни точки (View Point Requirements Definition (VORD))

- Разработен в университета в Ланкастър през 90-те години.
- Главно предназначен за специфициране на *интерактивни системи*
- Базиран на гледни точки, които се фокусират върху *потребителски проблеми и организационни въпроси*
- **Използва модел, ориентиран към услуги (service-oriented).** Гледните точки са аналог на клиента.
- VORD дефинира **два** главни типа гледни точки: *преки и косвени*

Преки и косвени гледни точки

- **Преки гледни точки**
 - Взаимодействат директно със системата
 - Те съответстват *на клиенти*, които получават услуги от системата и предоставят информация
 - Включват *оператори/потребители* или *други подсистеми* с интерфейс към анализираната система
- **Косвени гледни точки**
 - Не взаимодействат директно със системата
 - Те се „интересуват“ от някои или от всички услуги на системата
 - Генерират изисквания, които ограничават услугите, предоставяни на преките гледни точки
 - Включва гледни точки *на организацията, на околната среда, на проектирането и разработването*

Примери за преки и косвени гледни точки

- Гледна точка на системно планиране, която е засяга бъдещо предоставяне на библиотечни услуги 
- Потребител на библиотеката, който достъпва до системните услуги чрез интернет 
- Търговска гл. т., която описва ефекта на представянето на системата сред персонала и задълженията на персонала в библиотеката 

Подход за валидиране на изискванията, базиран на гледните точки (Leite and Freeman, 1991)

- Използва гледни точки, за да подпомага *ранното* валидиране на изискванията.
- Цел на подхода е *да идентифицира и да класифицира* проблемите, свързани със *завършеността и точността на описанието на системата*

Гледни точки, перспективи и изгледи (views) - дефиниции

- *Гледна точка* се дефинира като позиция (standing position), която някой човек използва, когато изследва обхвата на проблема/системата
- *Перспектива* се дефинира като набор от факти, които се наблюдават и моделират според определен аспект на реалността
- *Изглед (View)* се дефинира като обединение на перспективите
- За представянето на гледните точки се използва *език на гледните точки (VWPL)*

Стъпки на метода

- Включва *поне двама* аналитици (гледни точки), използващи VWPL
- Изглед (View) се конструира чрез описание на проблема с използване на три перспективи - *данните, процеси и актори*
 - Аналитиците използват *is-a* и *part-of* йерархии, за да подобрят своя собствен изглед (view)
- Перспективите и йерархиите се анализират и се съставя ‘списък с несъответствия’ и ‘видове несъответствия’
- Перспективите се обединяват в изглед
 - Expressed in the process perspective together with the hierarchies
- Когато има поне два изгледа се сравняват отделните гледни точки за точност и пълнота.

Процес за валидиране на изискванията, базиран на гледните точки

