

# Инженеринг на изискванията

Лекция 1



# Съдържание

- Основни концепции и дефиниции
- Роля на Инженеринга на Изискванията (ИИ) като част от системния 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

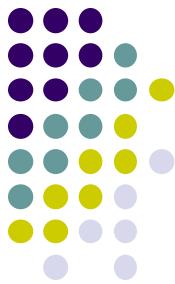


Institute of Electrical and Electronics Engineers  
IEEE Stud. 610.12-1990

**Дефиниция (софтуерно изискване):**

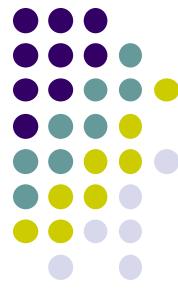
- 1) Условие или способност на софтуера, необходимо на потребителя, за да реши проблем или да постигне определена цел.
- 2) Условие или способност, което трябва да има системата или компонент на системата, за да удовлетвори договор, стандарт, спецификация или друг формално наложен документ.
- 3) Документално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

# Изискване на софтуерна система - 2



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

# 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

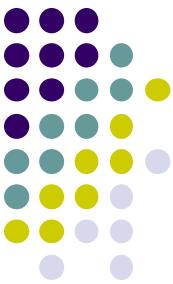
- So far we used words like requirements:
  - capture (обхващам)
  - gathering (събирам в сбирка, колекция ...)
  - trawling (трапене)
  - elicitation (извлечане, изваждане, разкриване)

# Изискванията се откриват, а не намират - 2

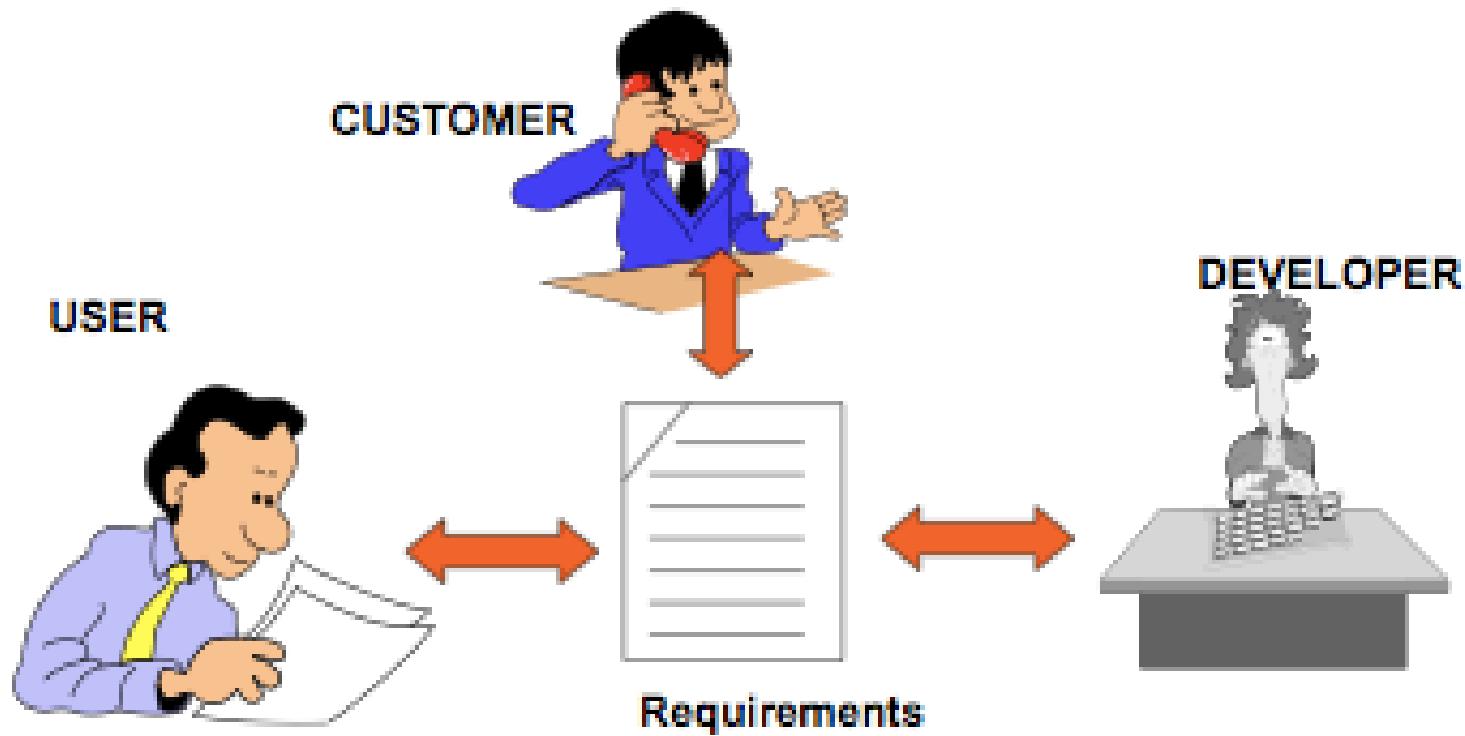


- “Откриване” означава много и различни дейности:
  - Нови идеи (being open to new ideas);
  - Креативност (applying creative effort);
  - Работа в екип (working as a team);
  - Фокусирани въпроси (asking questions that focus the search);
  - Да се намери специфичното (intending to find particular kinds 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!

# ... и още защо са важни изискванията?



Те са единствената част, която всички разбират.  
Основа за бъдещата разработка.





## Когато се изгражда софтуерна система се задават следните основни въпроса:

- 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?

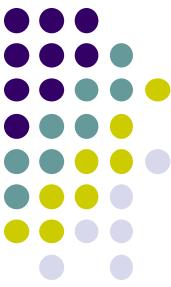
# Процес на инженеринг на изискванията (Requirements Engineering process) -1



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

*Ian Summerville*

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



# Инженеринг на Изискванията - класическото понятие

- **Дефиниция** (Инженеринг на изискванията): Прилагане на систематичен, дисциплиниран, количествено измерим подход към спецификацията и управлението на изискванията
  - Приложение на инженерството към изискванията.
- **Цел на ИИ: пълни, недвусмислени** изисквания преди проектирането
  - Спецификацията на идеалните изисквания трябва да описва бъдещата система (или промяна в системата) толкова пълно и точно, че разработчиците да могат да внедрят системата, без да искат обратна информация.

*However*

*Needs: paper, process*

*Reality check: Does this always work? 11*



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



Процес на разработка на софтуерния продукт



# ИИ (RE) – същност 1:

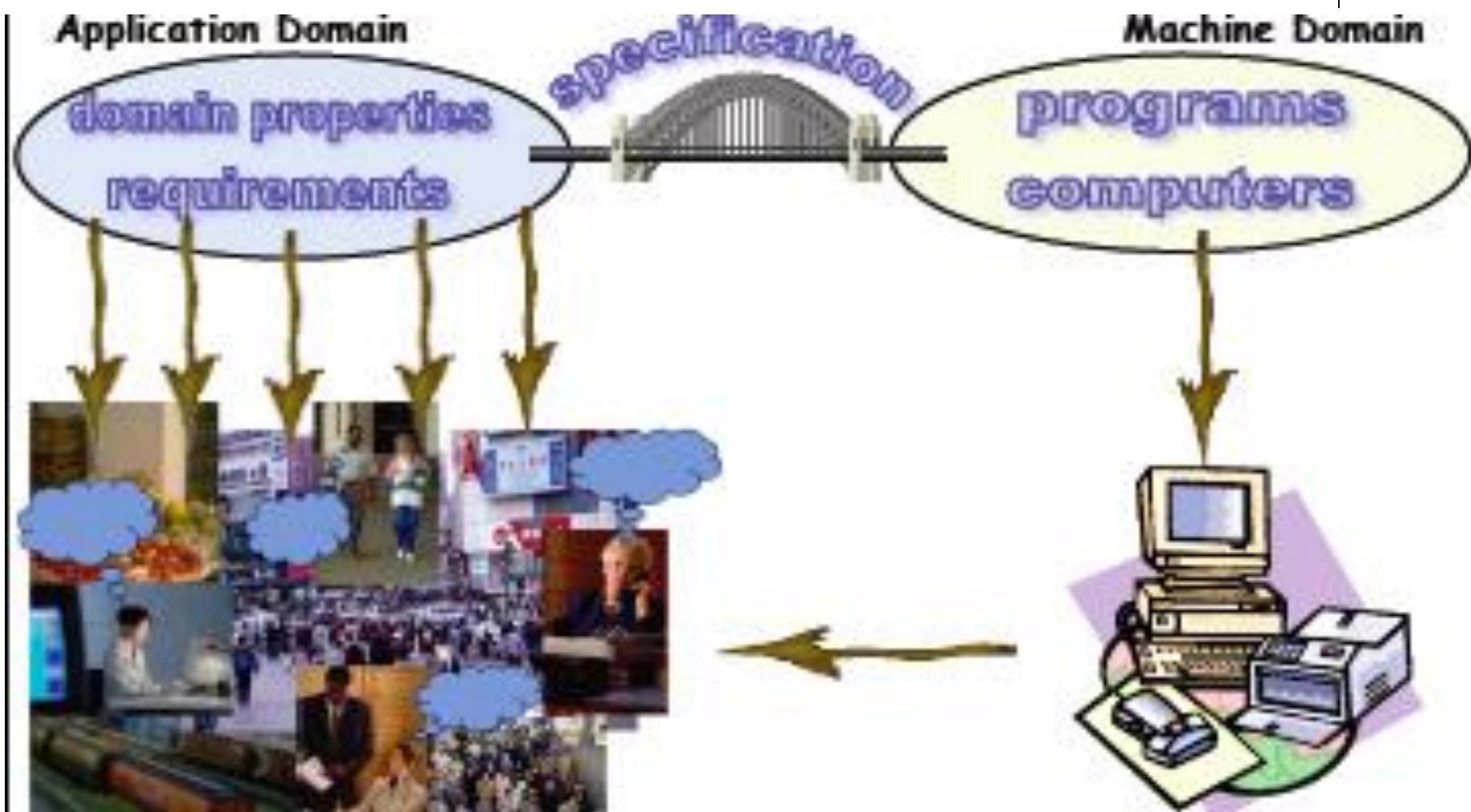
RE is a set of activities concerned with identifying the purpose of a software system, and the contexts in which it will be used. Hence, **RE acts as the *bridge* between the real world needs of users, customers, and other constituencies affected by a software system**, and the capabilities and opportunities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

**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.

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

Requirements are **partly about what we need ...**

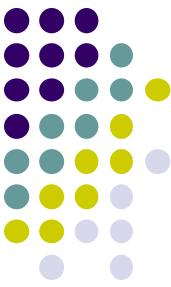
**... and partly about what is possible.**

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

Why it is “requirements” ?

Why it is “engineering” ?

(Kotonya, Sommerville, p.8)



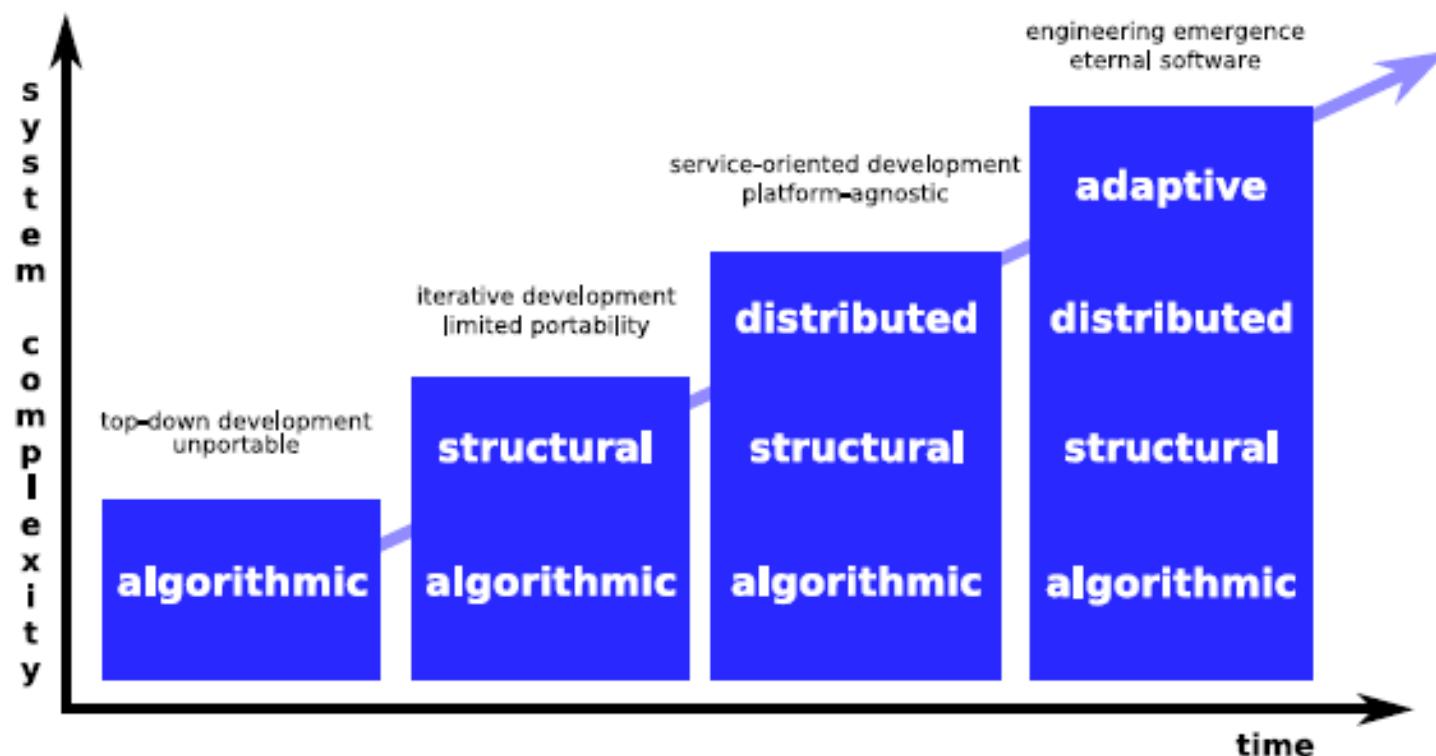
# Software-intensive systems – 1

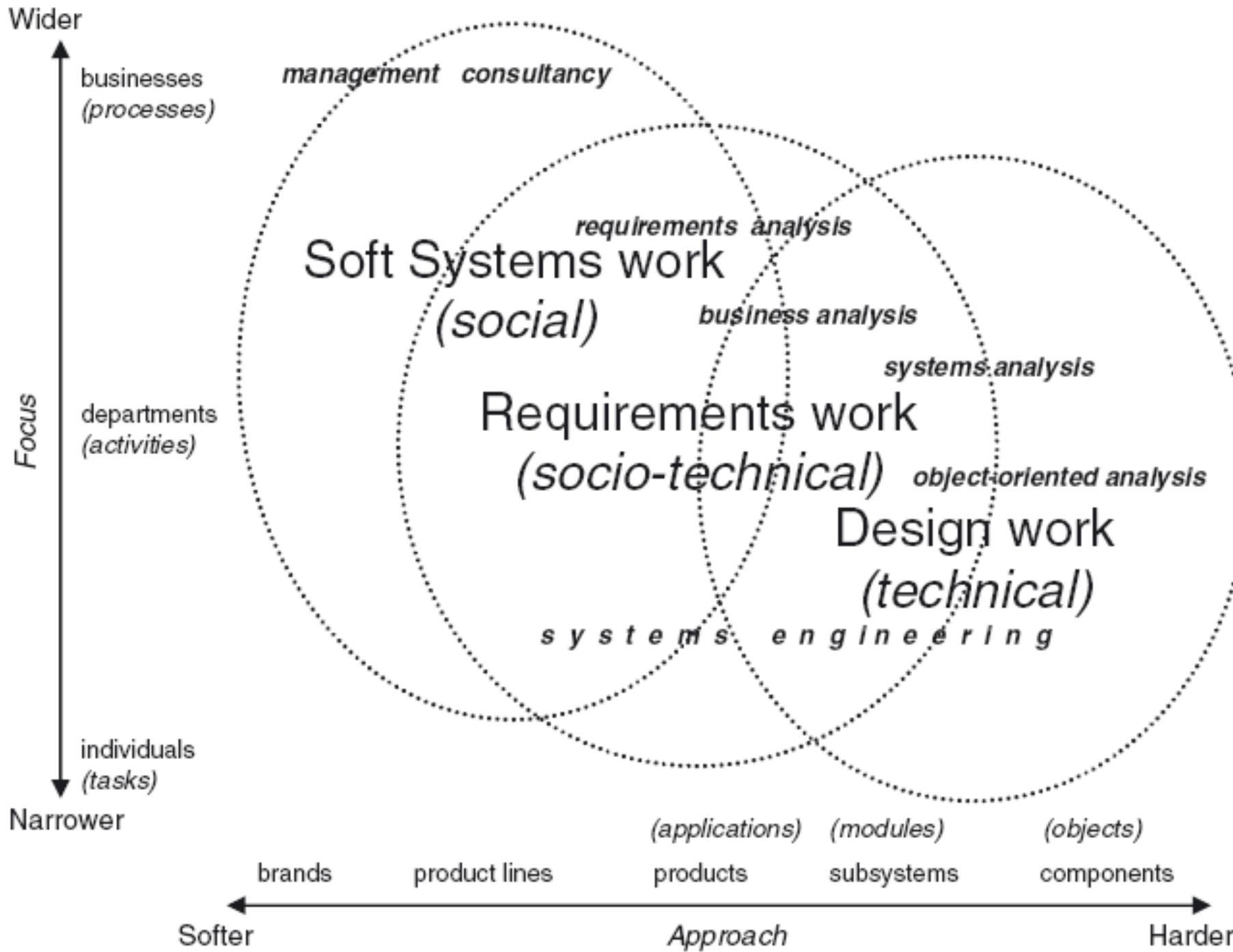
- Software (on its own) is useless
  - Software is an abstract description of a set of computations
  - Software only becomes useful when run on some hardware
  - Software + Hardware = Computing system
- A computer system (on its own) is useless
  - Only useful in the context of some *human activity* 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







# Какво е спецификация на изискванията?

What is a requirement specification?

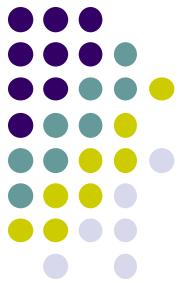
# Спецификация на изискванията - 1



- 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 съвкупност от свързани, зависими елементи на изискванията, включващи дефиниции, цели, обосновки, измервания и друга информация**

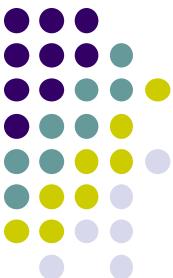
и още:

- ‘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.



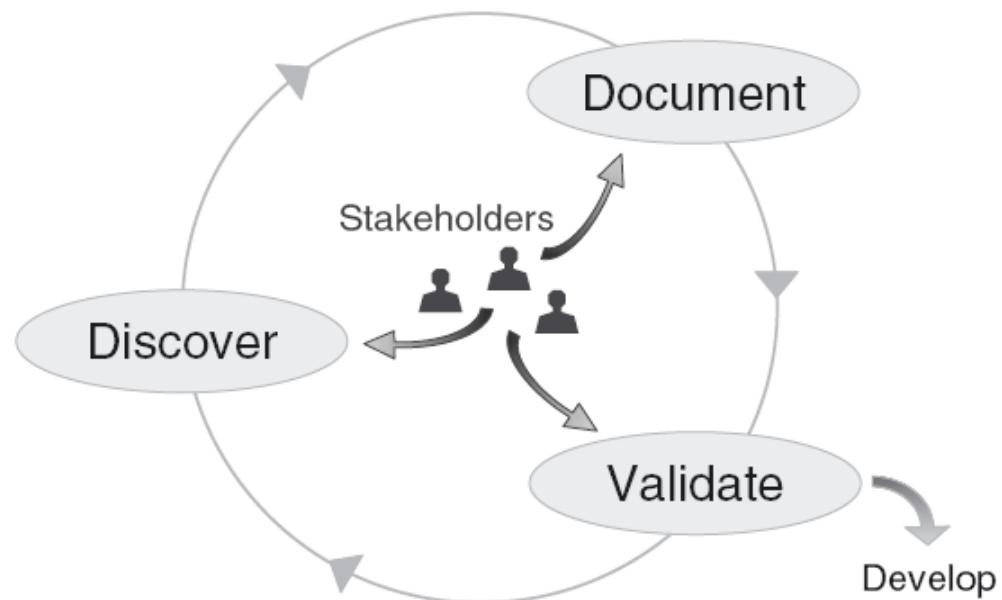
# Discovery as search

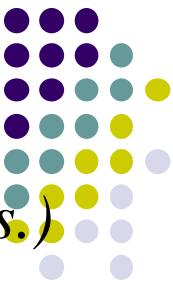
- Начин на мислене (A good way of thinking about discovery is as a search).
- Структуриране на знанието. (The structure of what you know drives what you discover next).
- **Колкото по-добре е организирано знанието за изискванията, толкова по-добре ще може да се открие нова, което наистина е необходимо.**
- *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 проучването

- Общийят процес може да бъде начертан като цикъл на проучване
- Цикълът на проучване е повече или по-малко това, което казва:
  - (Цикъл от) дейности в екип за формулиране на решение
  - Проверка на ефективността на решението



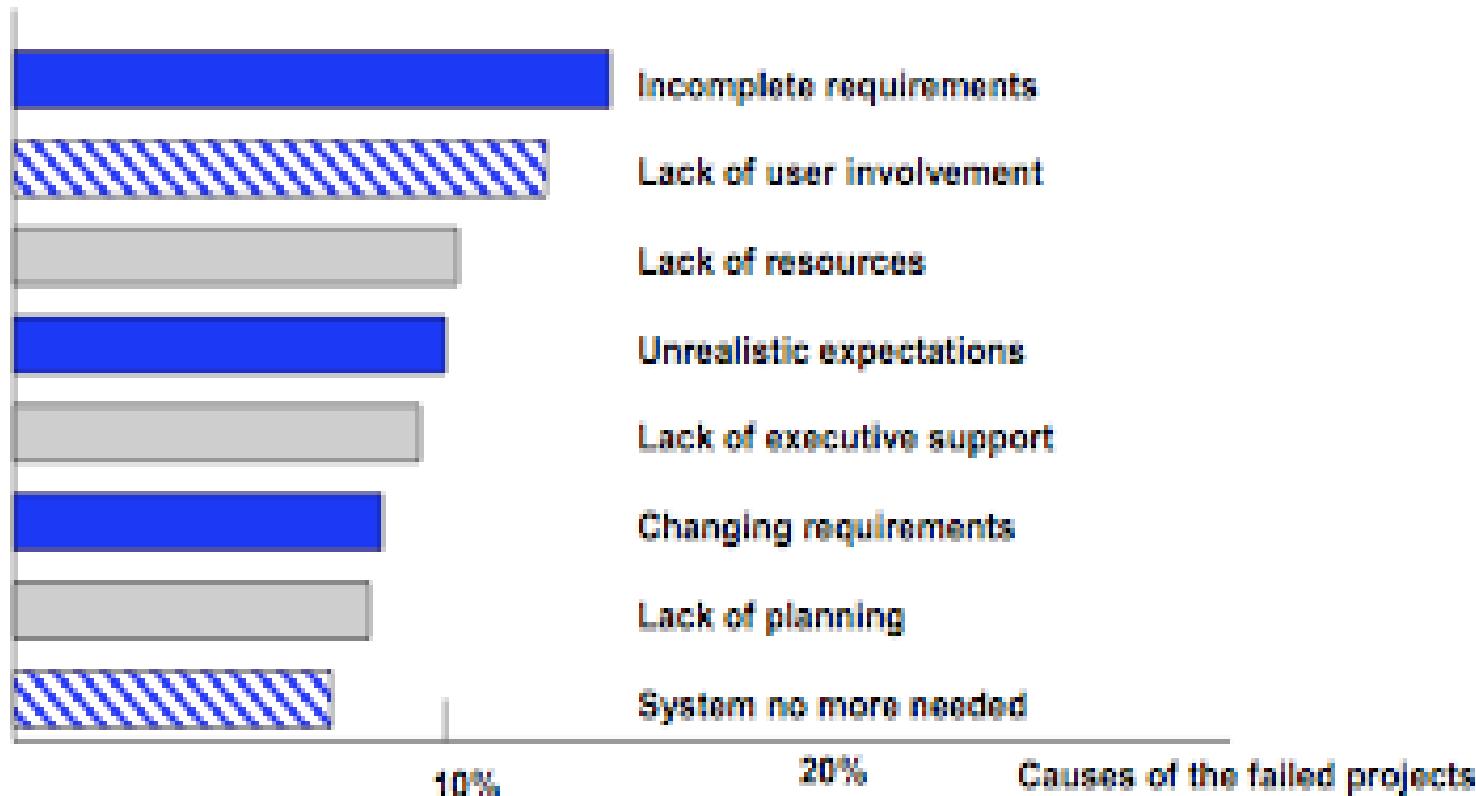


# Защо изискванията са (много) важни?

- Колко струва ИИ? (*About 15% of system development costs.*)

**НО !**

- *Непълните и неконсистентни изисквания* са най-честите причини за проблеми на системата.

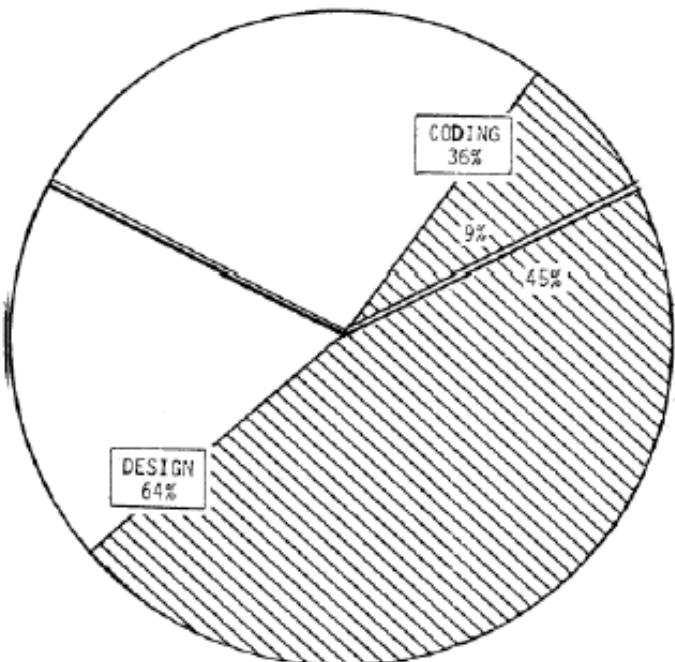




# Мотивация за ИИ - 1

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

Wording by [Endres2003]

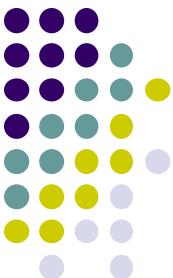


Barry Boehm  
[Image: Wikipedia]



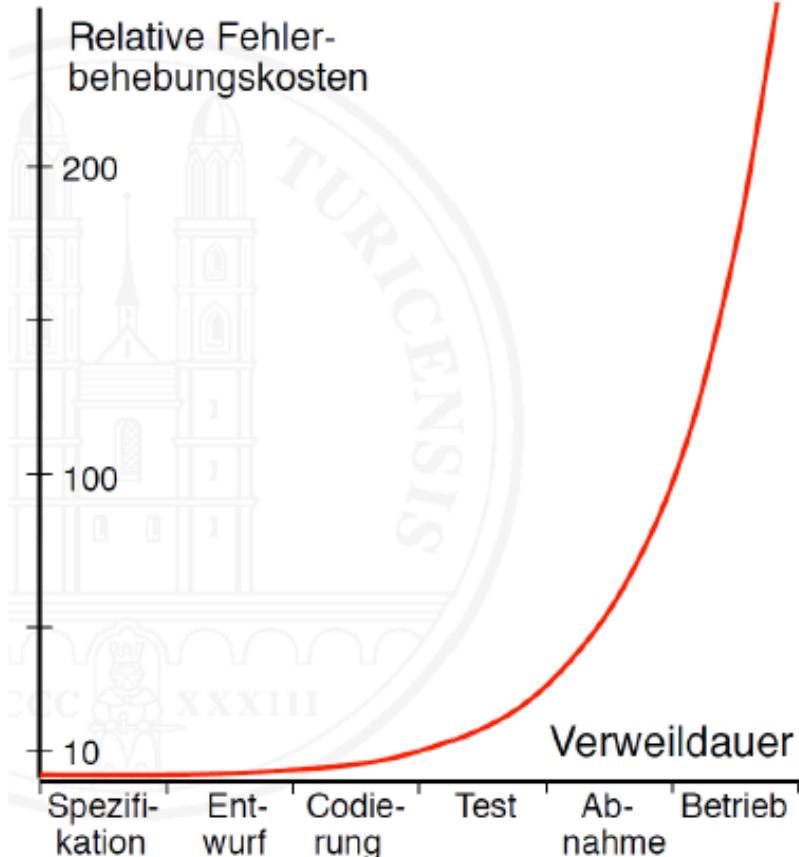
Design: Errors fixed by changing the design  
Coding: Errors fixed by changing only the code

From [Boehm 1975]



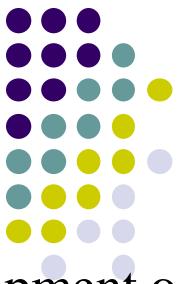
# Мотивация за ИИ - 2

## “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

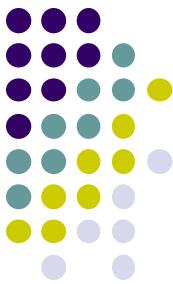
# Мотивация за ИИ - 3



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

- **Топ три фактора за успех са:**
  - участие на потребителите;
  - подкрепа на изпълнителното управление;
  - ясно/точно записани изисквания.
- **Първите три фактора, водещи до провал, са:**
  - липса на информация от потребители;
  - непълни изисквания и спецификации;
  - променящи се изисквания и спецификации

# Каква е връзката между изисквания и проектирането?



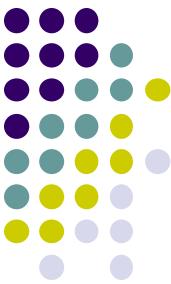
- Два основни (*парадоксални* ☺) принципа:

“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 (описанието на системата) 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 ☺.

# Systems Engineering vs. Software Engineering

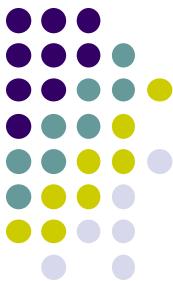


Инженеринга на изискванията се прилага както към софтуерни системи, така и към системите изобщо.

- В софтуерното инженерство обикновено се приема, че начинът, по който софтуерът взаимодейства със света, е фиксиран, като се използват стандартизириани входни и изходни устройства.
- Задачата в системното инженерство е да се проектира цяла система, от която софтуерът е само един компонент.



# Какво правят инженерите по изискванията?

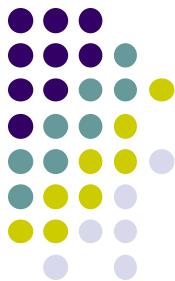


## Какво правят инженерите по изискванията ? - 1

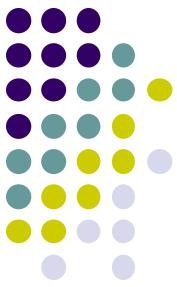
- Изявяват се в началото на проекта
  - Идентифицират проблем, който се нуждае от решение, резултат от:
    - неудовлетвореност от текущото състояние
    - нова възможност за бизнес
    - потенциал за спестяване на разходи и оптимизиране на бизнеса
  - *Анализът* на изисквания е агент на промяната

# Какво правят инженерите по изискванията?

-2



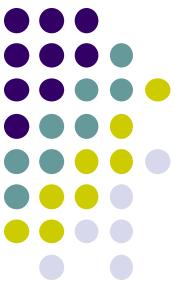
- Инженерите по изискванията трябва да:
  - Идентифицират проблема/възможността
    - Which problem needs to be solved? (identify problem boundaries)
    - Where is the problem? (understand the Context/Problem domain)
    - Whose problem is it? (identify stakeholders)
    - Why does it need solving? (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
  - Мениджмънт
  - Маркетинг
  - Управление на взаимоотношенията с клиенти
- 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



## ■ 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



# FAQS about requirements

- What are requirements?

[REDACTED]

- What is requirements engineering?

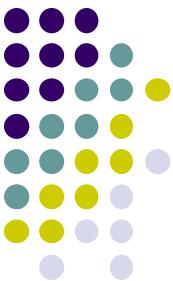
[REDACTED]

- How much does requirements engineering cost?

[REDACTED]

- What is a requirements engineering process?

[REDACTED]



# FAQs contd.

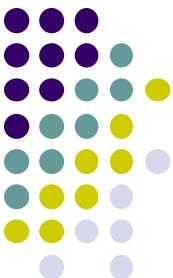
- What happens when the requirements are wrong?
- What is a requirements document?
- What are system stakeholders?
- What is the relationship between requirements and design?



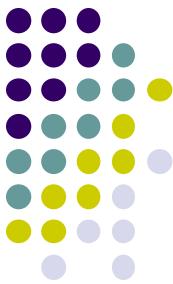
# Процес на инженеринг на изискванията

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



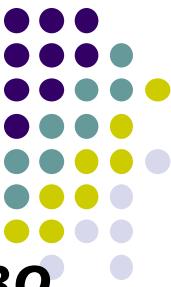
# Процес

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

- **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.

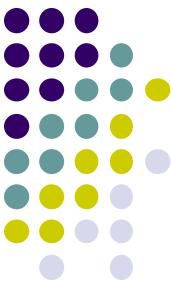
*Пр. : Кои са входовете и изходите на процесите, описани в:*

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

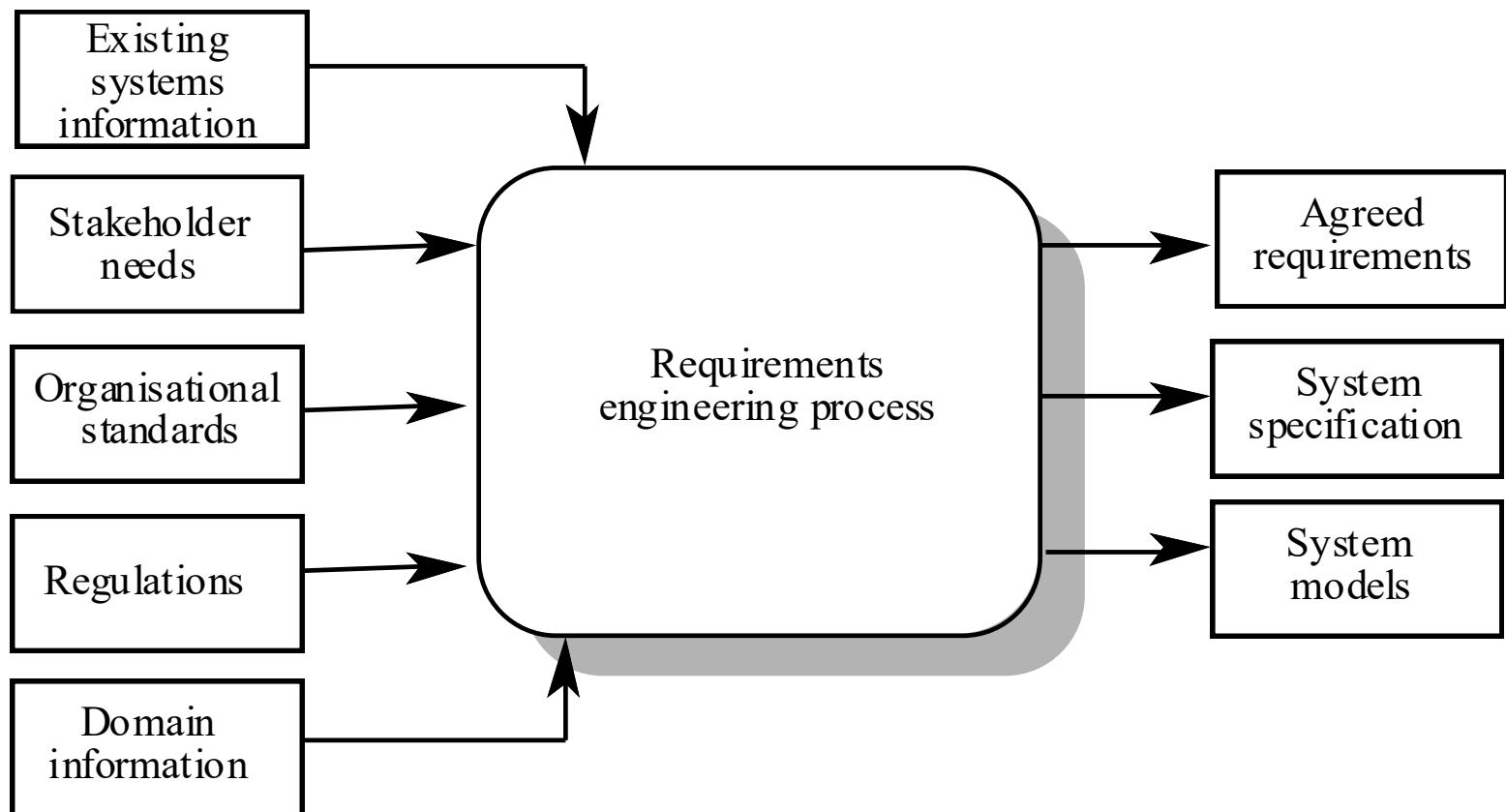


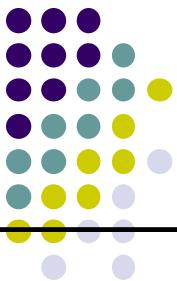
# Процес на проектиране

- Процесите на проектиране изискват *творчество, взаимодействие между много 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**



# Процесът на инженеринг на изискванията е процес на проектиране - описание тип „черна кутия“





# Входове и изходи на описанието

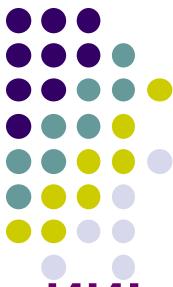
<b>Input or output</b>	<b>Type</b>	<b>Description</b>
Existing system information	Input	Information about the functionality of systems to be replaced or other systems which interact with the system being specified
Stakeholder needs	Input	Descriptions of what system stakeholders need from the system to 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

## **Пример: Специфицирайте по една от 5-те вида входна информация за ИИ на следната система:**

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.

Build a software-controlled system for managing the access of skiers to the chairlifts.





# Вариабилност на процеса по ИИ

Няма идеален процес на ИИ!

- Процесът на ИИ е различен в голяма степен в отделните организации
- Фактори за промяната на процеса на ИИ
  - *Техническа зрялост*: технологии и методи, използвани за ИИ
  - *Организационна култура*: други дейности, 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 и описание за процеса.** (зашо?)
- Видове модели на процес на ИИ:

## A. Според детайлността:

Модел на дейностите (Activity model)

А.1 Модели на общо описание на дейностите (Coarse-grain activity models)

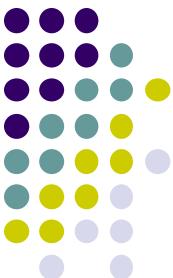
А.2. Модели на детайлно специфицирани дейности (Fine-grain activity models)

## Б. Според перспективата:

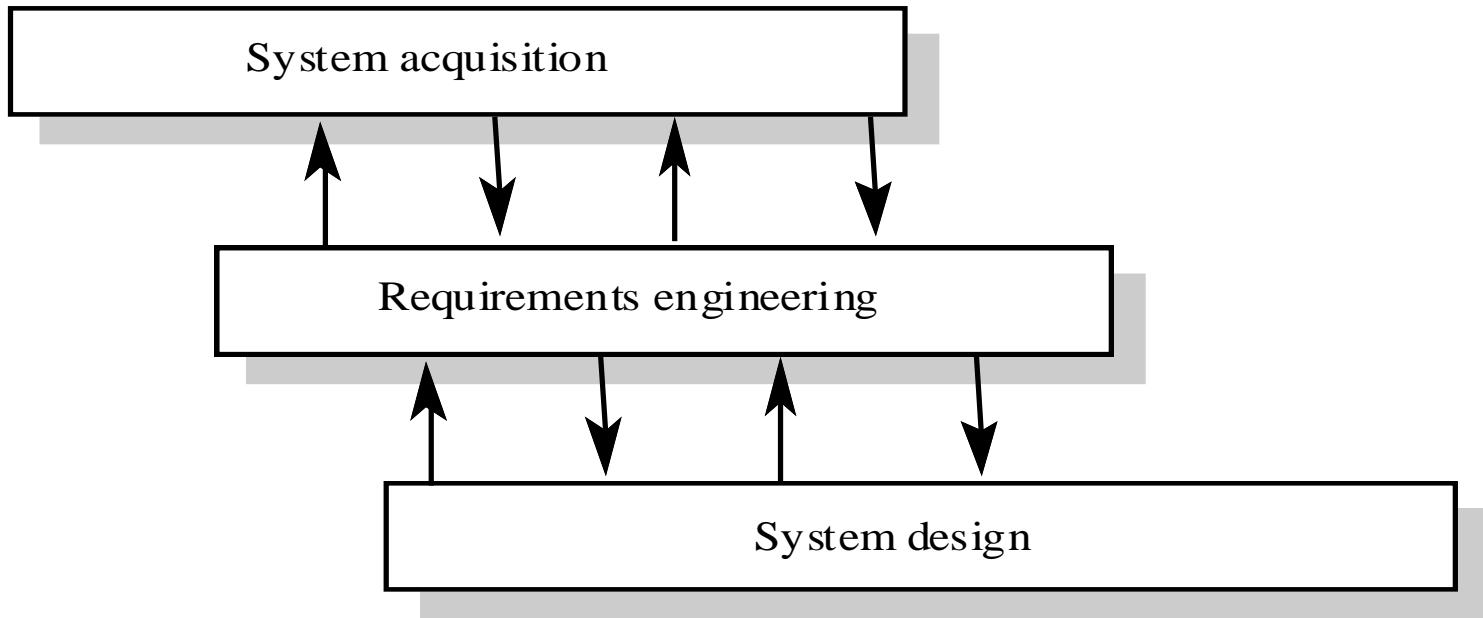
Б.1. Модели роля-действие (Role-action models)

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

От какво зависи изборът на модел?



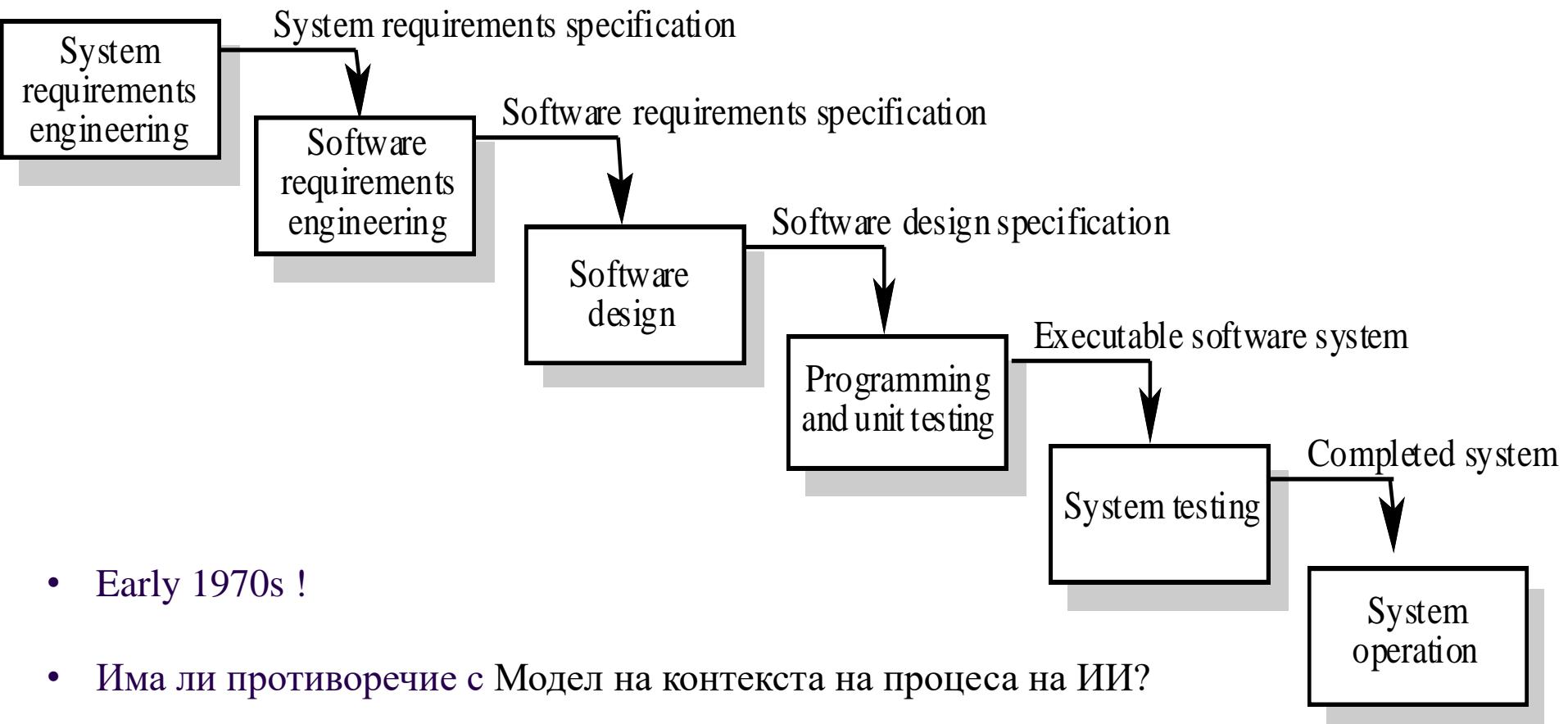
## A.1 пример: Модел на контекста на процеса на ИИ дефинира информационните потоци



(Част от) V-модел на софтуерен процес



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

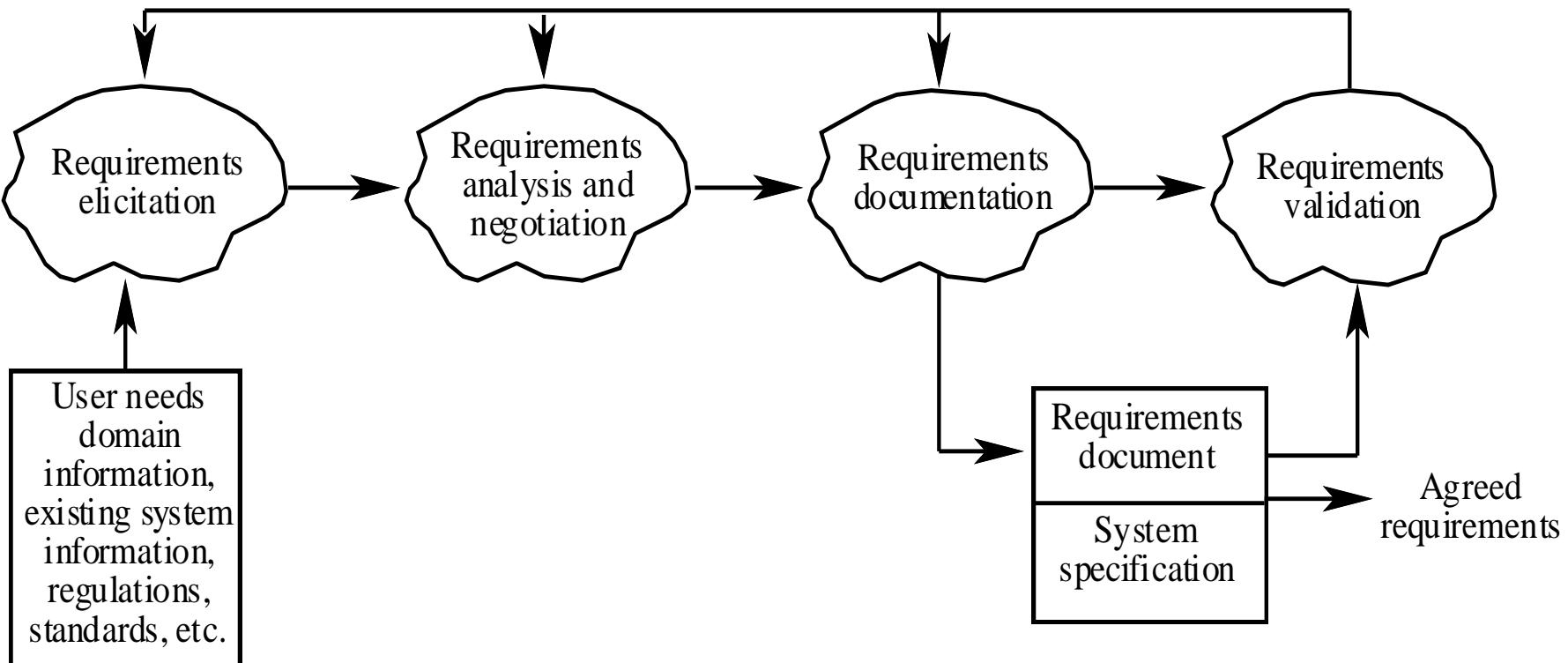


- Early 1970s !
- Има ли противоречие с Модел на контекста на процеса на ИИ?

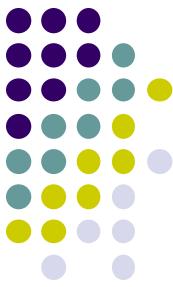


## A.1 Пример: Модел на общо описание на дейностите на ИИ (Coarse-grain activity model of RE)

- общовалиден модел. Защо?



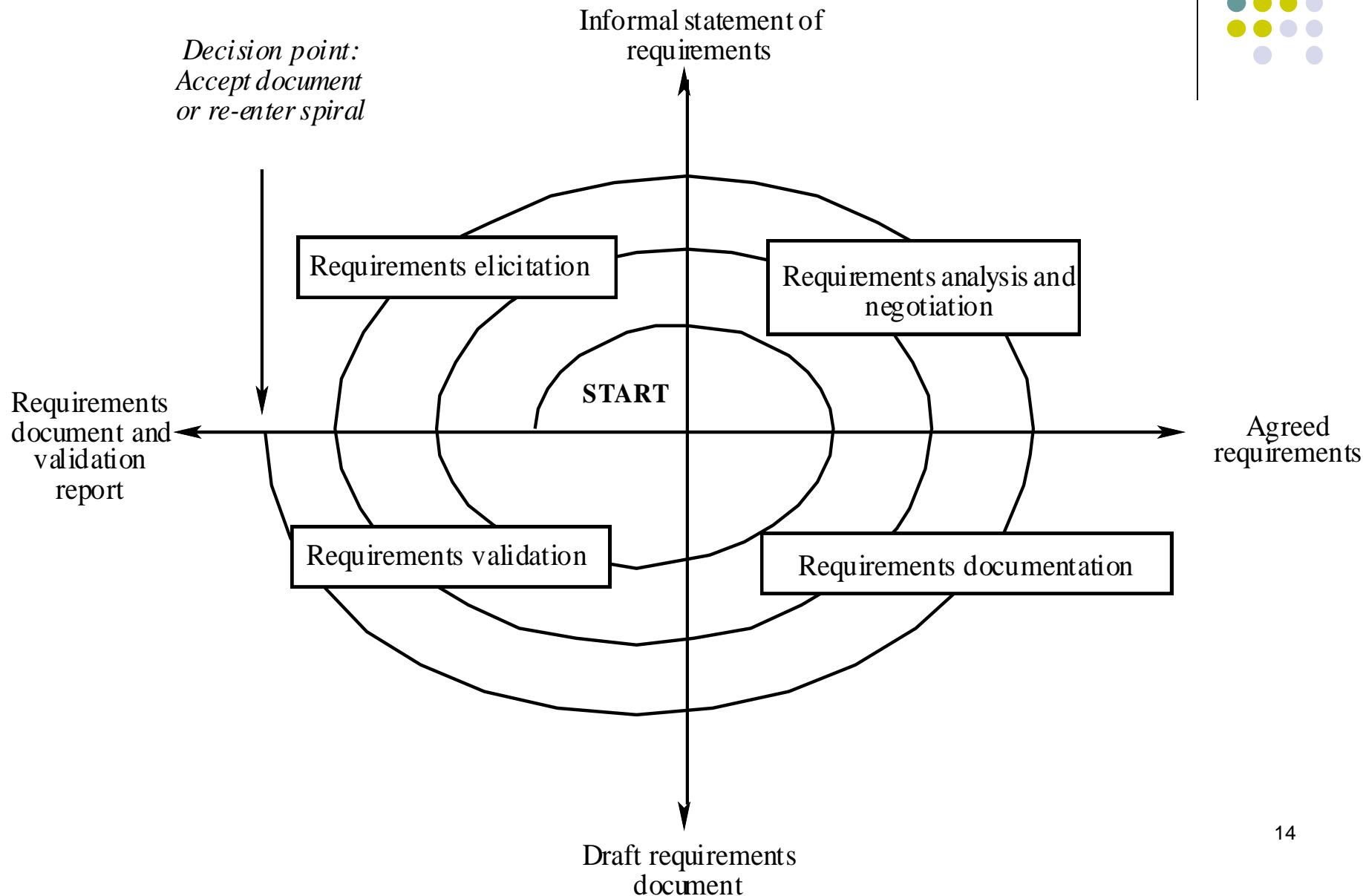
Cloud icon – why?!



# Действията на процеса на ИИ

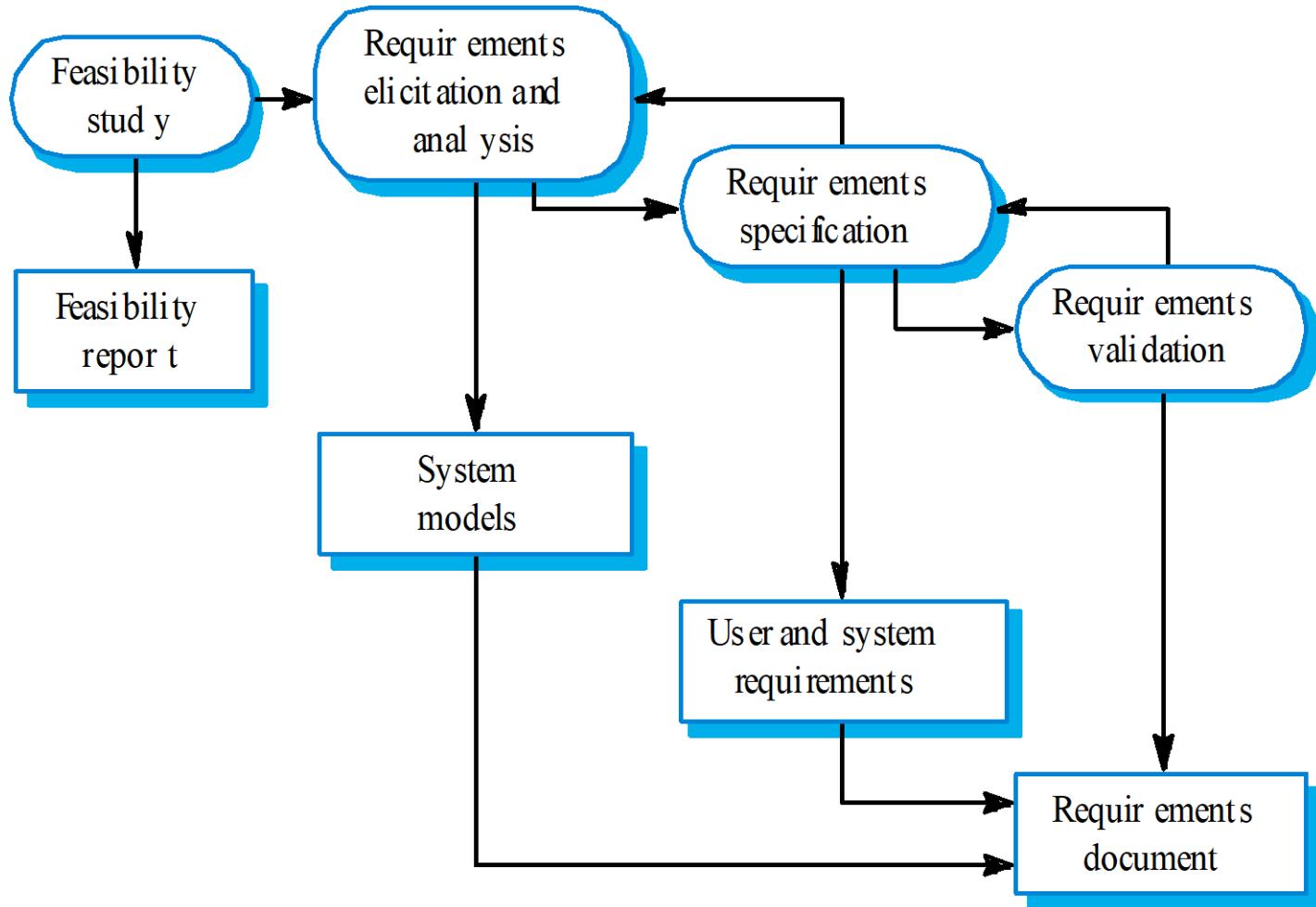
- **Извличане на изискванията (*Requirements elicitation*)**  
Изискванията се откриват чрез консултации със заинтересованите страни, и други техники, знания и информация.
- **Анализ на изисквания и преговори на изискванията (*Requirements analysis and negotiation*)**  
Изискванията се анализират; конфликтите се решават чрез преговори.
- **Документиране на изискванията (*Requirements documentation*)**  
Оформя се 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

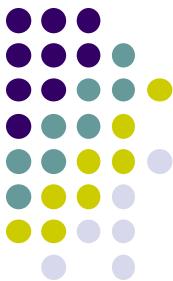


# Изследване за приложимост (Feasibility Studies)



Събиране на информация, отнасяща се до следните въпроси:

- Може ли системата да допринесе за общите цели на организацията?
- Възможно ли е да се постигнат целите на проекта?
  - Има ли *rиск* с висока степен на вероятност, който прави проекта нереализуем?
  - Може ли системата да се *реализира* чрез използване на наличните технологии и в рамките на определените разходи и график на ограничения?
  - Може ли системата да се *интегрира* с други системи, които вече са на място?
  - Може ли да се постигне *споразумение* съобразно общия контекст на работата?



# Извличане и анализ на изискванията – 1 (Requirements Elicitation and Analysis – 1)

- **Извличане на изискванията:**

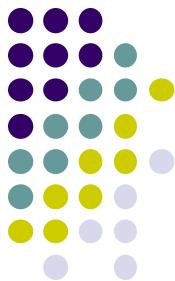
действие по изучаване на проблемите в областта и извлечане на изискванията, както и разбиране на необходимите стандарти, нормативи и други отбелязани документи

- Анализ на източниците на изискванията

- Задачи:

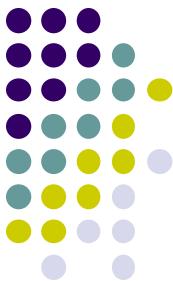
- разбиране за областта на приложение
- идентифициране на заинтересованите страни
- разговори/дискусии за целите на извлечането
- събиране на информация от други източници
- идентифициране и съгласуване на границите на системата

# Извличане и анализ на изискванията - 2



## ● Анализ на изискванията:

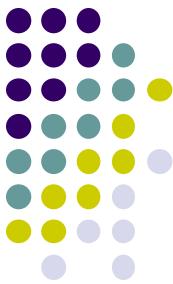
- оценка на извлечените изисквания
- *идентифициране и разрешаване на противоречията между страните*
- запазване на проекта да *не надхвърли рамките на своя обхват*
- прецизиране на събраната информация.  
(making the gathered information more precise).



# Спецификация/документиране на изискванията

- Дейността по *специфициране* осигурява документ със запис на резултатите от извлечане и анализ в документ, който по същество и усъвършенства и доразвива резултатите от фазата на анализ
- Усилия в две различни нива:
  - Заинтересованите страни трябва да се *споразумеят* за документите (договор + други необходими документи)
  - Разработчиците трябва да бъдат в състояние да *използват* документа, за да проектират/реализират системата (договор + документация на системата)
  - Степен на формалност и детайлност - зависи от подхода

# Спецификация на изискванията



- **Дефиниране на изискванията**
  - Изявление на естествен език плюс диаграми на услугите, които системата предоставя, както и оперативните ограничения.  
*Разбирам за клиенти*
- **Спецификация на изискванията**
  - Структуриран документ с подробни описание на системните услуги. Написана е като *договор между клиент и изпълнител*
- **Софтуерна спецификация**
  - Подробно описание на софтуер, който може да служи като основа за проектиране или изпълнение. Написано за *разработчици*

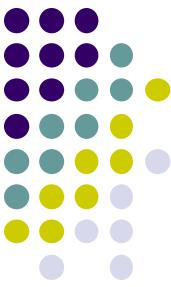
# Валидиране на изискванията

## Requirements Validation



Да покаже, че изискванията действително определят система, която клиентът желае

- Работата се извършва върху цялостен документ на изисквания спецификация (complete draft of requirements specification)
- **Различни видове проверки на изискванията:**
  - валидност
  - съответствие/последователност
  - пълнота
  - точност
  - реализъм и необходимост
  - проверимост (verifiable)



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

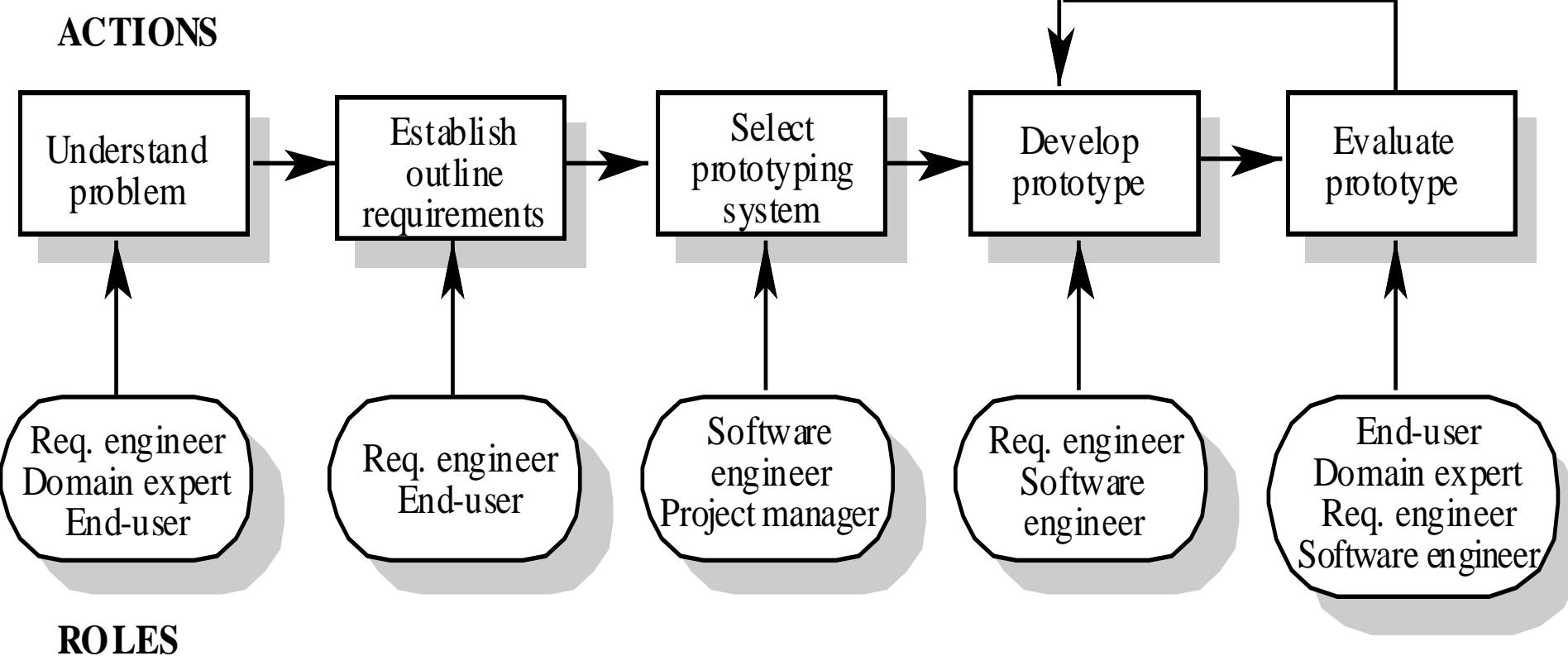
- **Актьори в процеса** са хората, които участват в ИИ.
- Актьорите обикновено се идентифицират с *техните роли в изпълнението на конкретна дейност*, а не персонално т.е. project manager, system engineer ...
- ИИ включва актьори, които
  - 1) се интересуват от проблема, за да бъде той решен (крайни потребители)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

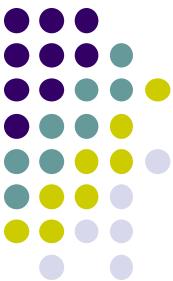
*Примери:* Дайте примери на актьори за системата на седалков лифт. Ниво на креативност?

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



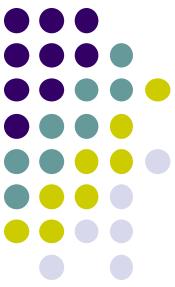
**Б.1 Пример - 1: Диаграма на ролите за създаване на прототип (RAD for software prototyping)**





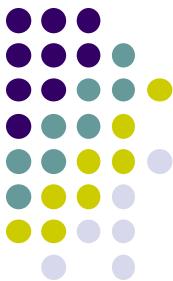
## Б.1 Пример - 2 : Описание на ролите -2 (Role descriptions)

<b>Role</b>	<b>Description</b>
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
- ...

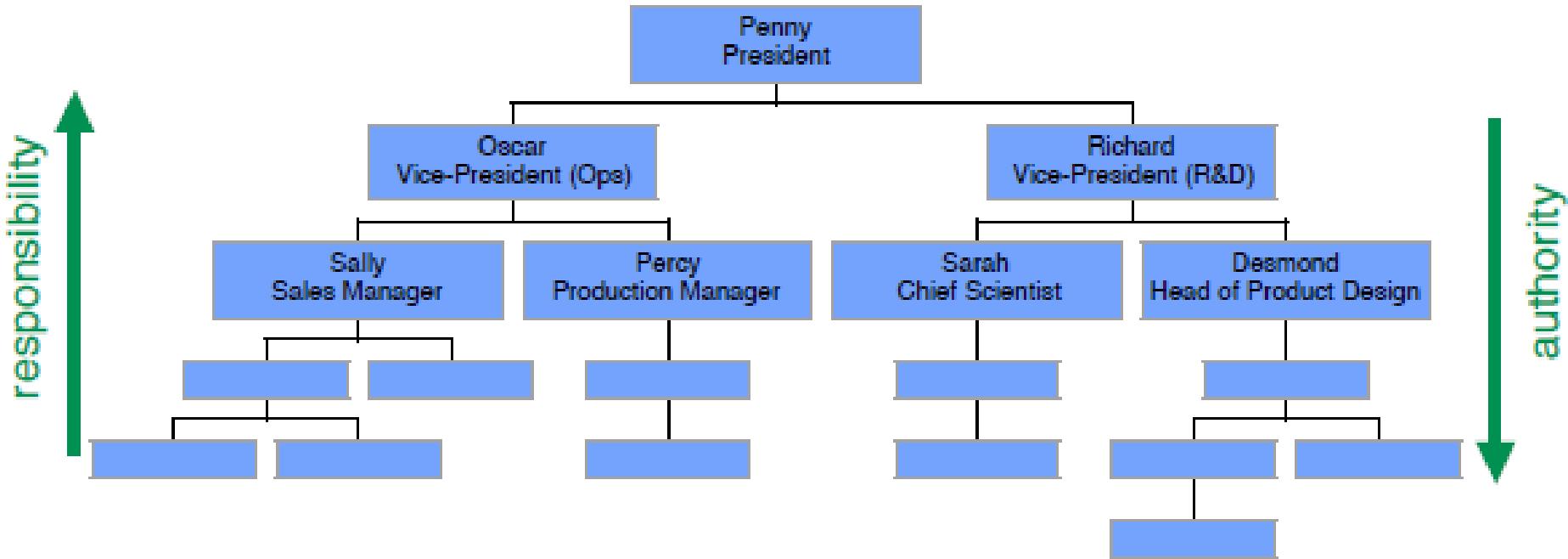
# Човешки фактори, които влияят върху изискванията



- 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

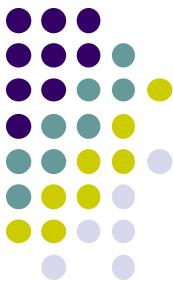


→ A useful tool for figuring out where the stakeholders are  
↳ ...but remember that most activities involve connections that cross the org chart



## 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 и обектно-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

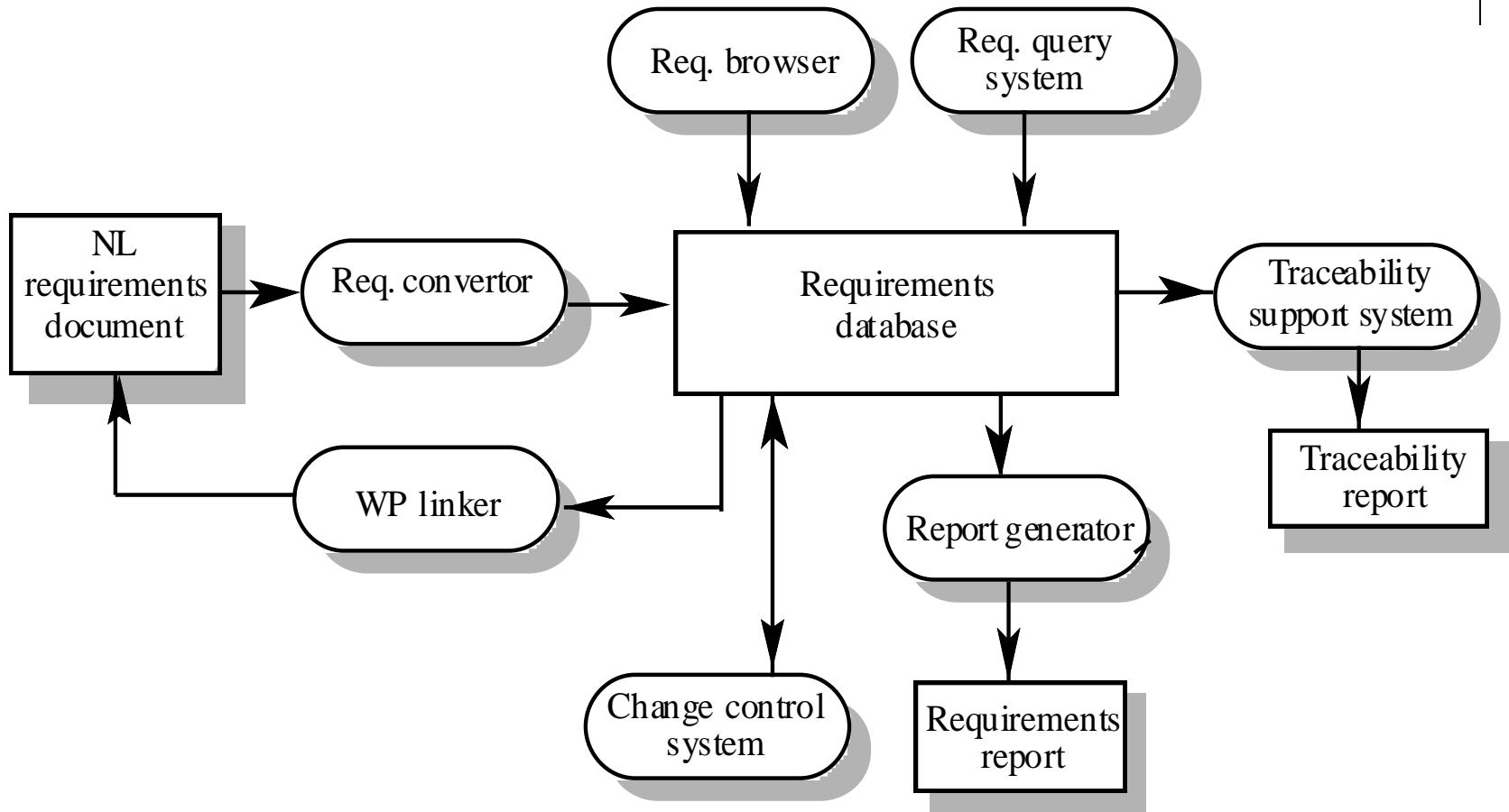
**Два типа CASE инструменти:**

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

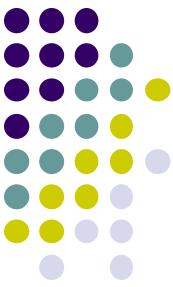


**Пример:**

## **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:

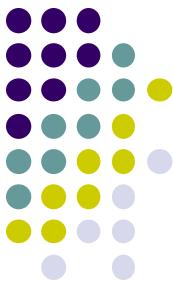
- Какви са **проблемите** с текущите процеси?
- Какво се **цели** да се подобри? **Реалистичност**
- Как трябва да се **въведат** подобренията, за да може да се постигнат тези цели?
- Как могат въведенията / подобренията да бъдат **контролирани и управлявани**?



# 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

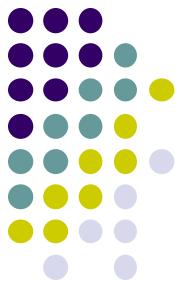
- Нивото на зрялост на компанията е важен фактор за подобрение на процесите на ИИ
- CMM (Capability Maturity Model, US Department of Defense's Software Engineering Institute, 1988-1995)
  1. **Initial level** – undisciplined process
  2. **Repeatable level** – basic cost and schedule management procedures
  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 да 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

# Въпроси:



- Защо процесът на ИИ е творчески процес?
- Коя е входната информация за процеса на ИИ?
- ИИ варира радикално от една до друга организация.  
Обяснете?
- Кои са общите/задължителни подпроцеси на ИИ?
- Какви видове модели на процеса на ИИ  
съществуват?
- Какъв е начинът за подобряване на процеса на ИИ?
  
- Според какво се класифицират процесите на ИИ при сравнението има в различните организации?

# 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, 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?

I.

## Бизнес изисквания 1

- Бизнес изискванията представлят целите на системата, записани на *високо ниво* на детайлност според вижданията на организацията или клиента, който е поръчал системата.
- Те дефинират *визията* и *обхвата* 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
  - клиентите*
  - менеджърът* на крайните потребители
  - отдела по маркетинг*
  - човек, 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 представлят **результатите - стоки или услуги** - от разработката 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.
- Същинските бизнес изисквания са *концептуални* и съществуват в бизнес среда

## 1) Същински изисквания 2

- 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.**
- Бизнес изискванията се разглеждат от гледна точка на бизнеса/потребителя и се изразяват на езика на бизнеса/потребителя

## 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.

Примерно изразяване на визията за проекта Chemical Tracking System в Contoso Pharmaceuticals. (Weigers, 2003):

За учените, които трябва да търсят контейнери с химикали, **системата** за проследяване на химикали е информационна система, която **предоставя обща точка на достъп** до складовете с химикали и до търговските компании.

**Системата ще съхранява местоположението** на всеки контейнер с химикали в рамките на компанията, **количеството материал**, който е останал в контейнерите, и **цялата история** на местоположението и използването на всеки контейнер.

**Системата ще спестява на компанията 25% от разходите** за химикали през първата година от нейната употреба, като **позволява** на компанията да **използва изцяло химикалите**, които са налични в компанията, да премахва/унищожава контейнери, които се използват частично или са остарели, и да използва един стандартен процес за закупуване на химикали.

**За разлика от сегашния ръчен процес на поръчка, системата ще генерира всички отчети, които трябва да спазват федералните и правителствените наредби**, които изискват отчитането на употребата на химикалите, тяхното съхранение и изхвърляне/унищожаване.

## II.

# Системни изисквания – детайлно описание на изискванията

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

# *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.

# *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.

**III.**

## **Софтуерна спецификация ( Software Design Specification)**

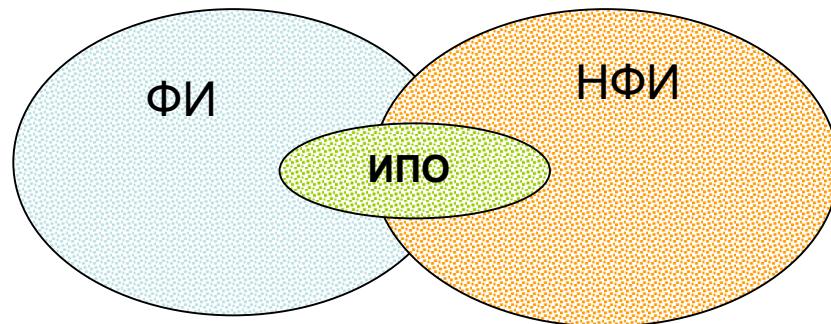
# Класификация на софтуерните изисквания – хоризонтална класификация

- 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, класифицират изискванията като:

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

## A. Функционални изисквания 1

- **Функционалните изисквания описват какви услуги или функционалности трябва да предоставя системата на потребителите 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.*

## 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**
  - **Функциональная** перспектива
  - Перспектива на **поведения**
- Принципно спецификацията на функционалните изисквания трябва да бъде едновременно **пълна** и **консистентна**. Практически това е трудно изпълнимо...
- **Проблемы** (взаимно изключващи се) при описанието:
  - Ако е твърде общо, преобладава неяснота/двусмислие;
  - Ако е твърде специфично може да затрудни проектирането, получава се обемист и трудно четим документ

# Примерни въпроси за извлечане на функционални изисквания

- **Функционалност**
  - Какво ще извършва системата?
  - Кога ще го извършва?
  - Има ли различни модели на действие/работка?
  - Какви изчисления или трансформации на данни трябва да се извършват?
  - Кои са подходящите реакции за възможните стимули?
  - ...
- **Данни**
  - Какъв трябва да бъде форматът на входните и изходните данни?
  - Трябва ли някои от данните да се пазят за някакъв период от време?
  - ...
- **Поведение**
  - Какви екрани (състояния) ще трябва да бъдат преминати от системата, за да получи изходните резултати?
  - ...

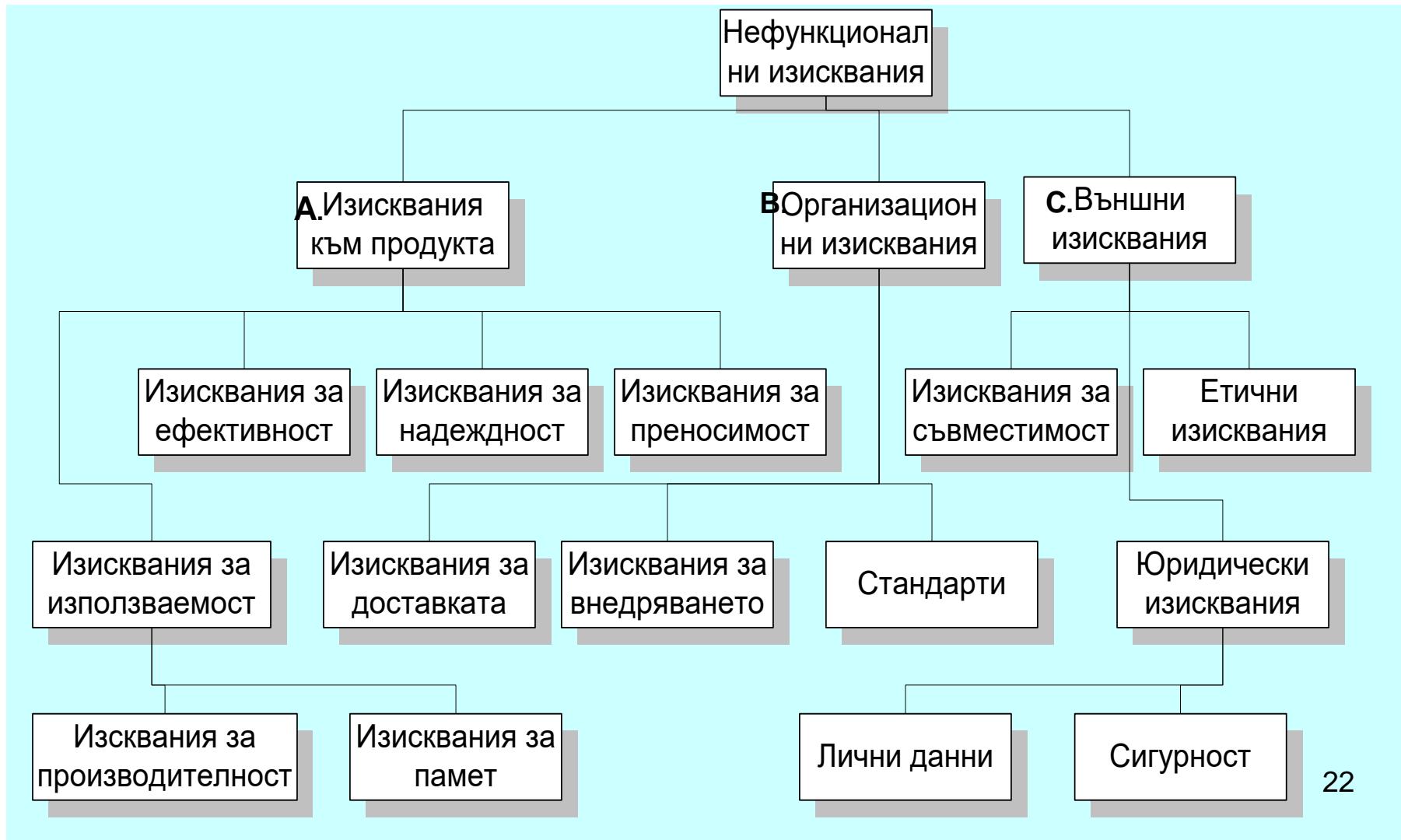
## **В) Нефункционални изисквания: Дефиниция**

- **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

- Системната услуга X ще бъде с наличност 99%
- Системата Y ще обработва минимум 8 транзакции в секунда
- Изпълнимият код на система Z ще бъде ограничен до 512 килобайта
- Потребителският PIN ще се изисква в рамките на 5 секунди, след като картата е била поставена в четящото устройство за карти
- Системата ще бъде разработена на PC и Macintosh платформи
- Системата трябва да криптира всички външни комуникации чрез алгоритъма RSA
- Системата X ще бъде реализирана с използване на версия 5 на библиотеката Y

# 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 работата като цяло на системата. Също и 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 пълно 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 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*.

# 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 изходните данни.

## ■ 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.

- *Зависимост/противоречие между различните качествени характеристики.*
- *Зависимост между функционални и нефункционални характеристики.*
- *Илюстрация:*

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.

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

- R: The system shall ensure that data is protected from unauthorised access.

In slightly more detail as follows:

- R.v1: 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 ( <i>не са единствени</i> )
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 (обем)	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? Как, от какво?

# 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 - примери**

## **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 - примери

### ■ Време за доставка/Разходи

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

## С) НФИ: Външни изисквания

- 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 система 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, налагани на разработваната система, 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 *електрозахранването, отоплението или климатизацията*?

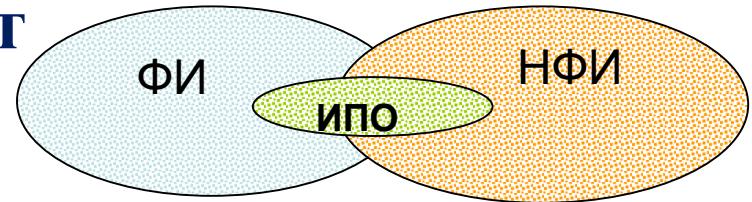
## (Въпроси за) ограничения върху дизайна – 2

- **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 (*напр.*: време за отговор) 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 (функционално 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:

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

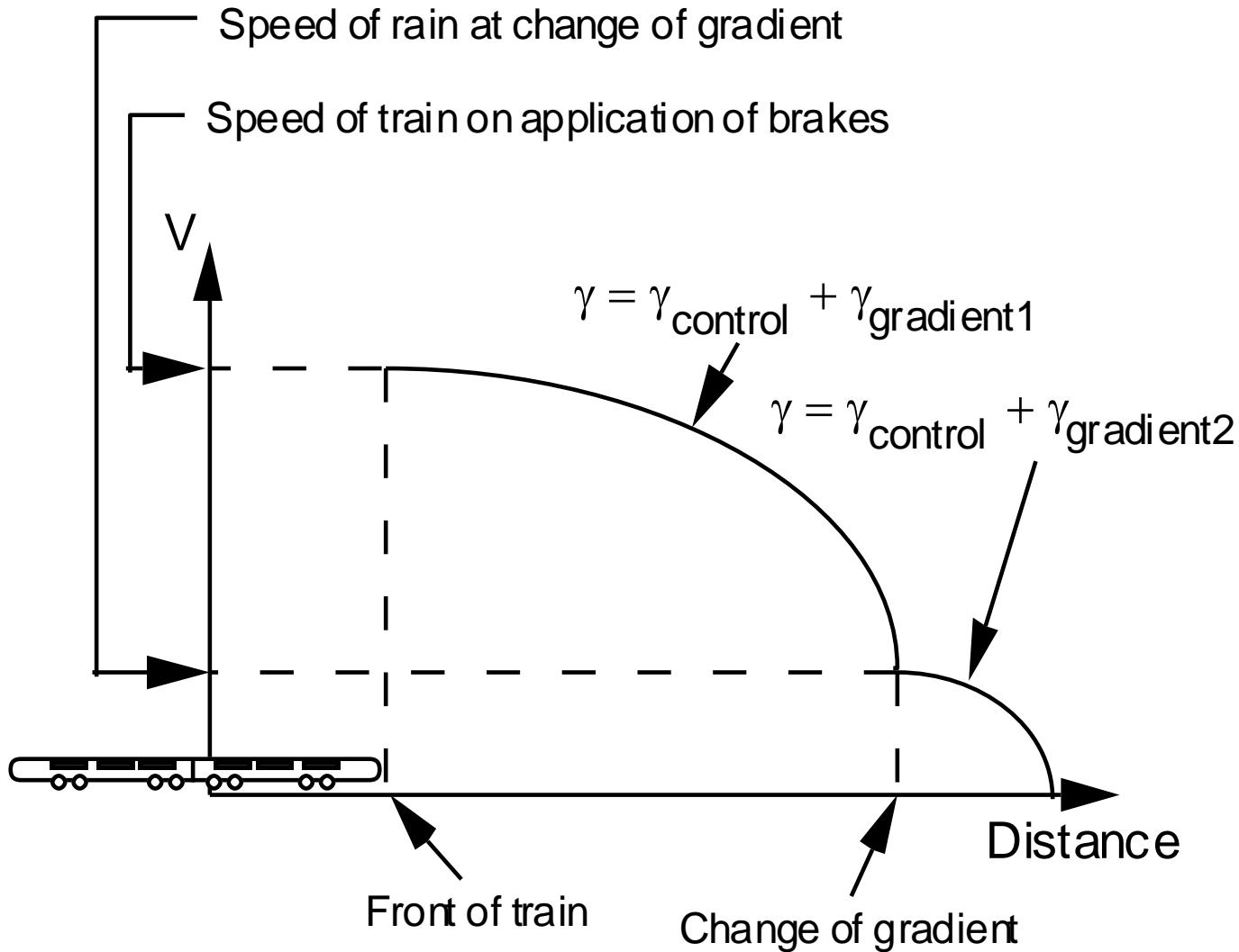
where:

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

where  $\alpha$  are known for **different types of train.**

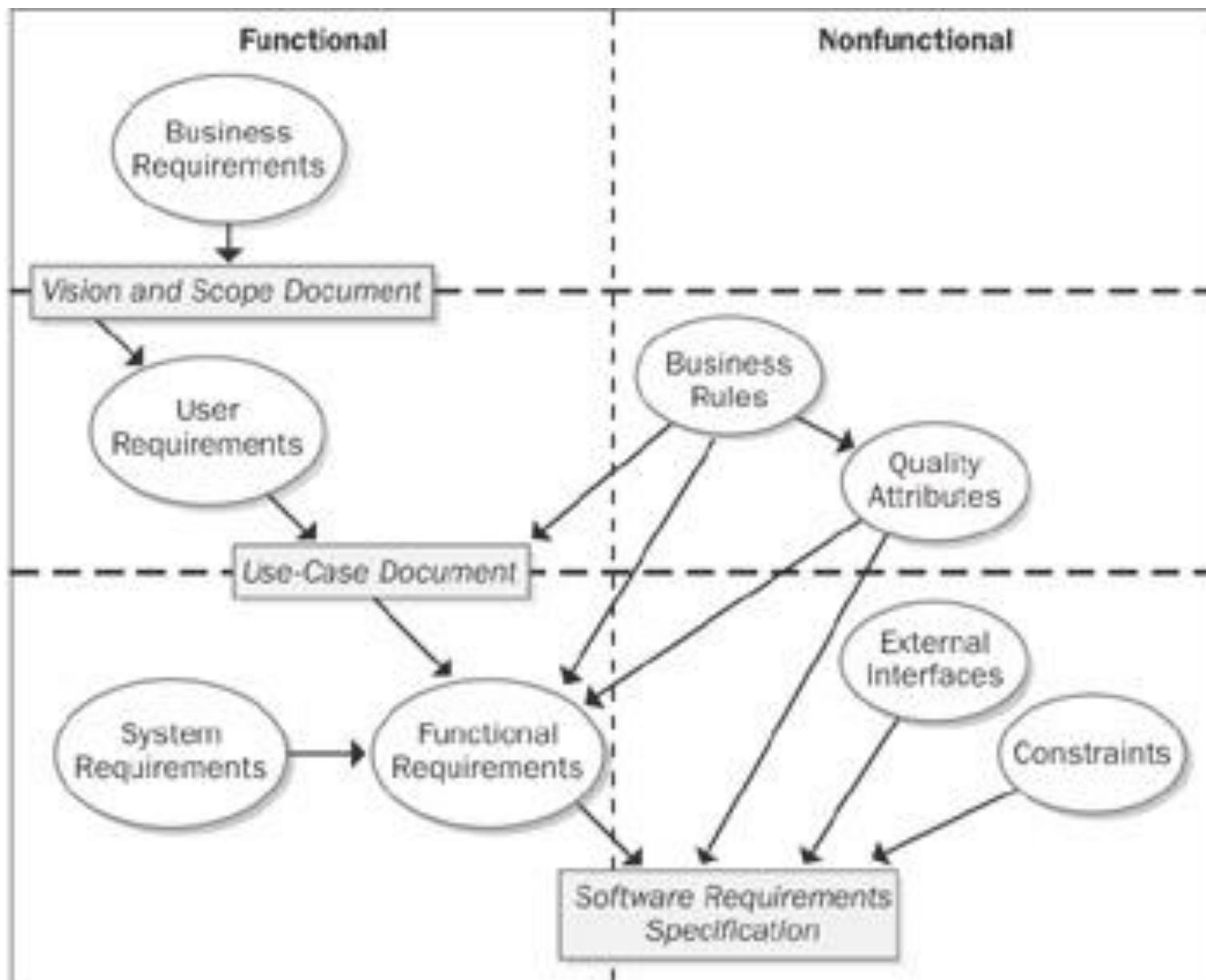
$g_{\text{control}}$  - this value being parameterised in order to remain adjustable.

## Външни изисквания: примери (продължение)



# Нива на изискванията: Връзки между информацията за различните видове изисквания

Karl E. Wiegers, 2003



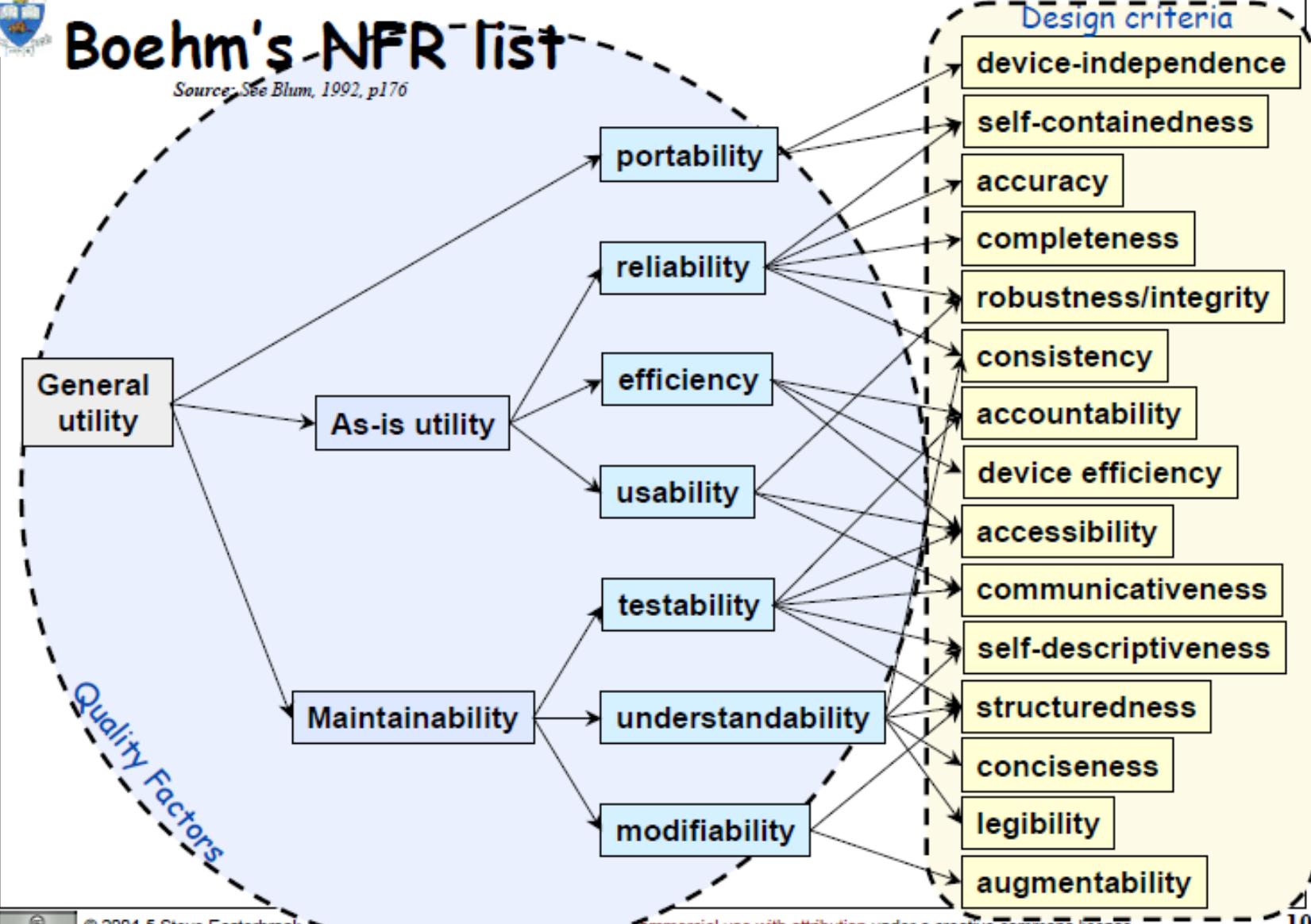
# Класификация на нефункционални изисквания според различни автори

- 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)



# Boehm's NFR list

Source: See Blum, 1992, p176





# McCall's NFR list

Source: See van Vliet 2000, pp111-3

**Product operation****usability****integrity****efficiency****correctness****reliability****maintainability****testability****flexibility****reusability****portability****interoperability**

operability

training

communicativeness

I/O volume

I/O rate

Access control

Access audit

Storage efficiency

execution efficiency

traceability

completeness

accuracy

error tolerance

consistency

simplicity

conciseness

instrumentation

expandability

generality

Self-descriptiveness

modularity

machine independence

s/w system independence

comms. commonality

data commonality

**Product revision****Product transition***Quality Factors***Design criteria**

# 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 и сигурността
    - 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

# Въпроси

1. Какви видове изисквания има според:

- Нивото на детайлност?
- Характера на изискванията?

2. Какво представляват ФИ и НФИ

3. Как се групират НФИ?

4. Какви са качествените характеристики на софтуера?  
Как се групират?

5. Кои са трудностите за описание на НФИ?

# **Инженеринг на изискванията**

---

## **Извличане на изискванията**

**Лекция 4**

# Съдържание

---

- **Извличане на изискванията- същност, необходима информация**
- **Цели и процес на извлечане на изискванията**
- **Начална информация за идентифициране на изискванията**
- **Техники за извлечане на изискванията**
- **Проблеми при идентифициране на изискванията**

# **Кога възниква необходимост от ИИ**

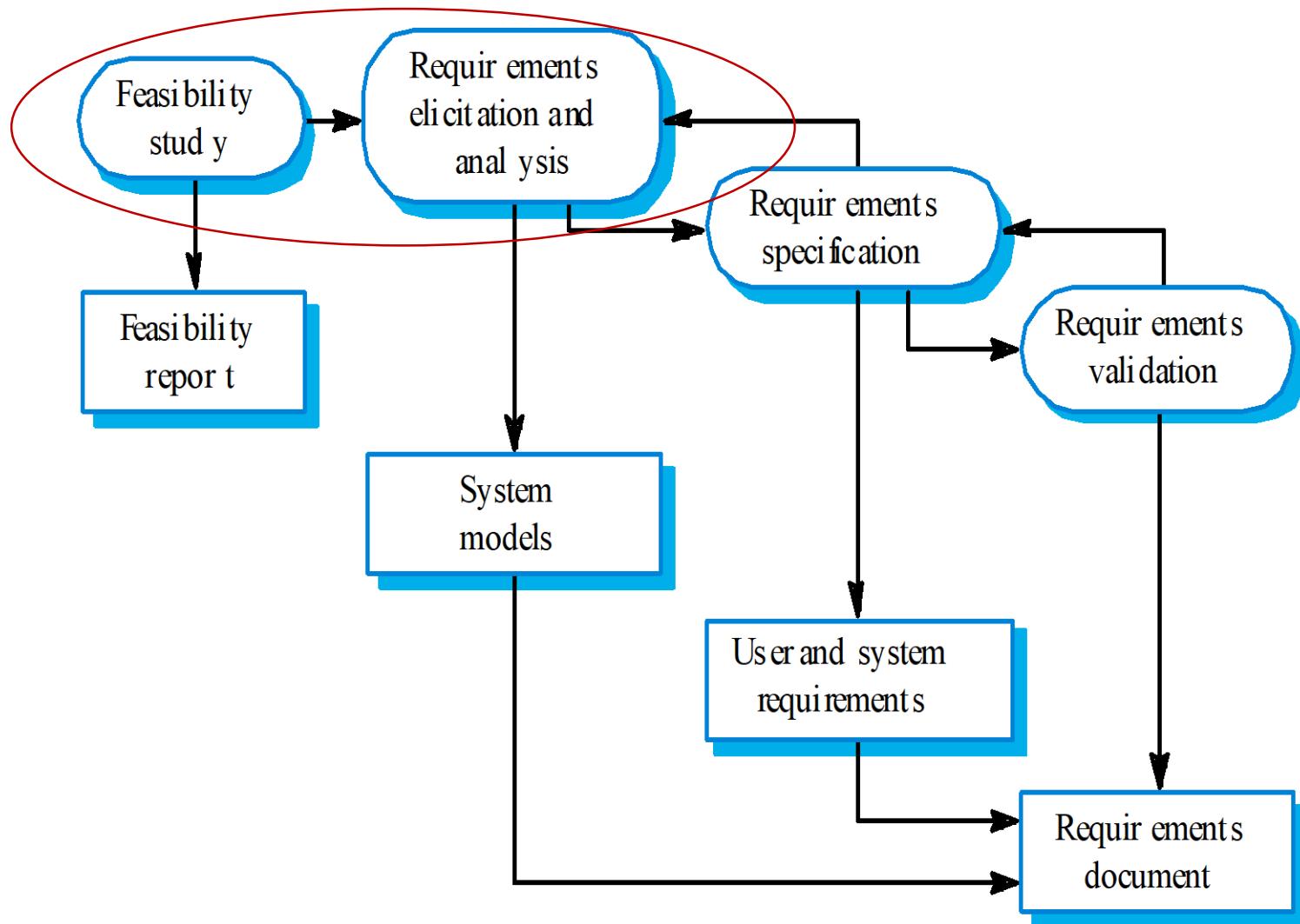
---

Начало е:

**Съществува “проблем”, който трябва да се реши,  
зашото съществува:**

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

# Процес на инженеринг на изискванията



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

---

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

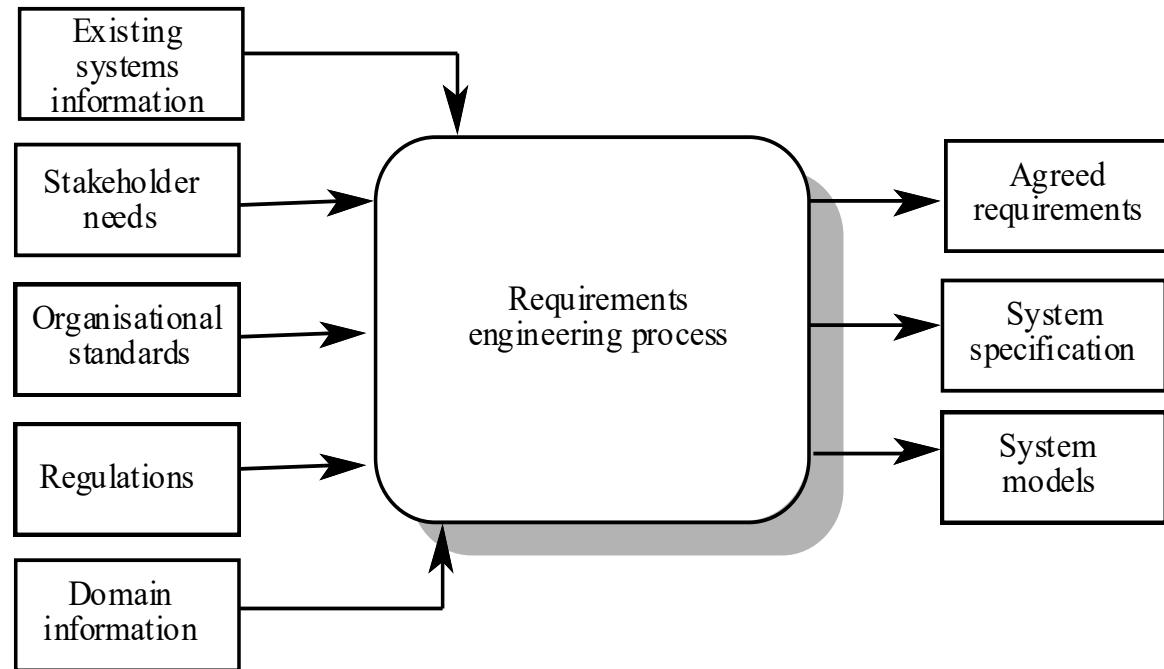
**Изходна информация:** Доклад-становище дали системата да бъде разработана:

- *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.

# Извличане на изискванията

## Използва се информация от:

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



# Извличане на изискванията – процес и общи принципи

## (Requirements elicitation)

**Дефиниция: Извличане на изисквания е процесът за търсене, откриване (идентифициране), обобщение на софтуерните изисквания от наличните източници.**

- *Включва работа за да се:*
  - проучи приложната област
  - анализират внимателноисканията 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 приложната област; Източници на информация

- Разбиране на проблема**

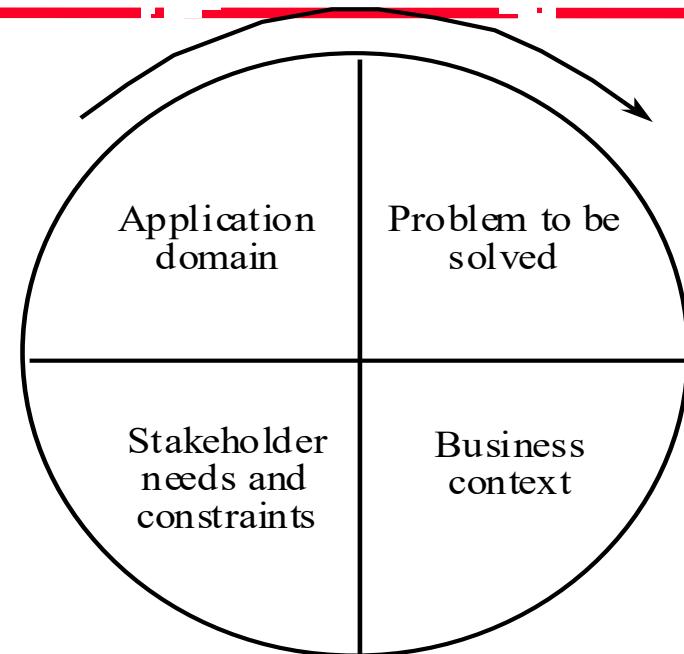
детайлите за специфичния клиентски проблем, предоставени от различни хора

- Разбиране на бизнеса**

Трябва да разбере, как си взаимодействват 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.



# Идентифициране на изискванията – начална информация

## Събиране на достатъчно информация за дефиниране на “проблема”

Кой проблем трябва да се разреши? (определяне на границите на проблема)

Къде е проблемът? (изясняване на контекста/проблемната област)

На кого е проблемът? (идентифициране на заинтересованите страни)

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 разработката)

Какво може да попречи на разрешаването? (определение на осъществимостта и риска)

The  
journalist's  
technique:

What?

Where?

Who?

Why?

When?

How?

(Which?)

## 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.

# **Идентифициране на изискванията**

---

**Идентифициране на изискванията.**  
**Изходна информация**

# Начална информация за идентифициране на изискванията

---

## Граници

- Определяне на обхвата на проблема

## Зaintересовани страни/лица

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

## Цели

- Определяне на критериите за успех

# 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.
  - Средата не може да се променя в рамките на този проект.

## Контекст на системата - 1

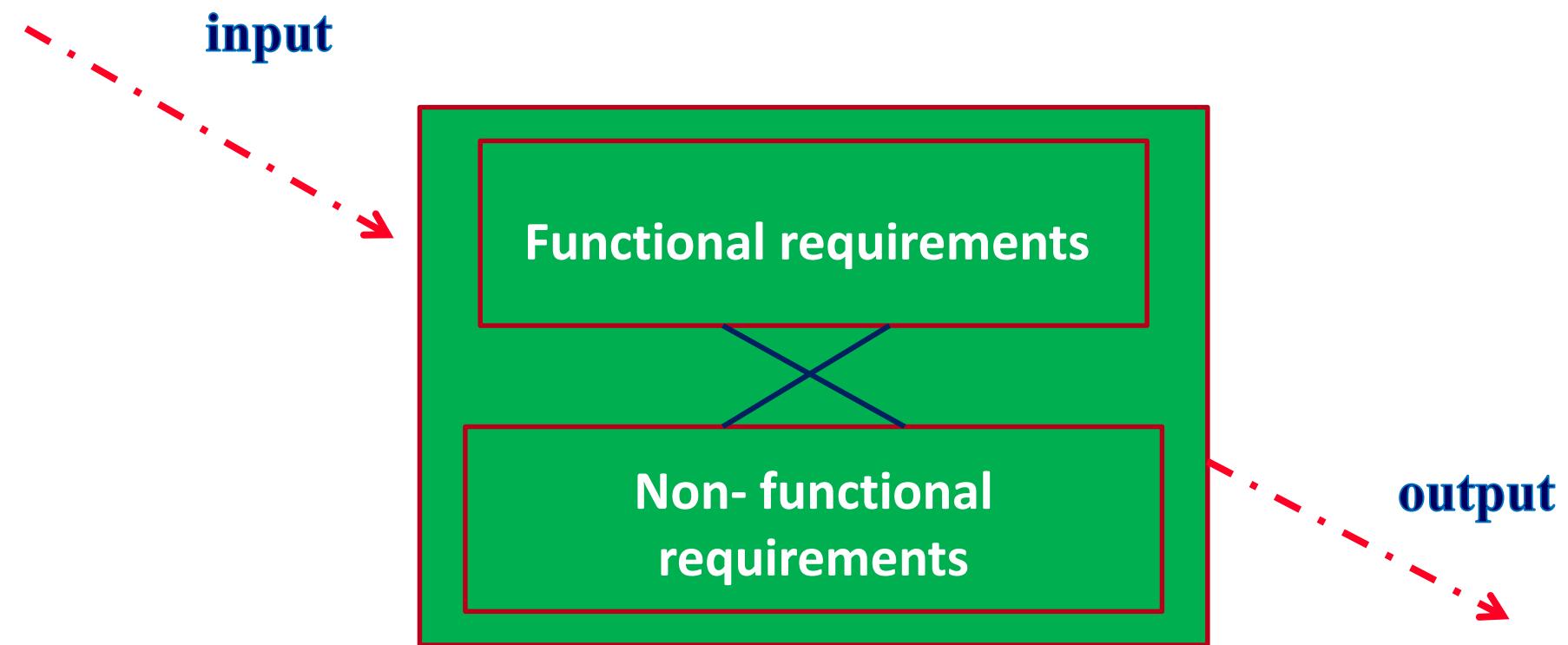
---

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

# Контекст на системата - 2

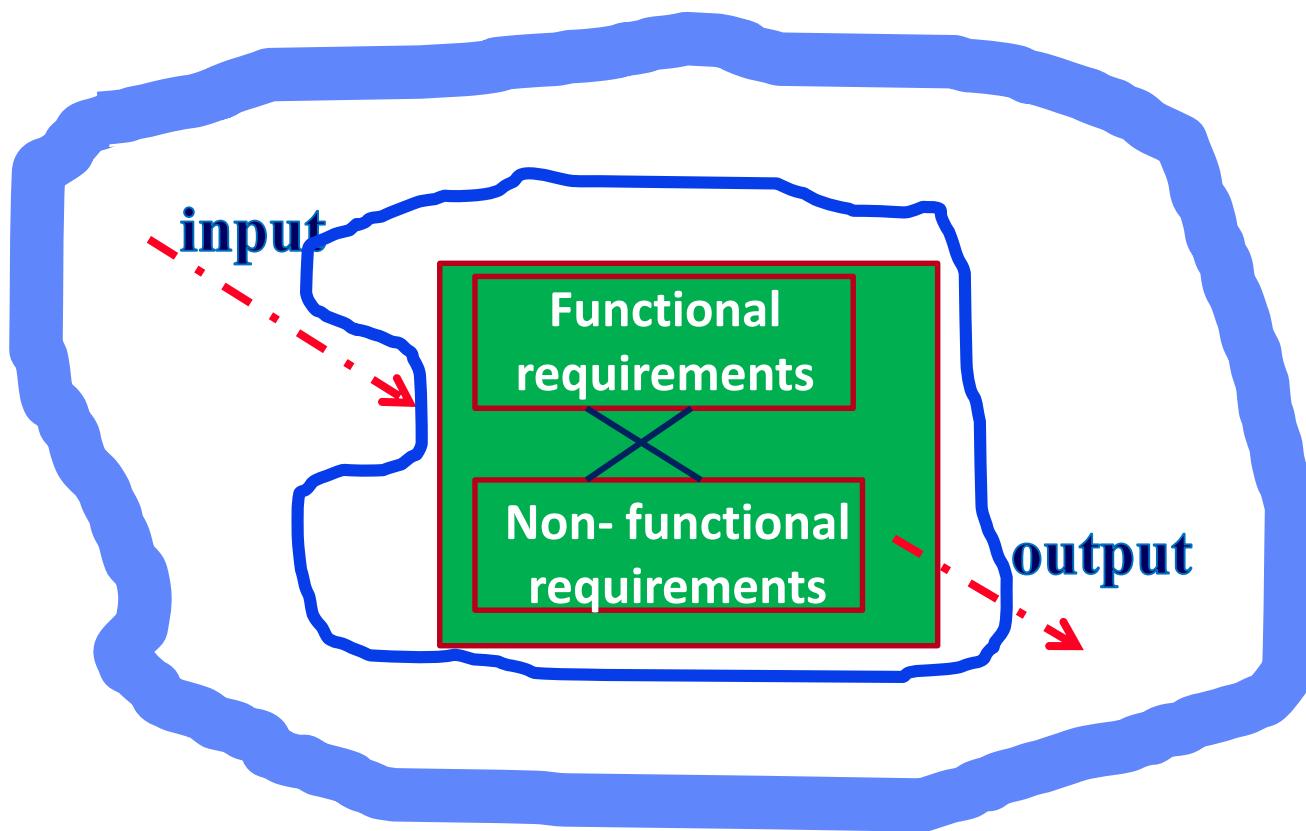
---

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



# Определяне на границите

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



Boundary zones are fact of life.

## **Инженерингът на изискванията се ръководи от контекста на системата чрез съществуващите в него източници 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*

# Диаграма на контекста

---

Диаграмата на контекста се състои от:

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

# 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:* Понякога не знаете какво не знаете – затова използвайте *списък и/или диаграма.*

# **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

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

# Влияние на заинтересованите лица

---

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

# Заинтересовани страни

## ■ Анализ на заинтересованите лица:

Да се определят всички хора, с които ще се дават консултации по време за исканата информацията.

## ■ 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**

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!

# Техники за определяне на заинтересованите лица

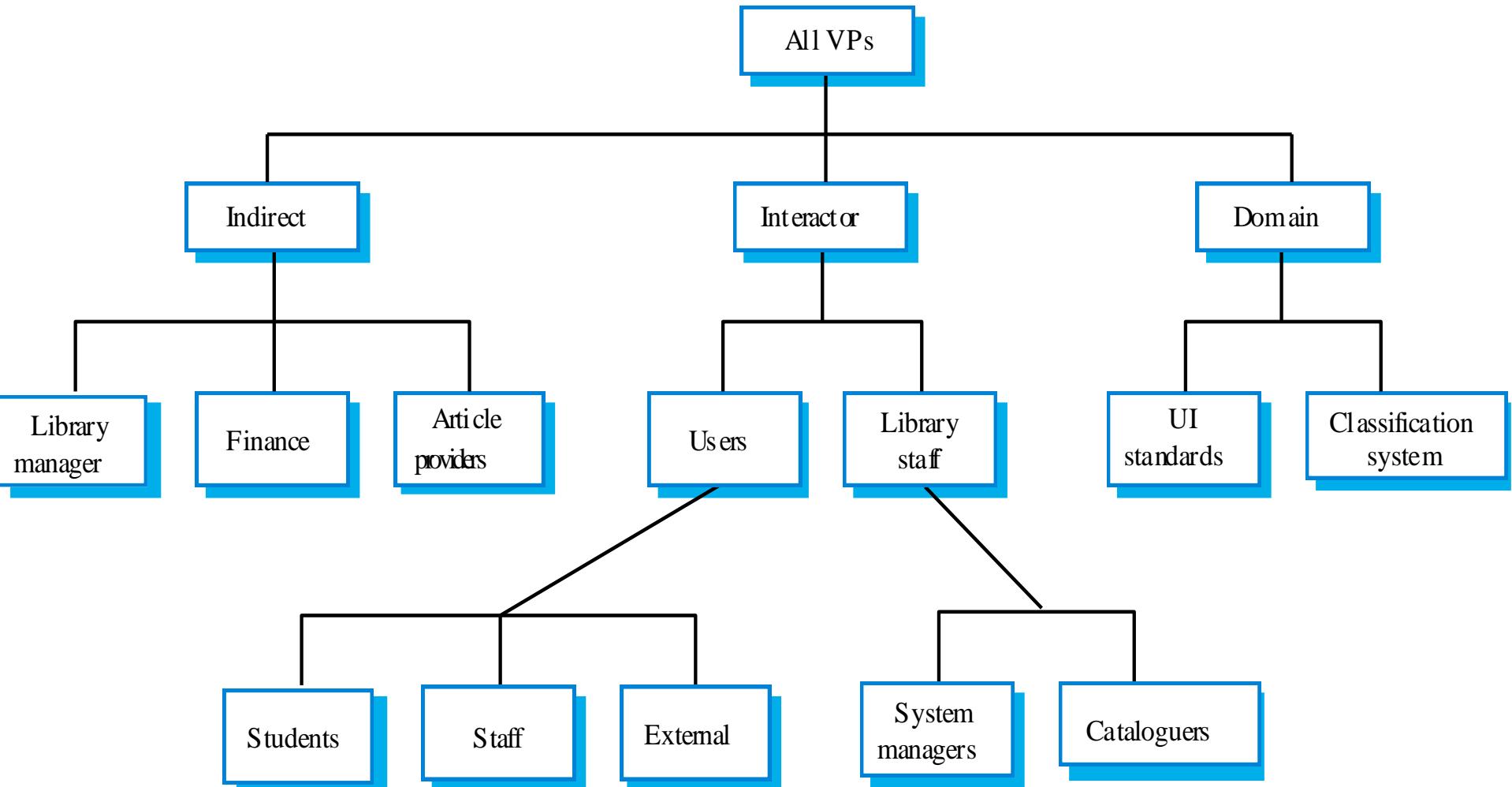
- *списъци*
  - въпроси: **кой** знае, работи, чувства, получава полза или вреда ...
- *класификация* на заинтересованите страни и идентифициране на конкретен човек за отделните роли
- **мозъчна атака (brainstorm)** срещи за откриването на идеи, заинтересовани лица
- **mind-maps** за формиране и откриване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



[Yu 1997]  
[van Lamsweerde 2001]

Glinz and Wieringa 2007]

# Пример: LIBSYS library system



# Проблеми на идентифицирането на изискванията - обобщение

- 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*

# Цели

**Крайната цел е намерение относно свойствата и/или употребата на системата**

- Дефинира се различно ниво на целите:
  - Високо ниво – крайна цел
  - Подцели – декомпозиция на крайната цел  
(*refinement-усъвършенстване*)
- **Подцелите дефинират съответни сценарии**

*Пример:*

**Цел:** Да се разработи система за управление на банкова сметка.

**Подцели:** Внасяне, теглене, депозиране, прехвърляне ...

## Видове техники за извлечение на изискванията

---

- Питащи (задаване на въпроси)
- Колаборативни
- Изграждане и разиграване (на прототип, сценарий, story др.)
- Наблюдение

# Техники за извлечение на изискванията

---

1. Сценарии
2. Интервюта, въпросници, анкети
3. Мозъчна атака
4. Семинари (срещи) за изискванията; FAST
5. Soft systems methods
6. Наблюдения и социален анализ
7. Прототипиране
8. Software Story

*Други:* Повторно използване на изискванията

---

# Сценарии и потребителски случаи

# 1. Сценарии - 1

---

- Сценариите са примери за реалното използване на системата.
- Точно определена последователност на взаимодействие между актор и система
- 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.

---

*Пример:*

Прием в болница – какво и как се случва по време на приемането?

*Типичен отговор:* “Вие, или човекът, който ви придружава, ще говори с лицето на рецепцията. Трябва да покажете своята здравна карта и да обясните кой ви е насочил към болницата. След това да опишете състоянието си. Да бъдете прегледан от лекар и ...”

# Сценарии - 2

---

## Сценарий:

Обикновено са кратки ( 3 и 7 стъпки)

Описват: *положителни* (необходимо 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

Кратките сценарии са много добри за бързо илюстриране на специфични взаимодействия

## Недостатъци

Липса на структура; Трудно се проверяват за пълнота

# 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 изиска 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 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 - структуриран метод за идентифициране и анализ на изискванията

---

- Потребителските случаи са техника на структурно представяне, базирана на сценарии, която идентифицира *актьорите* в едно взаимодействие и която *описва самото взаимодействие*.
- Наборът от всички потребителски случаи за една система - описва всички възможни взаимодействия със системата.
- Диаграмата на потребителския случай се допълва с:
  - Диаграма/и на последователността добавя детайли към потребителските случаи като показва 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.

Klaus Pohl, 2010

# Диграма на потребителските случаи

---

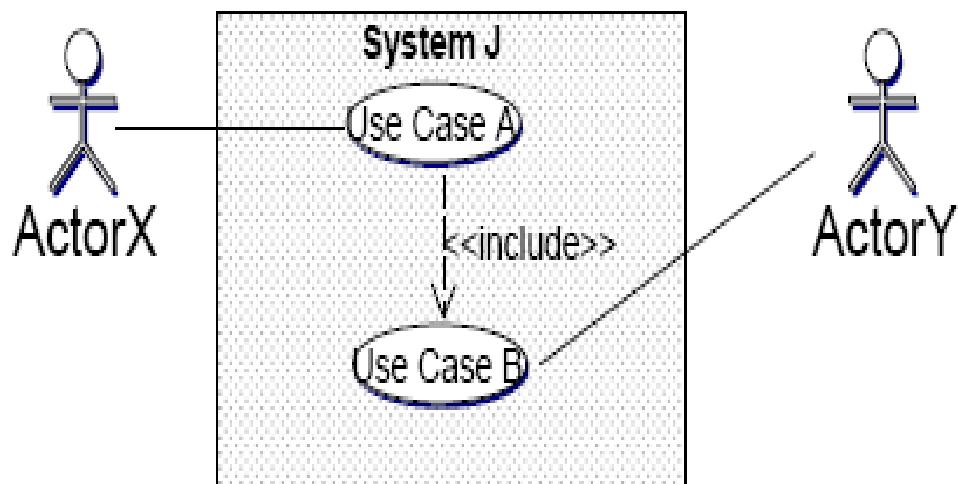
Диаграмата на потребителските случаи за една система представя:

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

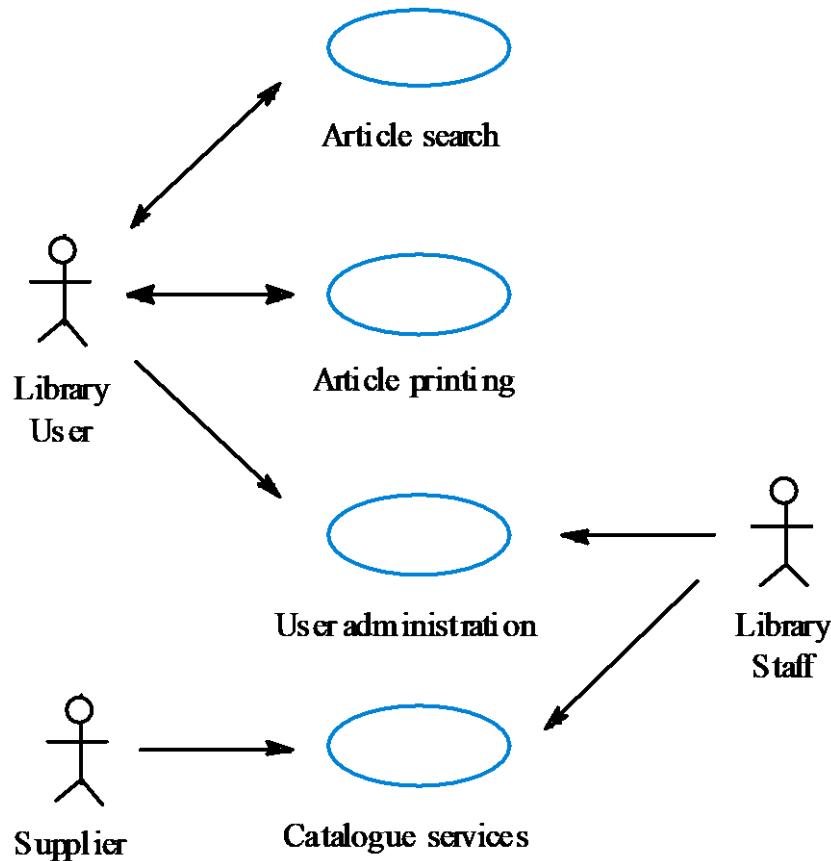
# Потребителски случаи в UML

- UML предоставя графично представяне на потребителските случаи, наречено *Диаграма на потребителските случаи (Use Case diagram)*.
- То позволява графично изобразяване на:
  1. Актьори
  2. Функции
  3. Връзки
  4. Зависимости: include, extend, генерализации (обобщения)

Дефинира границата на системата

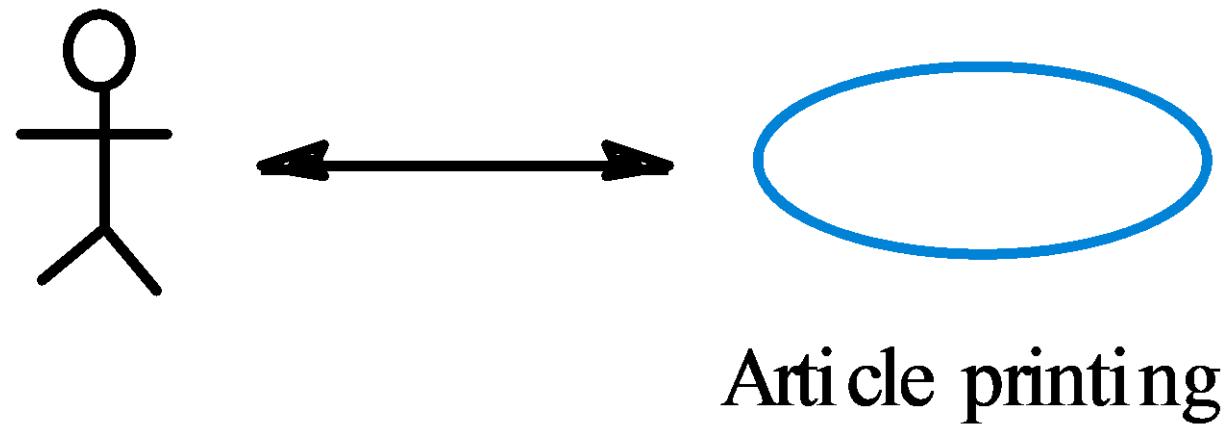


# Пример: Потребителски случаи в LIBSYS

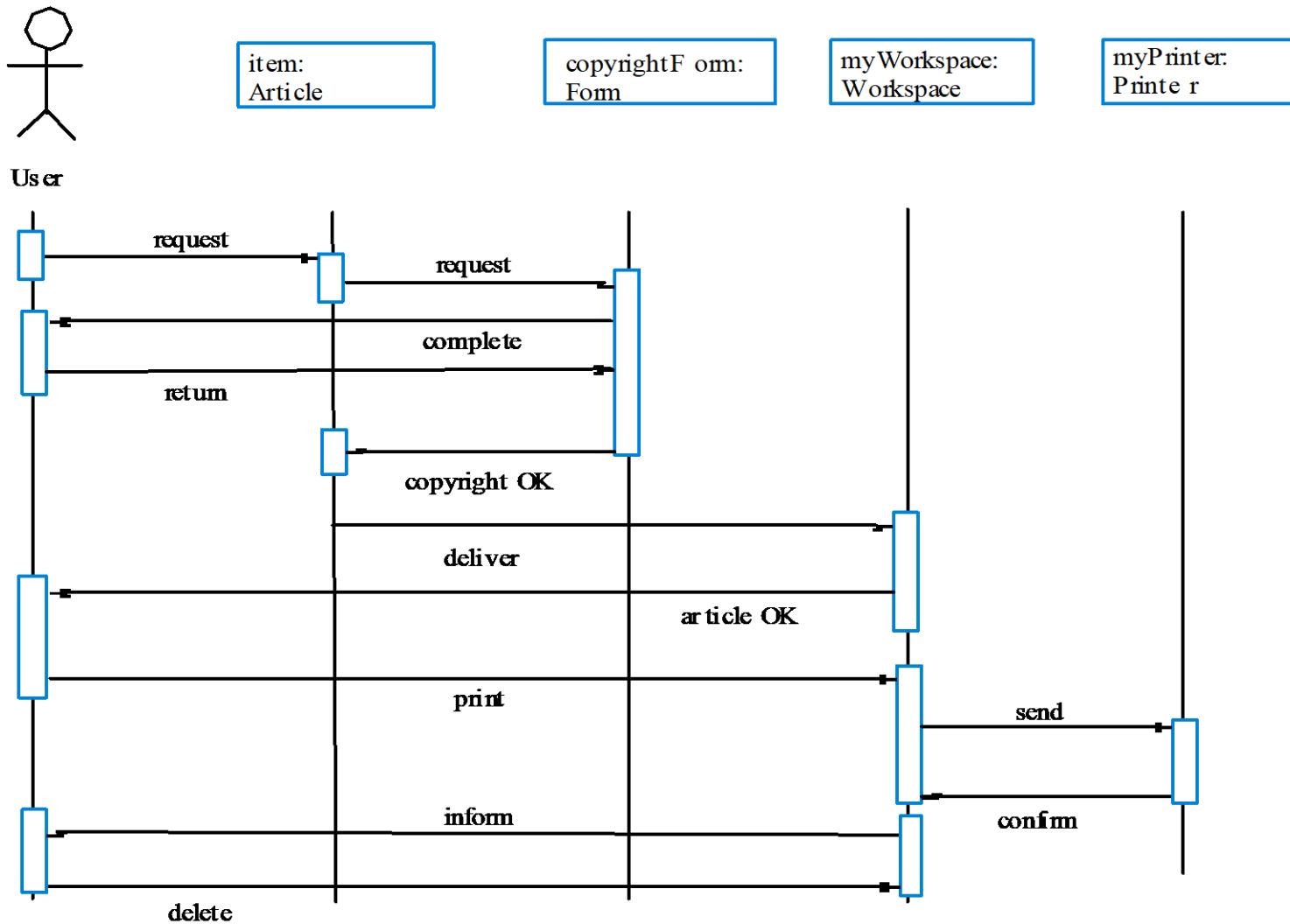


## Пример: Потребителски случай за принтиране на статья

---



# Пример: Диаграма на последователност за случай “принтиране на статия”



## 2. Интервюта -1

---

- Интервюирането е основен източник на информация за изискванията
- Интервюирането изисква:
  - търпение,
  - изслушване,
  - разбиране и
  - задаване на разнопосочни въпроси.

# Интервюта - 2

---

Видове интервюта:

- **Затворени интервюта** - търсят се отговори на предварително подгответи въпроси
- **Отворени интервюта.** Няма предварително подгответи въпроси. 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 изисква.
- На ЗЛ *трябва да се даде стартова точка* за дискутиране.
- Трябва да се има предвид организационната политика – някои от реалните изисквания може *да не се обсъждат с конкретно ЗЛ* заради определени (политически в общия смисъл) ограничения.

# **Характеристики на интервюирането**

---

**Интервюто се фокусира върху въпроси за:**

Бизнес случай за проекта

Функционални изисквания за проекта

Нефункционални изисквания

Рискове

Ограничения

Зainteresовани лица

**Какво не може да се открие по време на интервю?**

- изясняване на областта на системата
- знание за организацията

**Рядко са напълно достатъчни за инженеринга на изискванията**

### 3. Мозъчна атака (Brainstorming)

---

#### Метод за генериране на идеи в група

**Brainstorming е техника за работа в екип, която предизвиква откриването на знания от всеки участник - надграждането 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
- Включва не само дискусия, но и сесия за **съвместно управление на знания.**
- Едно от най-добрите средства да се определи дали получените изисквания са правилни или не.
- Продължителност и участници

## 4. Работни срещи (Requirements workshops)

---

Аналитикът **събира** (основни) заинтересовани страни, за да анализира системата и да разработи решението. **Изискванията се идентифицират и дискутират съвместно.**

В идеалния случай се провеждат в **контролирана среда** чрез обмен на знания.

Ръководителят на срещата води процеса/дискусията **фокусирано**, а **протоколчик** документира дискусията.

Често са необходими **множество срещи**, за да приключи процесът успешно.

*Недостатъци:* Може да се окаже трудно да се съберат всички нужни заинтересовани страни едновременно.

## /Facilitated Application Specification Technique (FAST) – вид работна среща

Има за цел да покрие празнината между мисленето на разработчиците и желанието на клиентите.

Всички заинтересовани страни се считат за част от екип, за който се организира следното:

- 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 чрез набор от изисквания.*

*Каква е разликата между работните срещи 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 е техника 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

*Пример:* 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 (*Пример: Социална мрежа*)
  - б) организационни изисквания (*Пример: Обслужване на клиент* )

# Подход и характеристики на етнографското изследване

---

Етнографските проучвания са ефективни, когато *работата е по-обширна, сложна и динамична и е невъзможно представяне с опростени системни модели.*

- ❖ Хората не трябва да обясняват или да говорят за своята работа, а да бъдат наблюдавани.
  
- ❖ Ефективността на етнографските проучвания зависят от:
  - личния характер на етнографа,
  - типа на процеса и
  - на участващите ЗЛ.

# Насоки за провеждане на 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 на описание – механизъм за организиране и структуриране на етнографския архив

---

## ➤ **Обстановката на работата**

Описва се *контекстът и физическото местоположение* на работата и как хората използват обектите, за да изпълняват задачите си.

*Пример:* проучване за описание на *helpdesk*.

## ➤ **Социална и организационна перспектива**

Разкрива *какво се случва ежедневно* в работата така, както го виждат *различните участници*. Тази гледна точка се опитва да *обедини* (*различието*) на възприятията на отделните участници.

## ➤ **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 (*Пример: Design patterns*).

## Възможности за повторно използване

---

- Когато изискването се отнася до *предоставяне на информация за приложната област (ограничения на системата)*.
- Когато изискването се отнася до *стила* на представянето на информацията (*Пример: 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)
  - User story
  - Прототипиране
- Анализ на софтуерните изисквания
- SMARTT изисквания

## 8. Потребителска история (User story)

- **Кратки, прости, общи описание на функция, разказана от конкретна гледна точка.**
- Гледната точка е на човек - потребител, клиент или друга заинтересована страна, който желает функцията.
- Една потребителска история трябва да уточнява крайната цел (ползата, стойността), която заинтересованата страна ще получи.
- Артефакт, който определя работата на разработването на софтуерната система.

# Цели на потребителските истории

- Способства за деформализиране на описанието на изискванията.
- **Променя фокуса на работата от описание на изискванията към дискусия за тях.**
- Потребителската история запазва фокуса върху потребителя и напомня, че има действителна цел, която трябва да бъде постигната чрез прилагане на функционалност.

## 8. Прототип

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

- ✓ Хардуерни системи
- ✓ Софтуерни системи

# 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**

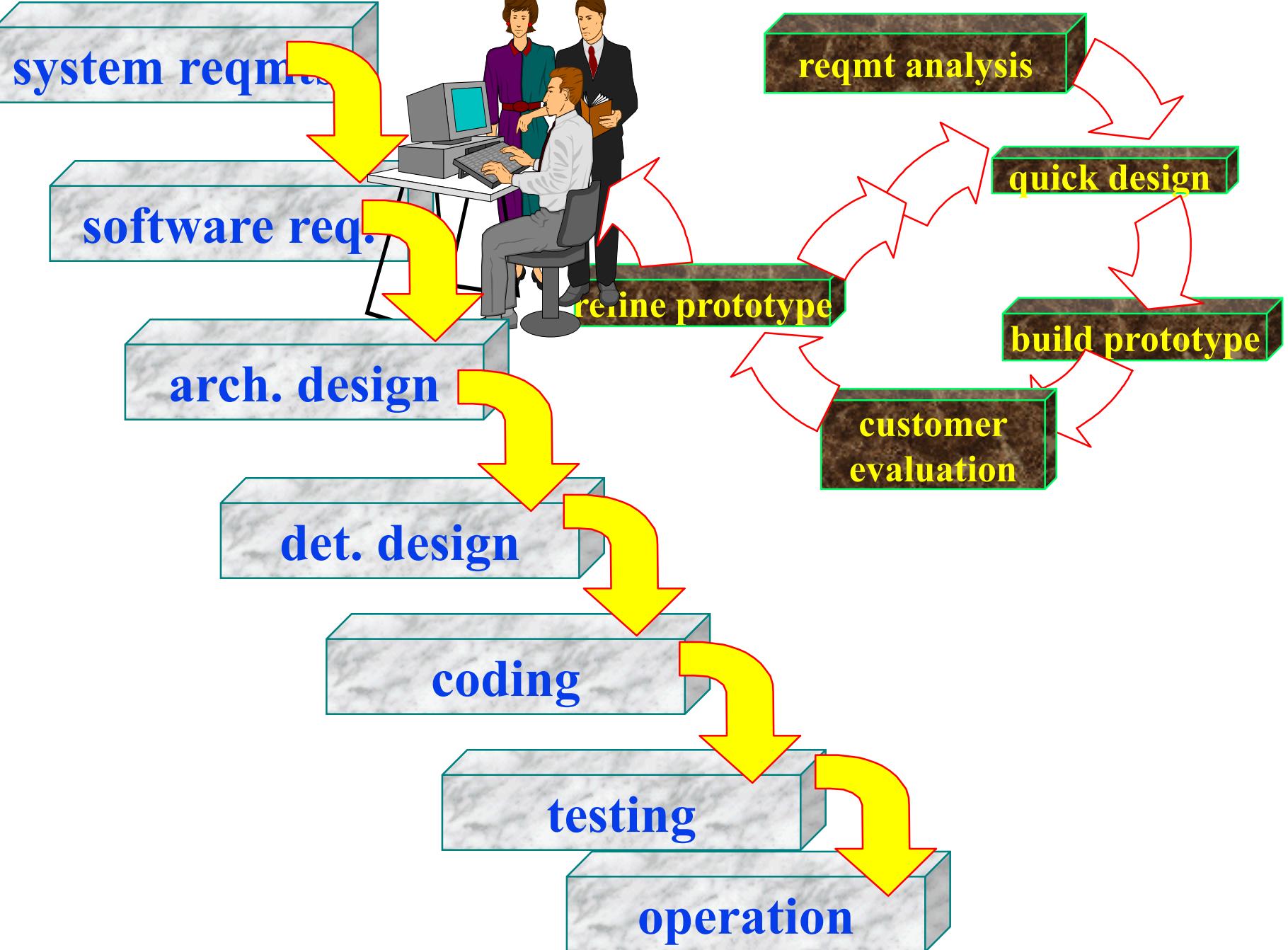
1. Прототипите са полезни за *идентификацията на изискванията*, тъй като потребителите могат да експериментират със системата и да посочат нейните силни и слаби страни. Те имат *нещо конкретно*, което да критикуват.
2. Чрез прототип при *валидиране на изискванията* лесно може да се коригира/оптимизира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

# 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 *определенi функционалности*
  - *не се реализират* някои *качествени изисквания* (RT, memory, error handling...)
  - *може да не се реализират* някои *основни процеси* на „*класическата*“ разработка (управление, тестване ...);



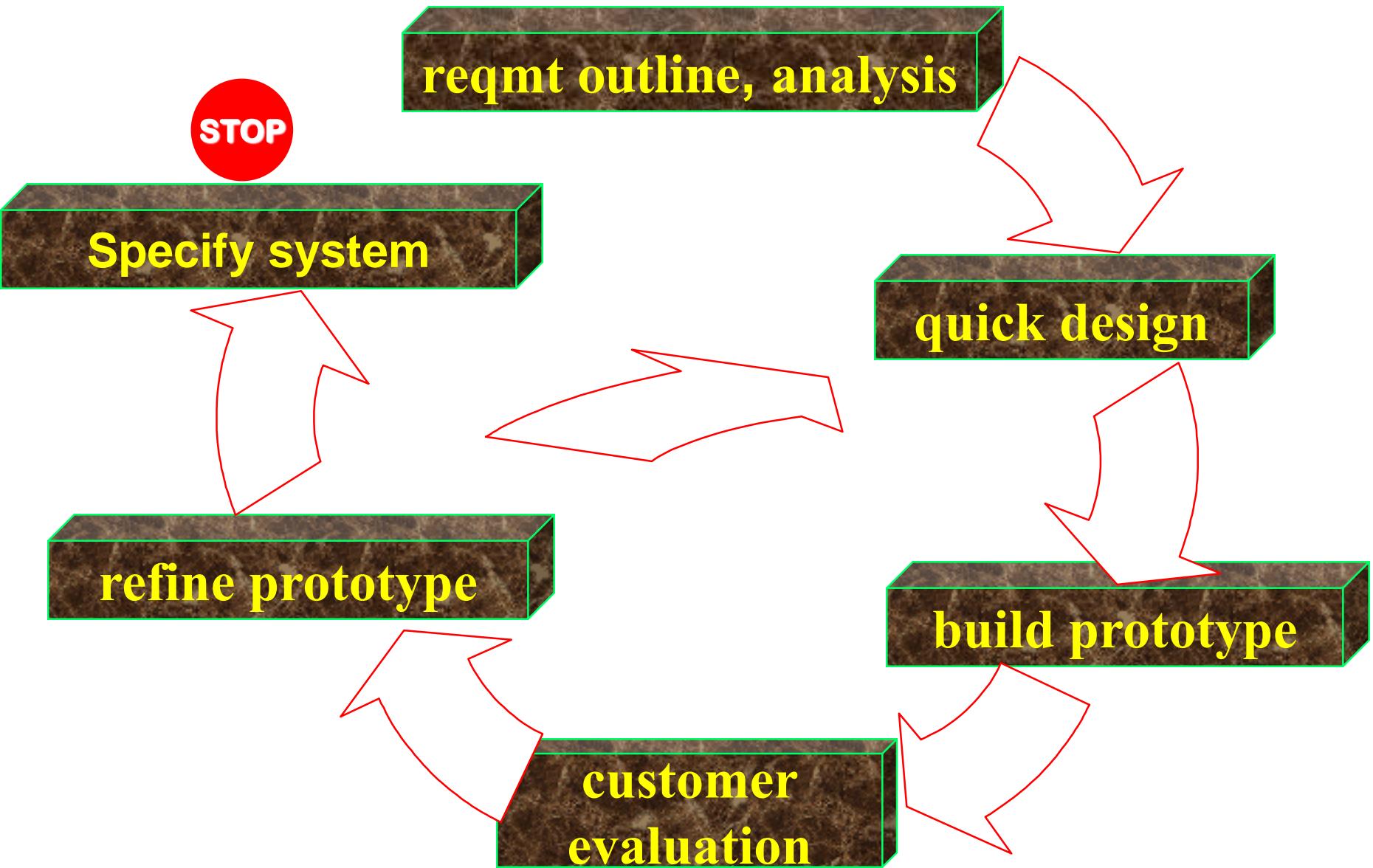
# Видове прототипи – 1

- **Throw-away прототип**
  - подпомага специфицирането на изискванията
  - 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 изисква прецизност в изработката.
  - основна техника в съвременните софт. технологии- **зашо?**
  - за малки и средни по големина системи, но не и за големи проекти  
**(зашо?)**

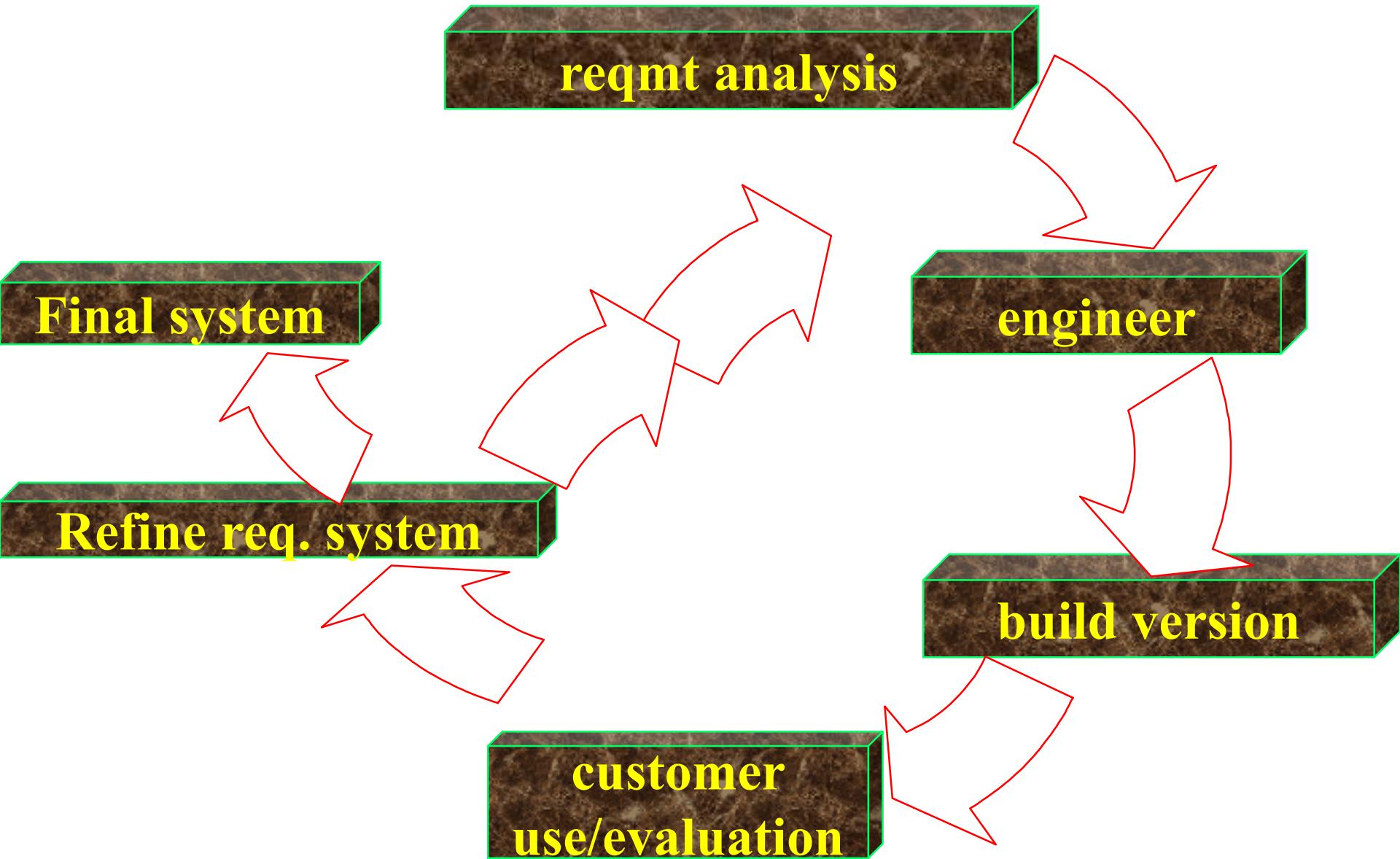
*Приложение* - e-commerce приложения, web-site разработване

*Boundaries ...?*

# Модел на Throw-away прототипиране



# Модел на еволюционното прототипиране



## **Видове прототипи и видове изисквания, които те представят - 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 представят 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.

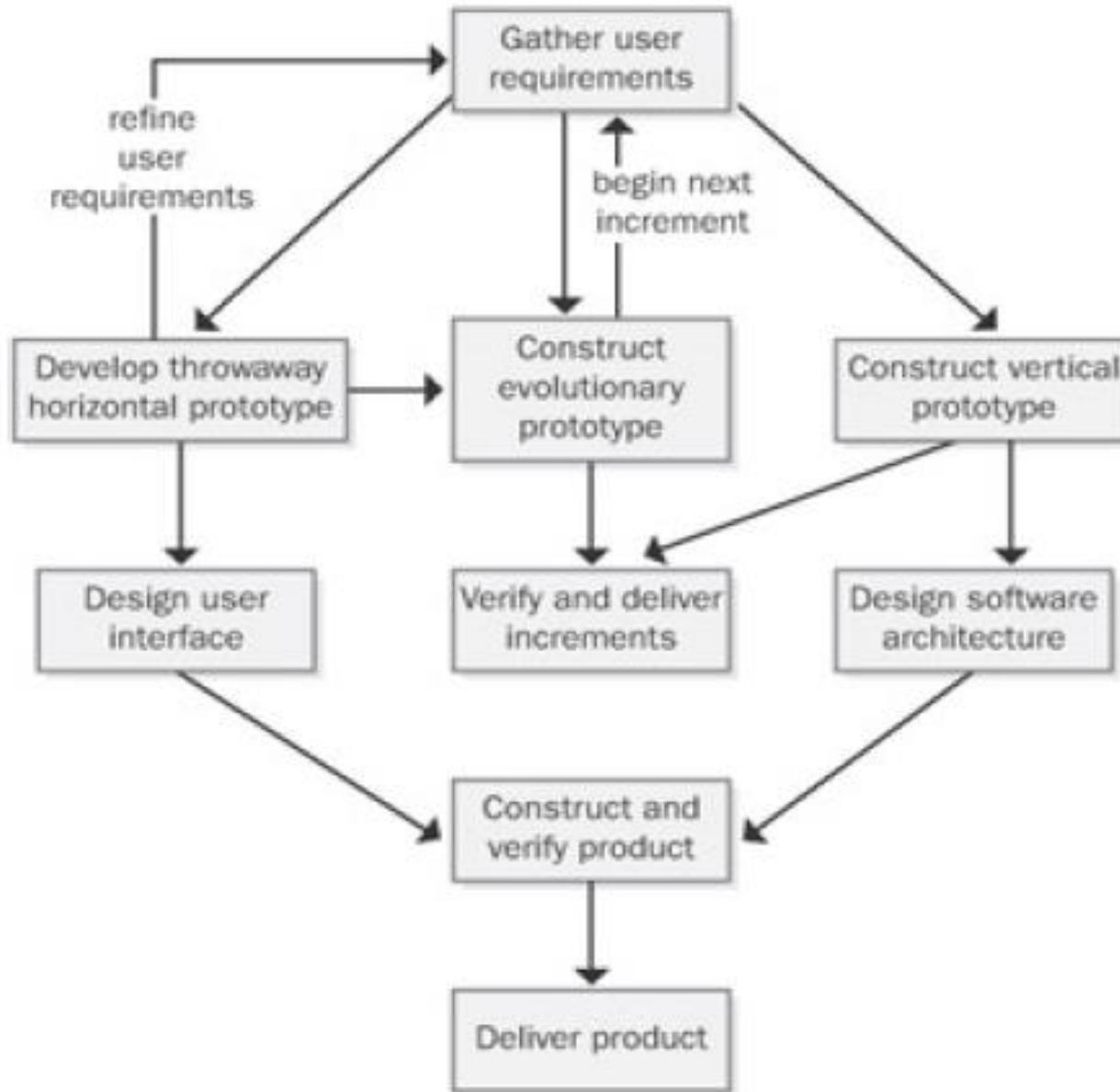
- **Вертикален (structural or proof of concept) прототип**

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

Прилага се в случай на неизяснени въпроси от *архитектурата, оптимизиране* на алгоритъм, *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"><li>Clarify and refine use cases and functional requirements.</li><li>Identify missing functionality.</li><li>Explore user interface approaches.</li></ul>	<ul style="list-style-type: none"><li>Implement core use cases.</li><li>Implement additional use cases based on priority.</li><li>Implement and refine Web sites.</li><li>Adapt system to rapidly changing business needs.</li></ul>

# Typical applications (2)

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

## 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*.
- Но ! *допълнително* 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

(Mock-up - to create user interfaces that show the end user what the software will look like without having to build the software or the underlying functionality. Software UI mockups can range from very simple hand drawn screen layouts, through realistic bitmaps, to semi functional user interfaces developed in a software development tool)

- ‘*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** – 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 извличане на  
изискванията?

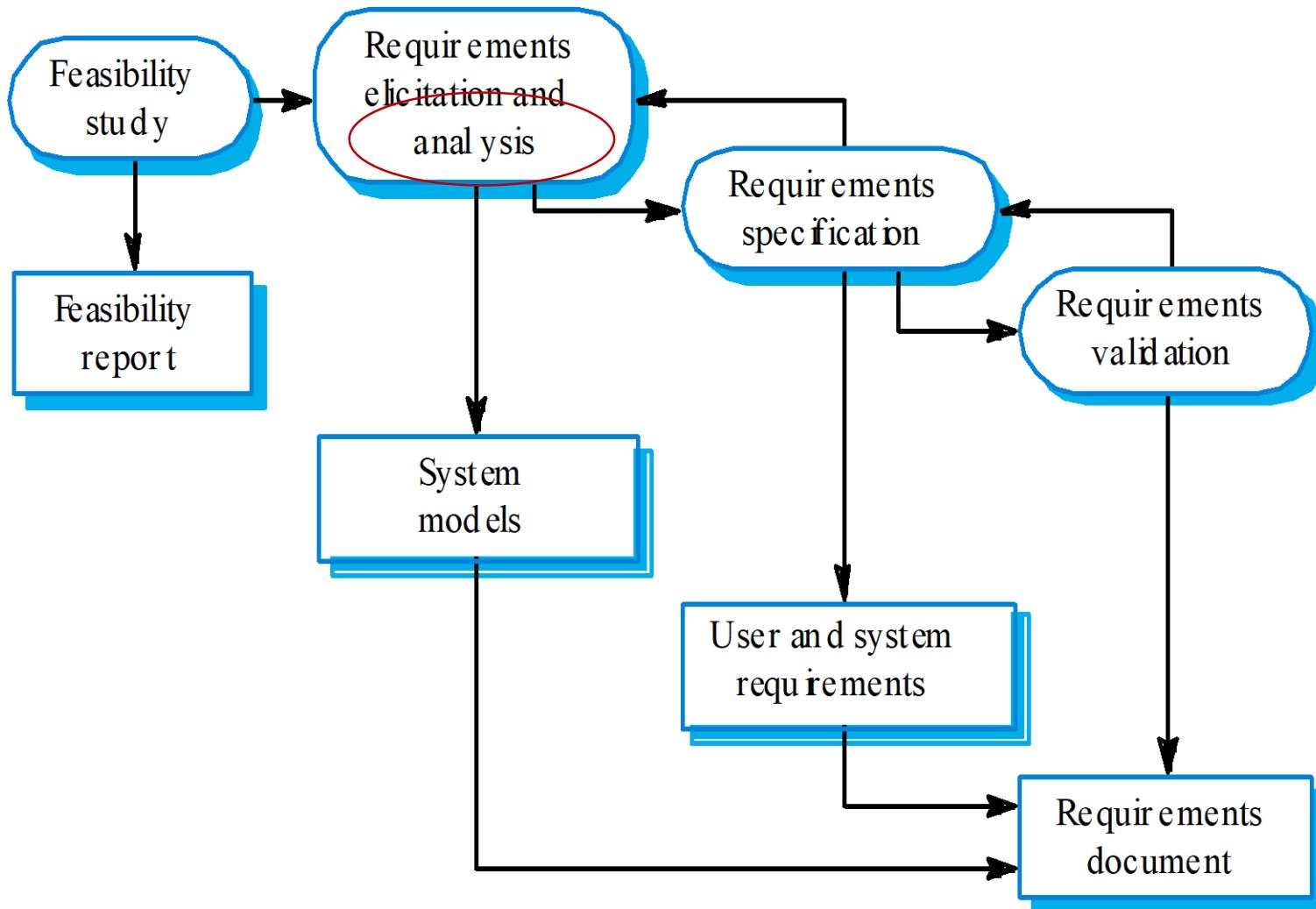
# Идентифициране на изискванията

## Анализ на изискванията

### Лекция 5

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

# Процес на инженеринг на изискванията

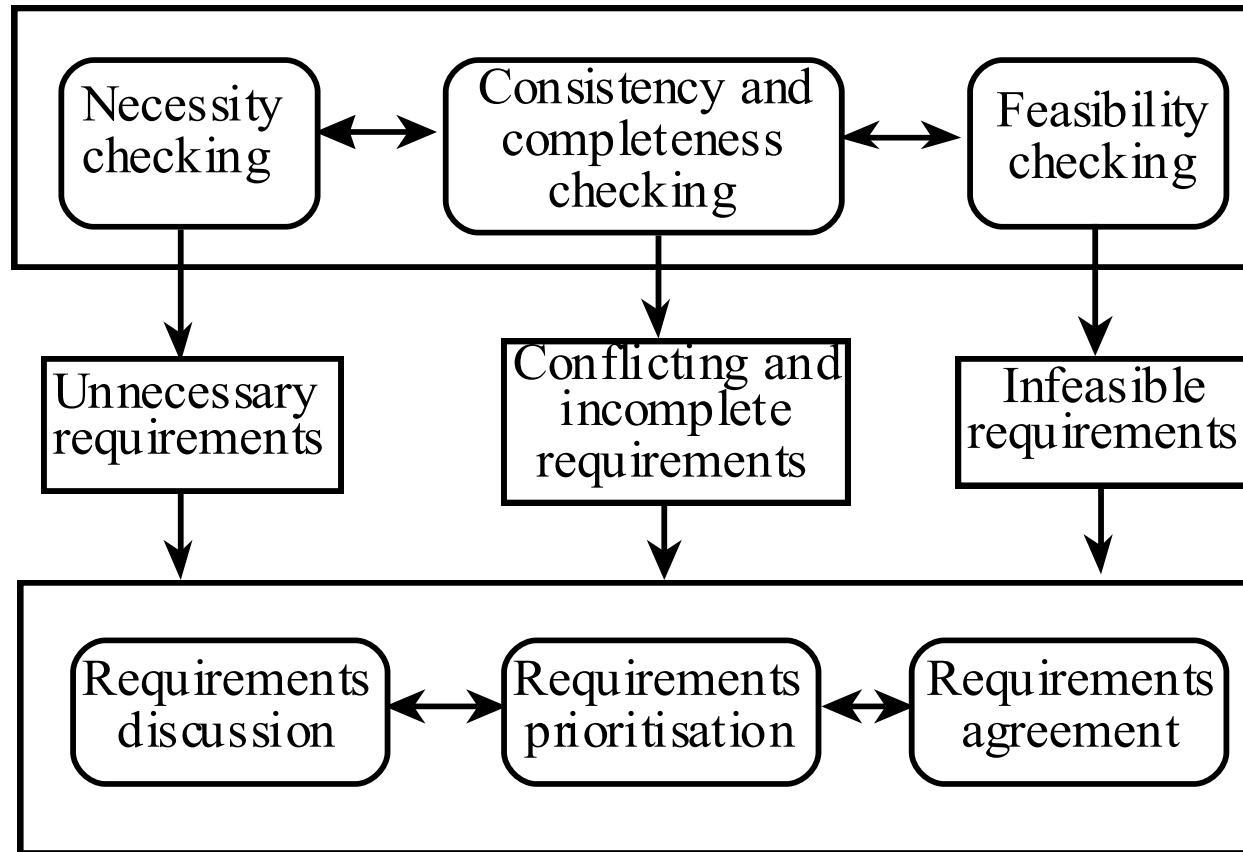


# Анализ на изискванията

- 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**

за да не допускаме изисквания, които *не допринасят за постигането на бизнес целите* на организацията или не спомагат за решаването на специфичния проблем, адресиран от системата.

- **Проверка за съгласуваност и завършеност**

*Съгласуваност* означава, че изискванията не трябва да си противоречат; *завършеност* означава, че никой от необходимите услуги и ограничения не са пропуснати.

- **Проверка за осъществимост**

Дали изискванията *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

## ***Неделимо*** (Cohesive)

Изискването се отнася само за едно единствено нещо.

## ***Пълно***

Изискването е пълно изказано без липсваща информация .

## ***Съгласувано***

Изискването не противоречи на никое друго

изискване и е напълно съгласувано с цялата външна документация.

## ***Без връзки* (атомарни)**

Изискването е *атомарно*, т.е., не съдържа връзки.

## ***Проследимо***

Изискването 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 - 1**

## **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 - 2

- **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, отнасящи 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.