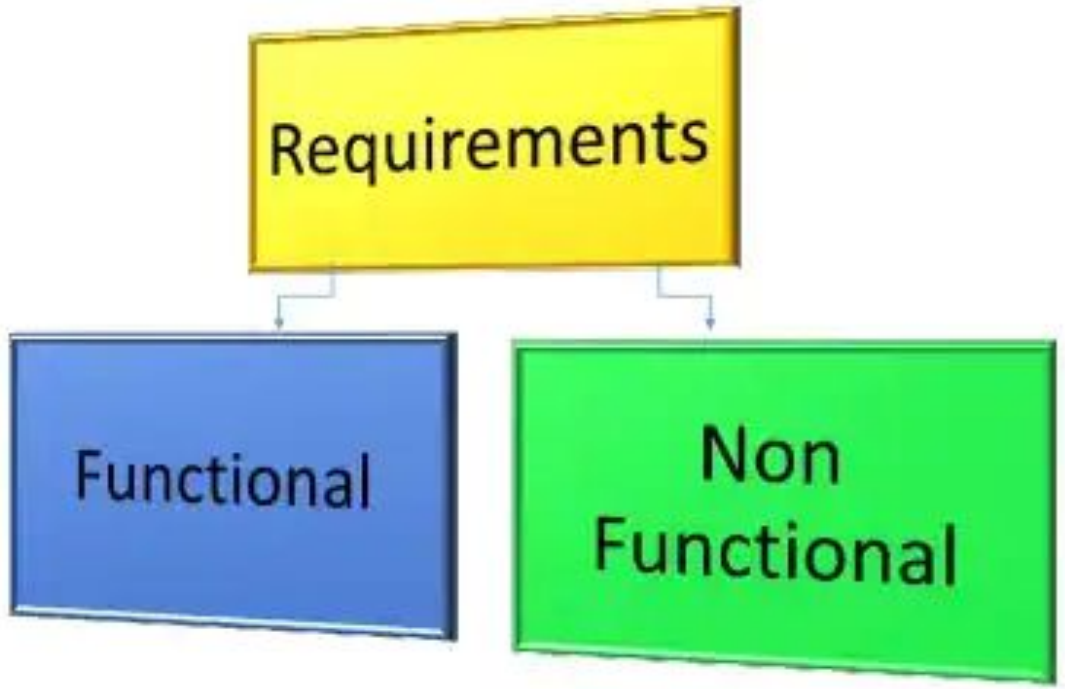
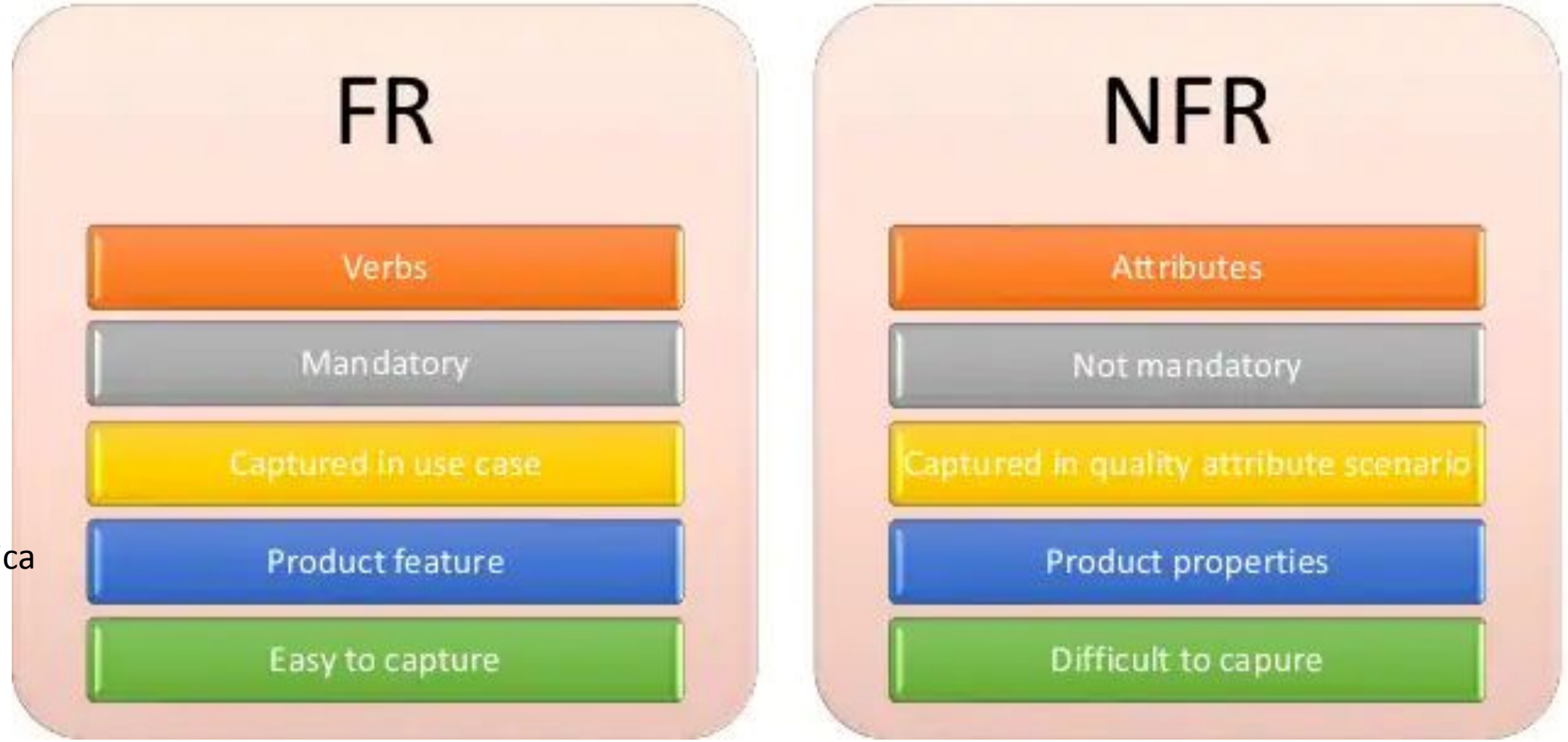
Нефункциональные требования

Non-Functional Requirements (NFRs)

REQUIREMENT TYPES



Non Functional vs. Functional



obligatoriu

caracteristica produsului

Что такое нефункциональные требования?

- ***Функциональный vs.нефункциональный***
  - ***Функциональные требования описывают, что должна делать система***.

-вещи, которые могут быть идентифицированы в вариантах использования

- вещи, которые можно проанализировать, рисуя функциональные диаграммы, диаграммы процессов, диаграммы состояний и т. д.
- ***Функциональные требования больше относятся к отдельным модулям системы***

- ***Нефункциональные требования*** охватывают критерии, которые можно использовать для анализа аспектов, ***связанных с функциональностью системы***, ***а не с ее поведением***.
- Они накладывают ограничения на функциональные требования на уровне проектирования или реализации (например, требования, связанные с производительностью, безопасностью или надежностью).
- ***Нефункциональные требования*** часто называют ***качествами системы***, но также могут называться атрибутами качества, целями качества, характеристиками качества или ограничениями.

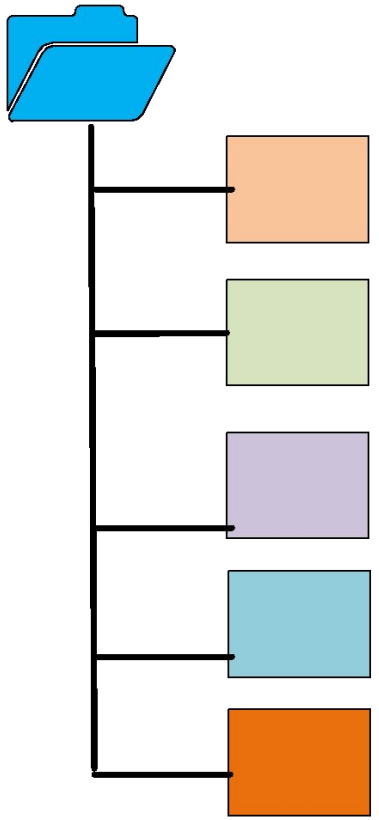
- ***Нефункциональные требования*** — ***это глобальные ограничения на информационную систему,***
- Например:

- затраты на разработку,
- эксплуатационные расходы,
- производительность,
- надежность,
- техническое обслуживание,
- Портабельность,
- Прочность и др.

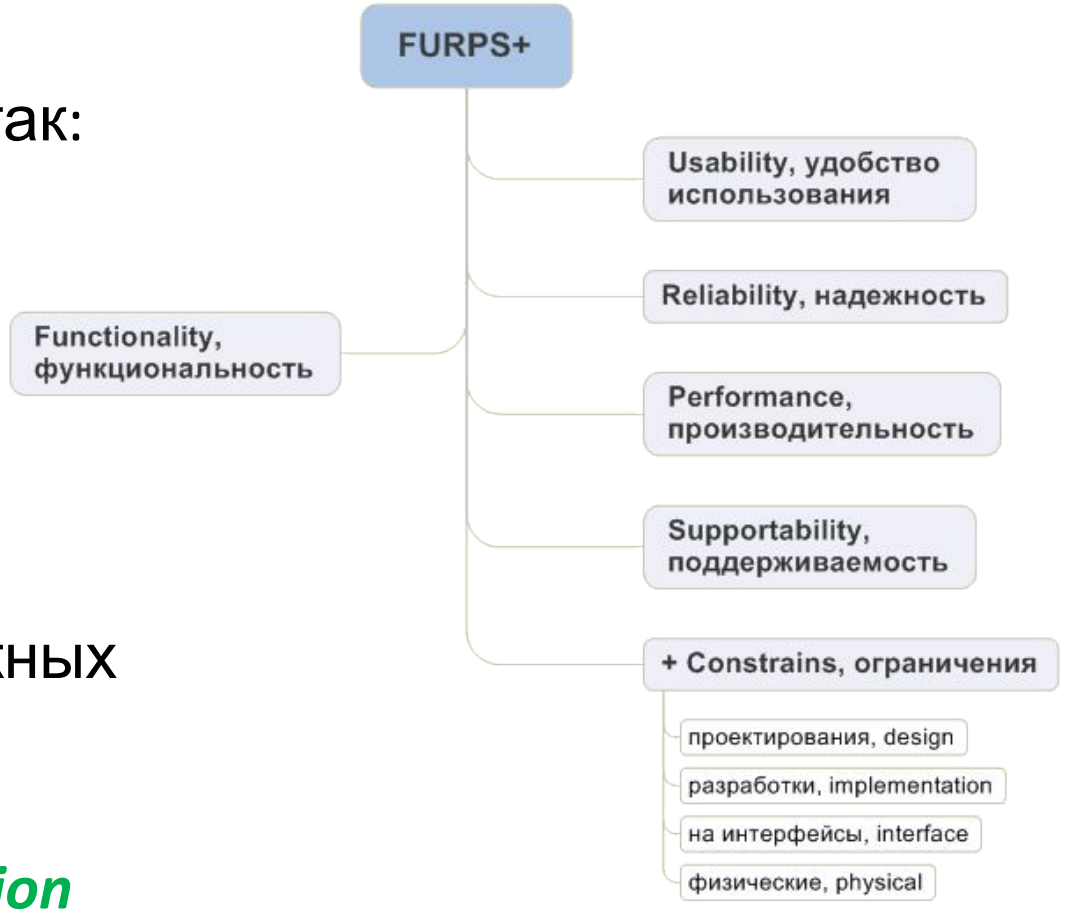
- ***Как правило, они не могут быть реализованы в одном модуле системы и влияют на всю систему***.

Общесистемное ограничение называется нефункциональным требованием.

Модель FURPS объединяет все нефункциональные требования в пять категорий.



Классификация требований к системе FURPS+ была разработана Робертом Грэйди (Robert Grady) из Hewlett-Packard и предложена в 1992 году.
Сокращение FURPS расшифровывается так:
Functionality, функциональность
Usability, удобство использования
Reliability, надежность
Performance, производительность
Supportability, поддерживаемость



+ необходимо помнить о таких возможных ограничениях, как:

ограничения проектирования***, design***
ограничения разработки, ***implementation***
ограничения на интерфейсы***, interface***
физические ограничения***, physical***

- ***Удобство использования***
  - Уровень, предполагаемый опытом пользователя.
  - Стандарты пользовательского интерфейса.
  - Документация предоставлена.
- ***Надежность***
  - Требования безопасности и защищенности.
  - Доступность, устойчивость и надежность системы.
  - Обработка исключений
  - Среднее время наработки на отказ
  - Отказоустойчивость
  - Устойчивость к потере данных

- ***Производительность***
  - количество одновременных поддерживаемых пользователей
  - время отклика
  - количество транзакций в секунду
- ***Доступность***
  - Как система будет расширяться?
  - Кто обслуживает систему?

- **F. Функциональные требования**
- Функциональные требования содержат основные свойства/функции системы. Однако не все они обязательно будут относиться к предметной области системы.
- ***Журналирование, auditing*** — инструменты отслеживания действий пользователей и системы путем записи в журнал безопасности конкретных типов событий.
- ***Лицензирование, licensing*** — средства для отслеживания, приобретения, установки и контроля над использованием лицензий.
- ***Локализация, localization*** — средства поддержки различных естественных языков.
- ***Почта, mail*** — службы отправки и получения сообщений.
- ***Помощь, online help*** — возможность оказывать поддержку пользователей в реальном времени (например, «Необходима система online-помощи»).
- Печать, printing — средства для печати документов.
- Отчетность, reporting — инструменты создания и получения отчетов.
- ***Безопасность, security*** — средства защиты доступа к определенным ресурсам информации.
- **Управление системой, system management** — инструменты, позволяющие управлять приложениями в распределенной среде.
- ***Технологический процесс, workflow*** — поддержка документооборота, включая процессы проверки, визирования и утверждения.

- **U. Удобство использования**
- К удобству использования относятся следующие виды требований:

- ***эстетика и логичность*** пользовательского ***интерфейса***,

- ***защита от человеческого фактора,***

- ***эксплуатационная документация***, ее состав (руководства пользователей, администраторов и др.), отраслевые и гос. стандарты оформления,

- ***квалификация пользователей*** и их обучение,

- ***справочная информация*** в системе.

•R. Надежность

- Надежности включает такие характеристики системы, как:
- сбои:
  - допустимая частота/периодичность сбоев,
  - среднее время сбоев и их серьезность,
  - возможность восстановления системы после сбоев, в т.ч. возможность предварительного резервного копирования данных,
- предсказуемость поведения,
- время готовности системы к работе, режим работы или время доступности системы (например, «Система должна быть доступна 24 часа в сутки 7 дней в неделю»),
- точность вычислений.

•*P. Производительность*

- Производительность системы составляют следующие характеристики:

- ***скорость работы, время отклика системы,***
- ***результативность/эффективность***,
- ***пропускная способность,*** включая общее и допустимое количество одновременно работающих пользователей, количество пользовательских запросов, число обращений системы к БД и объем запрашиваемых/передаваемых данных в единицу времени,
- ***время, необходимое на восстановление*** — скорость восстановления (необходимо отличать эту характеристику P/производительности от характеристик R/надежности «возможность восстановления» и «время доступности»),
- ***время, необходимое для запуска и завершения работы*** — скорость запуска и завершения,
- ***потребление ресурсов***.

• *S. Поддерживаемость*

- К поддержке относятся возможности:
- тестирования,
- расширения — наращивания дополнительного функционала системы,
- масштабирования — тиражирования, например, в филиалах/подразделениях организации,
- адаптации/приспособления к использованию в заданной среде,
- конфигурирования — оперативной, регулярной настройки, переопределения параметров,
- совместимости,
- сопровождения, поддержки работоспособности: исправление ошибок, обновление данных, частота архивации и резервного копирования,
- сервисного обслуживания и ремонта, их удобство,
- установки,
- локализации (например, «Продукт будет поддерживать несколько естественных языков»),
- портативность,
- соответствие международным стандартам.

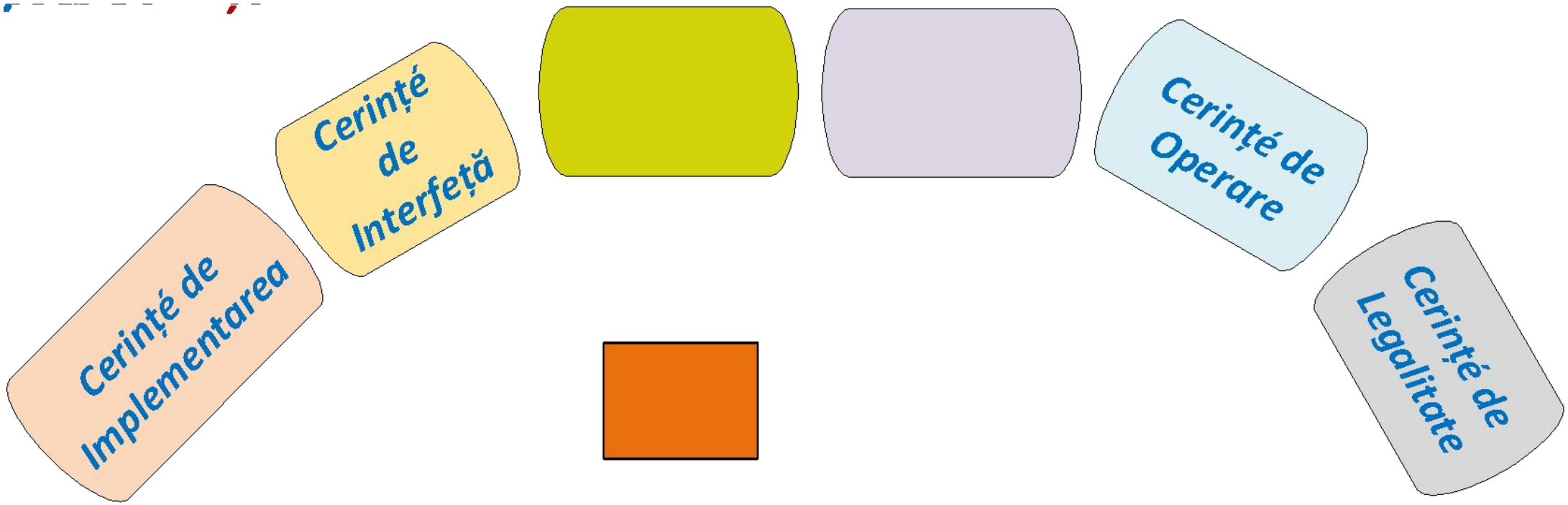
Cerințe nonfuncționale (NFR)

- ***Performanță*** *(cuvinte cheie):*

performance, space, time, throughput,
response, memory, consumption, fast,
index, triggers, storage, low, run, runtime,
perform, execute, mean, peak, compress,
dynamic, offset, reduce, fixing, early, processing

•***Securitate*** *(cuvinte cheie)*

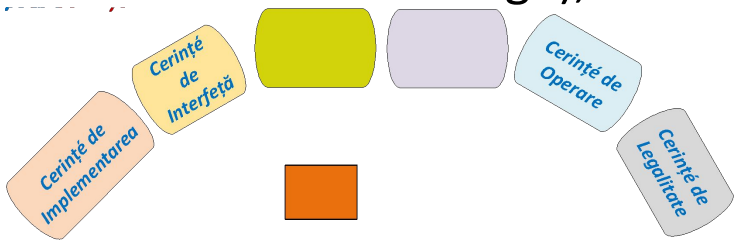
security, confidentiality, integrity, availability, accuracy, completeness, secure, access, registration, authorization, identification, authentication, validation, transaction, user, password, control, encryption, key, spoofing, attack, policy, logging, permission.


Cerințé de Implementarea
Cerințé de Interfață
Cerințé de Operare
Cerințé de Legalitate

•+. Ограничения

- Рассматриваемая классификация выделяет следующие ***4 группы ограничений***:

1. ***Ограничения проектирования***:
   - ограничения на технологии (например, «Хранение необходимо реализовать с помощью реляционной БД»),
   - процесс («RUP»),
   - средства разработки («диаграммы должны создаваться в MS Visio, документация — в MS Word»),
   - прочие.
2. ***Ограничения реализации***, разработки, построение, написания программного кода:
   - стандарты разработки,
   - стандарты качества ПО, в т.ч. кода,
   - языки программирования (например, «Вся бизнес-логика должна быть реализована на языке Visual Basic»),
   - средства разработки («В качестве СУБД должна быть использована Oracle 10g»),
   - ресурсные ограничения,
   - лицензионные ограничения,
   - ограничения на техническое (аппаратное) обеспечение,
   - прочие.

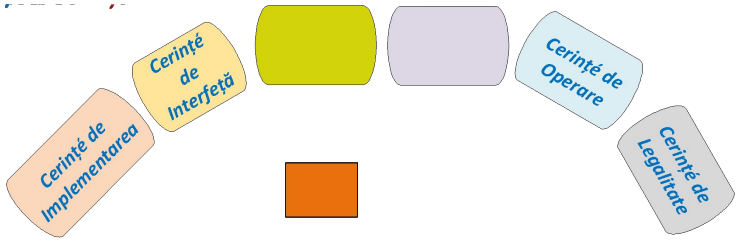


3. Требования к интерфейсам — ограничения (например, на форматы, протоколы) накладываемые необходимостью взаимодействия с другими системами:

- форматы данных,
- протоколы взаимодействия,
- внешние системы,
- прочие.

4. Физические ограничения, накладываемые на технические (аппаратные) средства и окружение системы:

- форма,
- размер,
- вес,
- температурный режим,
- влажность,
- ограничения на вибрацию,
- прочие.



- **Conjoining FURPS and MoSCoW to Analyse and Prioritise Requirements**

•*Метод приоритезации MoSCoW*

- **MoSCoW** является методом приоритезации, используемый в бизнес-анализе и разработке программного обеспечения.
- Данный метод позволяет прийти к общему пониманию важности каждого требования заинтересованными сторонами.

- **The MoSCoW Model**
- MoSCoW is based on another acronym for ***Must have, Should have***, **Could have** and ***Won't have*** (MSCW) and was used originally within the DSDM (Dynamic System Development Method) to consider the priorities from a relatively simple perspective (Stapleton 1998) and is now applied ubiquitously for system requirements prioritisation (Achimugu et al. 2014).
- ***Mo - Must have*** – Defines any requirements that the release of the product or system MUST include.
- *S – Should have* – Is the category of requirements that are high priority and SHOULD if possible, be included within the release.
- ***Co – Could have*** – Are the requirements that are desirable or 'nice-to-have' if they COULD be included without too much divergence or cost.
- ***W - Won't have*** (this time) – Are those requirements that are wanted but WILL NOT be included within the current release.

- **Метод MoSCoW**
- Его создатель, консультант компании Oracle Дай Клегг предлагает ***разбивать дела на четыре группы.***
- Название метода — это **акроним**, составленный по **первым буквам названий категорий**, к **которым автор добавил «о»,** чтобы слово было легко произносить:
- ***Must (должен)*** — срочные и важные дела. Например, подготовка к рабочей презентации, которую вы должны провести завтра перед иностранными партнёрами.
- ***Should (стоит)*** — важные, но несрочные дела. К этой категории относятся серьёзные задачи, которые могут подождать. Если вы не успеете сделать его сегодня, вполне можете перенести на завтра.
- ***Could (мог бы)*** — дела, которые неплохо было бы сделать. Если у вас останутся на это время и силы, займётесь этим сегодня. В противном случае — отложите на потом.
- ***Won't (не стану)*** — всё то, что спокойно можно и не делать. Только это не значит, что к таким делами не нужно приступать никогда. Просто они подождут несколько дней, пока вы заняты более важными задачами.
- Изначально этот подход был создан для эффективного ведения проектов (Oracle — крупнейший производитель ПО). ***Однако метод MoSCoW легко применять и в личных целях.***

- **Как использовать метод MoSCoW**
- ***Разделение задач на категории*** — первая ступень к успеху.
- ***1. Добавляйте задачу только в одну из четырёх категорий***
- Если вы не уверены, куда отнести задачу, всегда добавляйте её в категорию уровнем ниже, например в Should вместо Must. Во время работы вы сможете изменить приоритетность дел, поэтому не стоит переживать об абсолютной точности.
- ***2. Внесите дела в свой календарь***
- Начните с задач в категории ***Must,*** з***атем разберитесь со всем, что попало в Should***, но не забивайте своё расписание под завязку. Остановитесь, как только поймёте, что ваш график становится слишком плотным.
- Некоторые дела могут занять больше времени, чем кажется, поэтому важно оставить свободные промежутки на случай рабочего форс-мажора.
- ***3. Постоянно пересматривайте все четыре категории***
- Приоритеты изменяются. Просматривайте свой календарь каждую неделю и проверяйте актуальность распределения дел по категориям.

- Модель ***FURPS*** - ***MoSCoW***
- ***комбинирование моделей FURPS и MoSCoW*** в рамках одной модели позволяет аналитику учитывать и адаптировать требования к этим приоритетам и обеспечивает отличный инструмент, понятный большинству заинтересованных сторон.
- Этот простой формат позволяет заинтересованным сторонам быстро визуализировать ***матрицу требований и приоритетов*** и более плодотворно участвовать в процессе определения приоритетов и развитии последующего невыполненной работы.

FURPS - MoSCoW Analysis				
	MoSCoW			
FURPS Requirements	**M**	**S**	**C**	**W**
F Functional	Must	Should	Could	Wont
U Usability	Must	Should	Could	Wont
R Reliability	Must	Should	Could	Wont
P Performance	Must	Should	Could	Wont
S Supportability	Must	Should	Could	Wont

Jonathan Dyson (2015)

- ***Подходы к нефункциональным требованиям***

- ***Продукт vs. Процесс*?**
  - ***Подходы, ориентированные на продукт***
    - Сосредоточьтесь на качестве системы (или программного обеспечения).
    - Цель состоит в том, чтобы иметь способ измерения параметров построенного продукта.
  - ***Процессно-ориентированные подходы***
    - Сосредоточьтесь на том, как NFR можно использовать в процессе проектирования.
    - Цель состоит в том, чтобы иметь возможность принимать соответствующие дизайнерские решения.

- ***Количественное vs. качественного***?
  - ***Количественные подходы***
    - Найдите измеримую шкалу для атрибутов качества
    - Рассчитать степень, в которой дизайн соответствует целям качества
  - ***Качественные подходы***
    - Изучите различные взаимосвязи между целями в области качества
    - Определите причину компромиссов и т. д.

- ***Качества программного обеспечения*** (продукта)
- **Возмём конкретный объект**

- ***Например*** - ***Смартфон***
- ***Как бы вы оценили его «качество»?***

✔ ***качество сборки?*** (... Физическая надёжность,...)

✔***эстетическая ценность?*** (цвет, элегантность,...)

✔***пригодность для...?*** (функциональности, комфорт,...)

✔***Елементная база?*** (фирма и страна сборки, ...)

•Следует отметить, что:

- все показатели качества относительны:

- нет абсолютной градации (шкалы).

- иногда мы можем сказать, что **А** лучше, чем **Б**...

-но де-факто ***трудно сказать, насколько лучше!***

•Выражение ***«лучше» не является мерой***

- **Значения для информационных систем и программное обеспечение:**

- ***качество сборки?*** - программное обеспечение не собирается молотком (не осязаемо).
- ***эстетическая ценность?*** - программное обеспечение является интеллектуальным продуктом – *не имеет цветовую гаму* и невидимым .
- ***эстетическая ценность имеет значение только для пользовательского интерфейса*** и является незначительной проблемой
- ***пригодный для цели?*** - Нам нужно знать, какова цель

•*Пригодность (способность)*

Качество программного обеспечения означает соответствие назначению:

- ✔ Программный продукт делает то, что требовалось?
- ✔ делает ли это так, как требуют пользователи?
- ✔ Программный продукт это делает это достаточно надежным?
- ✔ достаточно быстро?
  - уверен достаточно?
- ✔ достаточно безопасно? - емкости, подлежащие измерению
- ✔ Программный продукт будет доступен?
- ✔ будет ли Программный продукт готов, когда он понадобится пользователям?
- ✔ может ли Программный продукт быть изменен по мере необходимости?
- ✔ ....

- ***Качество программного обеспечения*** — это не индивидуальная мера программного продукта — ***оно находится в отношениях***…
- ***оно измеряет взаимосвязь между программным обеспечением и областью его применения.***
- мы не можем измерить это, пока не развернем эту систему/программное обеспечение в его операционной среде……
- ***и качество будет разным в разных средах!***

- ***в процессе проектирования мы должны предвидеть***, насколько хорошо система/программное обеспечение будет соответствовать заявленной цели.
- нужны квалифицированные предикторы (*проектный анализ)*
- Во время анализа требований нам необходимо понять, *как будет измеряться пригодность для заявленной цели:*

✔ Какова предполагаемая цель?

✔ Какие факторы качества будут иметь значение для заинтересованных сторон?

✔ Как эти факторы будут операционализированы?

Факторы качества - Критерий дизайна

- **Факторы качества**
- Это проблемы клиентов, такие как:
- -эффективность,
- интегральность,
- надежность,
- коректность,
- выживаемость,
- полезность,...

- **Критерий дизайна**
- Это технические (ориентированные на разработку) проблемы, такие как: - аномалии

- управление,

- полнота,

- последовательность,

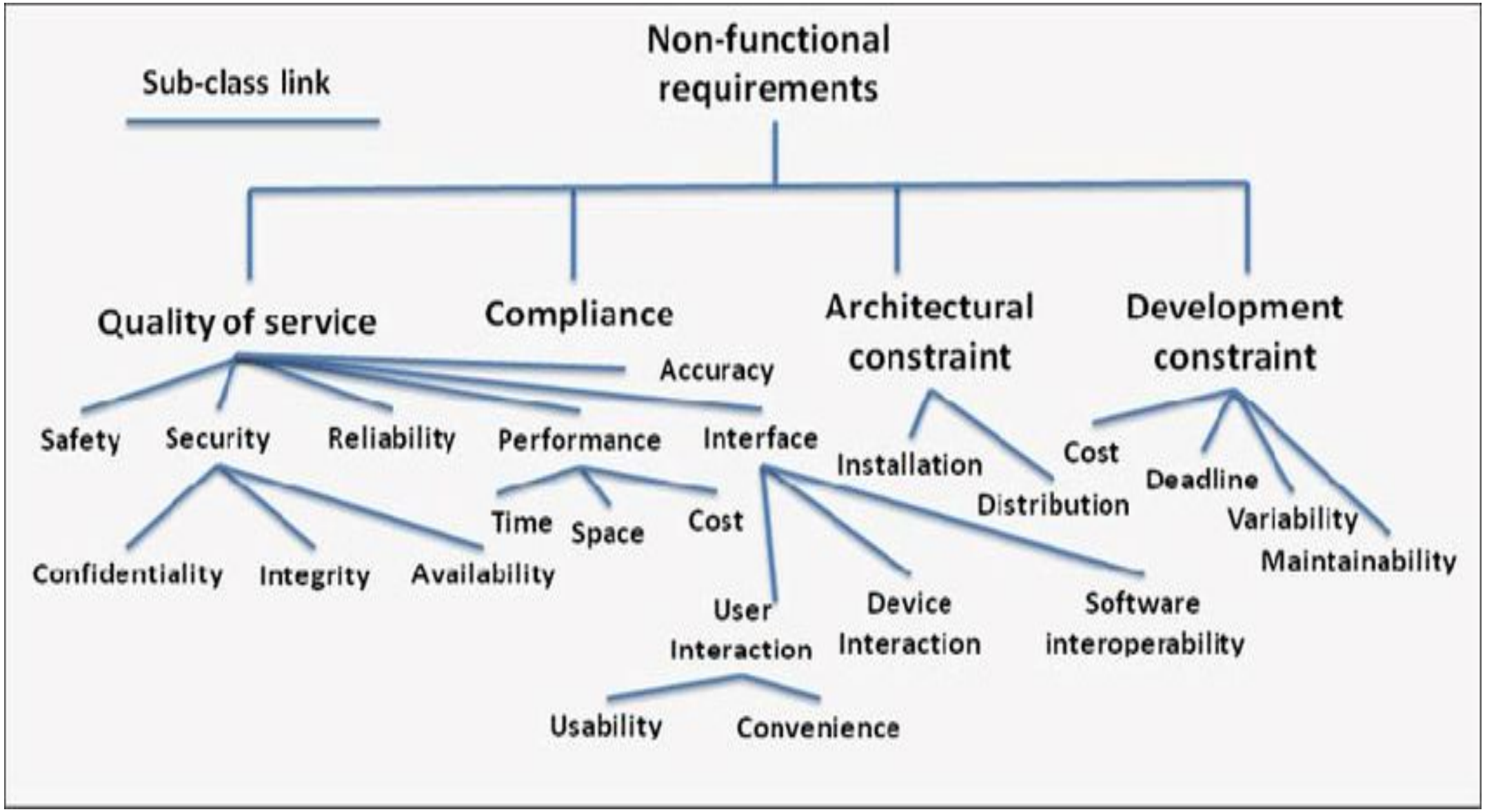
-прослеживаемость,

- видимость,...

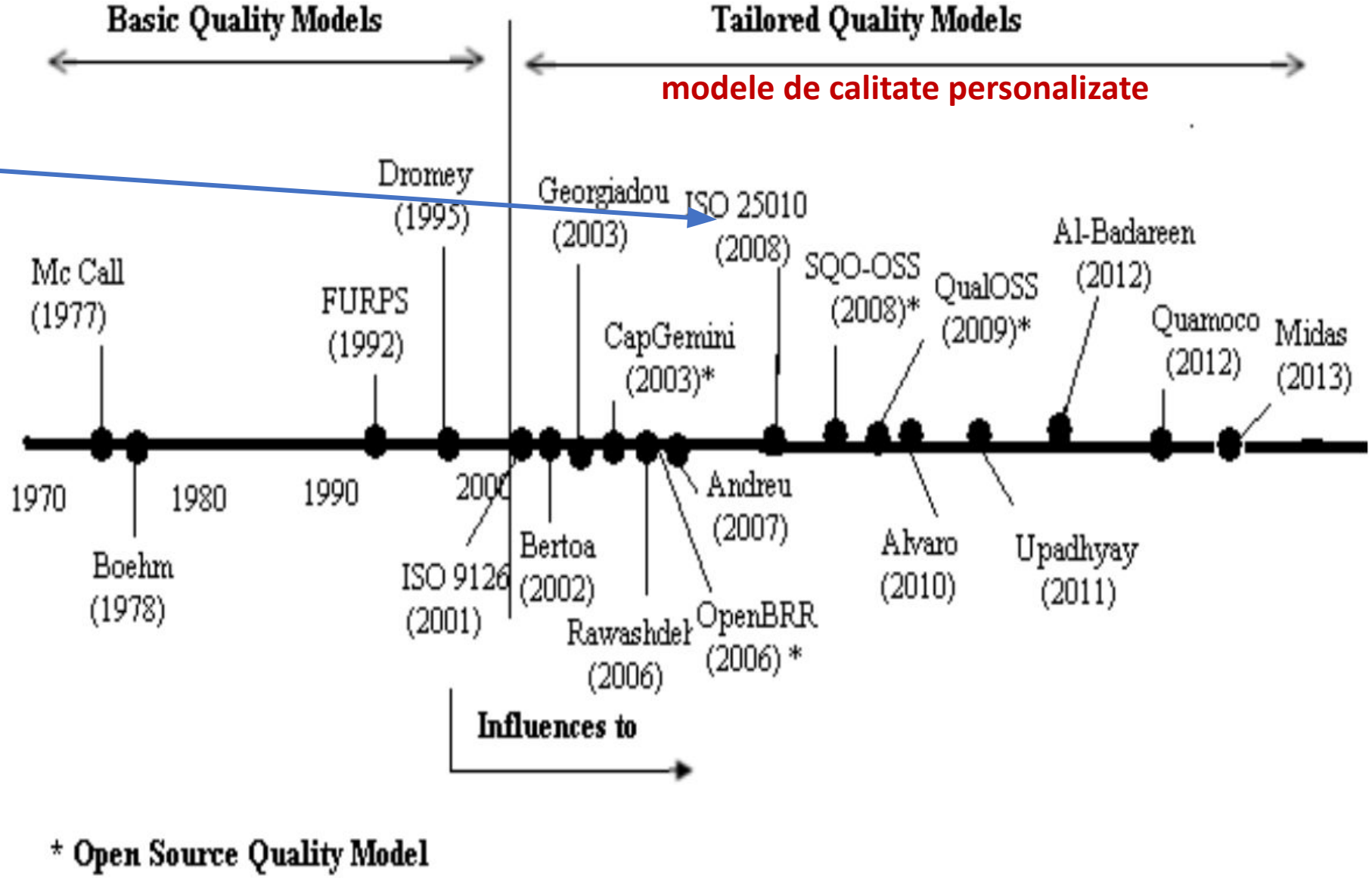
- ***Факторы качества и критерии проектирования взаимозависимы:***
- Каждый фактор зависит от ряда сопутствующих критериев:
- Например:
- **Правильность зависит** от полноты, непротиворечивости, прослеживаемости,...
- **- Проверяемость зависит** от модульности, информативности и простоты...

- ***Некоторые стандартные процедуры, которые могут помочь***
- ***Во время анализа:***
- Определите относительную важность каждого фактора качества - ***С точки зрения клиента!***
- Определите критерии проектирования, от которых зависят эти факторы.
- ***Сделайте требования измеримыми***

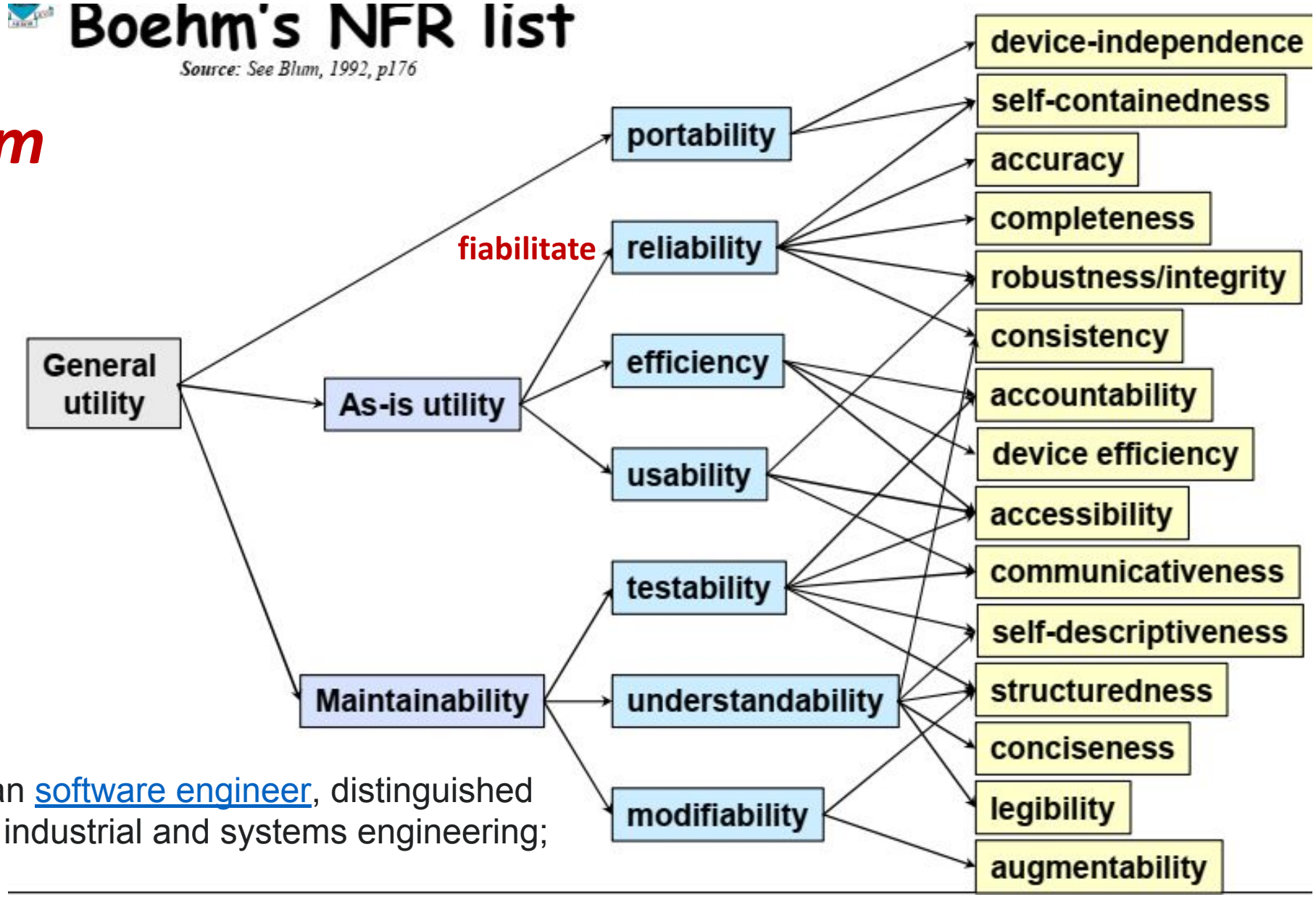
- Нефункциональные требования


Non-functional requirements
Sub-class link
Quality of service
Compliance
Architectural constraint
Development constraint
Accuracy
Safety
Security
Reliability
Performance
Interface
Installation
Distribution
Cost
Deadline
Variability
Maintainability
Confidentiality
Integrity
Availability
Time
Space
Cost
User Interaction
Device Interaction
Software interoperability
Usability
Convenience

ISO 2510 de atribuit calitatea de Basic

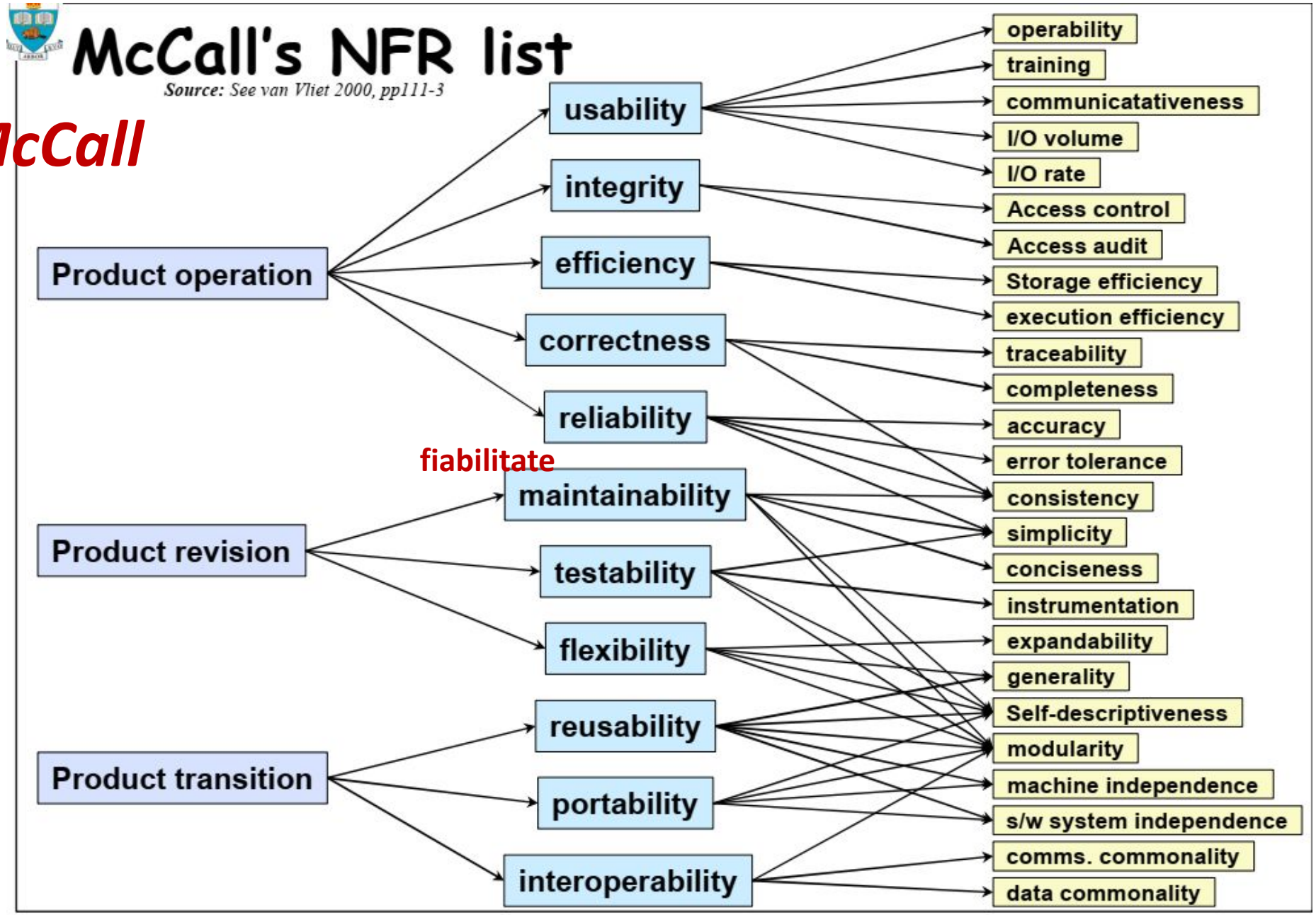


Abordarea Boehm

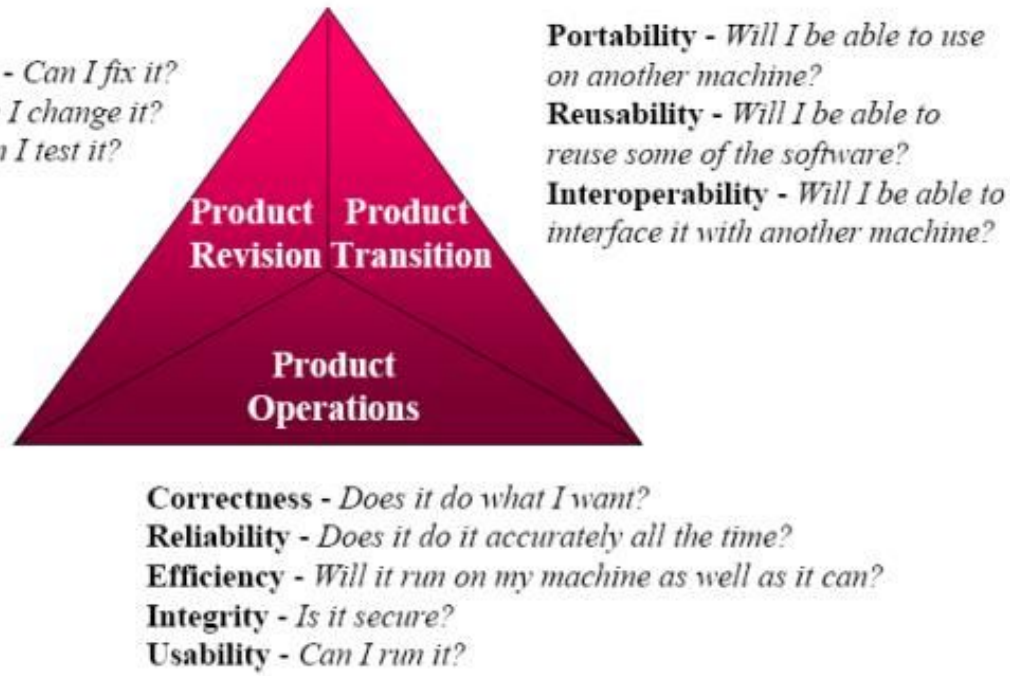


Barry W. Boehm is an American software engineer, distinguished professor of computer science, industrial and systems engineering;

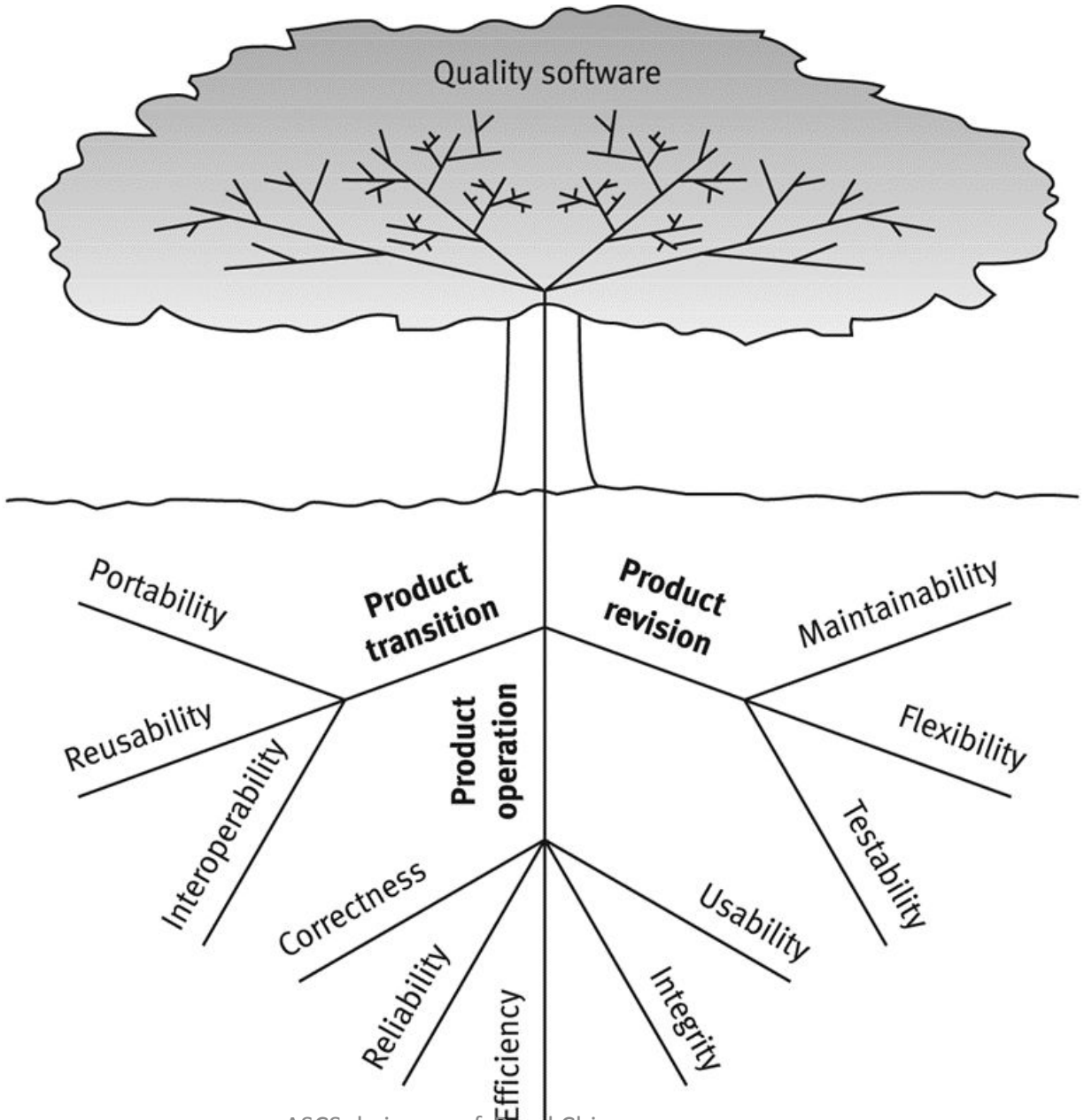
Abordarea McCall



McCall's Quality Model



Jim McCall a produs acest model pentru US Air Force, intenția a fost de a reduce decalajul dintre utilizatori și dezvoltatori.
A încercat să cartografieze și să dezvolte


Quality software
Portability
Product transition
Product revision
Maintainability
Reusability
Interoperability
Product operation
Flexibility
Testability
Correctness
Reliability
Efficiency
Integrity
Usability

Делаем требования измеримыми

Расплывчатые представления о качестве должны быть преобразованы в измеримые представления.

Концепции качества

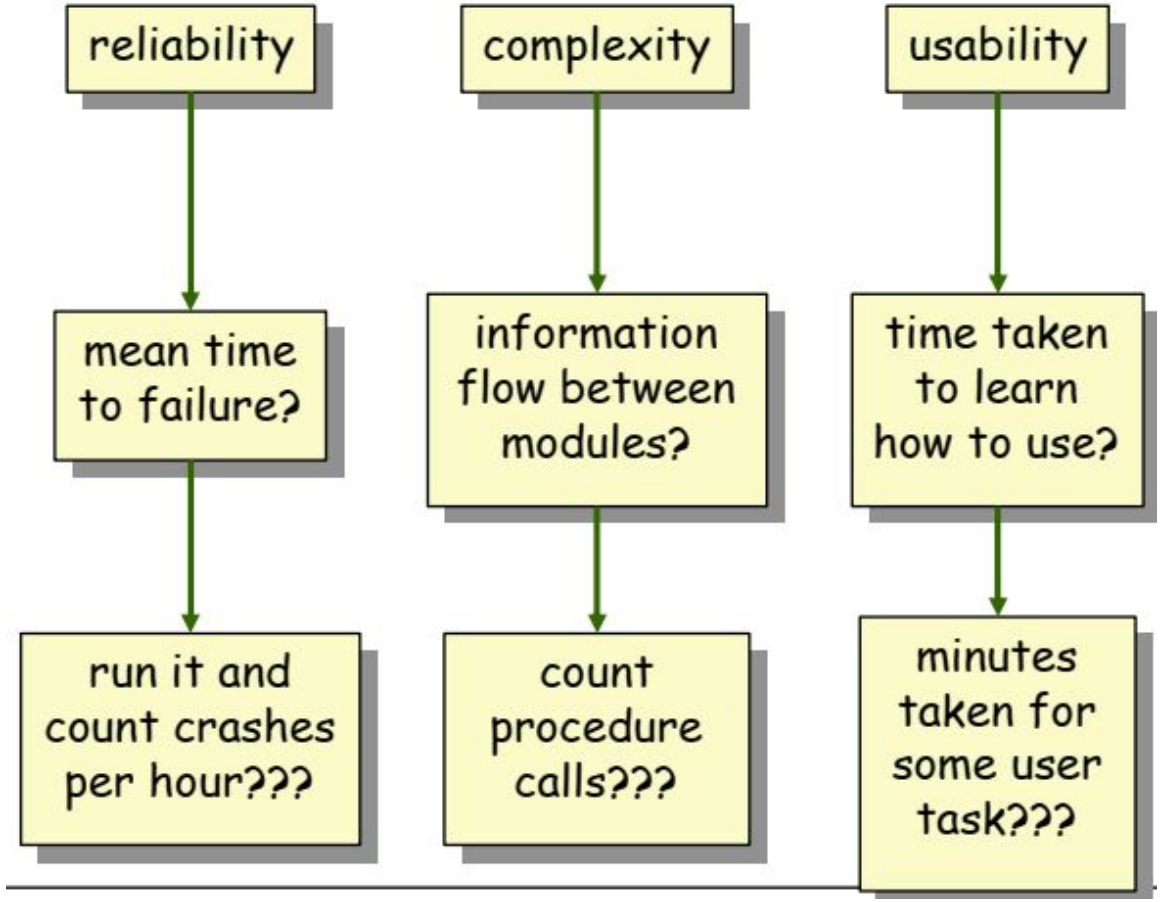
(абстрактные понятия свойств качества)

Измеряемые величины

(определить некоторые значения)

Реализация метрик

Счетчики взяты из какого-то дизайна



- Примеры
- показателей

Calitate	Măsurabilitatea
Viteză (Speed)	tranzacții/sec timp de raspuns timpul de reîmprospătare ecran
dimensiune	Kbytes numărul de cipuri RAM
Ușor de utilizat	Timp de antrenament numărul de cadre de ajutor
Fiabilitate	timp mediu până la eșec, probabilitatea de indisponibilitate rata de eșec, disponibilitate
Robusteţe	timpul de repornire după defecțiune procentul de evenimente care cauzează eșec
Portabilitate	procent de declarații dependente de țintă numărul de sisteme țintă

- Пример: ***измерение надежности***
- Определение- ***Надежность - Способность системы последовательно вести себя приемлемым для пользователя образом при работе в среде, для которой она предназначена.***
- Комментарии:

- ***Надежность*** может быть определена в процентах (скажем, 99,999%). Это может иметь разные значения для разных приложений;

- ***Телефонная сеть:*** вся сеть может выходить из строя не чаще, в среднем, 1 час в год, но отказы отдельных коммутаторов могут происходить гораздо чаще.
- ***Система мониторинга пациентов***: система может выходить из строя не более 1 часа в год, но ***в этих случаях врачи/медсестры должны быть предупреждены о сбое***

- ***Делаем требования измеримыми***

- Определите «критерии соответствия» для каждого требования

- Укажите «критерии соответствия» для каждого требования

Например.

для нового программного обеспечения банкомата

- ***Требование:*** «Программное обеспечение должно быть интуитивно понятным»

- - Критерии соответствия: «95% клиентов банка смогут снимать деньги и вносить чеки в течение двух минут после первого знакомства с продуктом».

- ***Выбор критериев соответствия***

- Заинтересованные стороны редко бывают настолько конкретными,

- Правильные критерии могут быть неочевидны:

- Вещи, которые легко измерить, не обязательно нужны заинтересованным сторонам.

- Стандартные значения не нужны для того, чего хотят заинтересованные стороны

- Заинтересованные стороны должны построить свои собственные сопоставления на основе требований, чтобы соответствовать критериям.

- **Примеры нефункциональных требований**
- ***Требования к качеству***

I. ***Эксплуатационная Безопасность*** - Система не должна работать, если температура наружного воздуха падает ниже 4 градусов С. - Система не должна работать в случае пожара. - Система должна перестать работать, если наблюдаются очевидные атаки.

II. ***Безопасность доступа к данным.*** Права доступа к данным могут быть изменены только системным администратором данных. - Резервное копирование всех системных данных должно выполняться каждые 12 часов, а резервные копии должны храниться в безопасном месте, которое не находится в том же здании, что и система. - Все внешние коммуникации между сервером данных и клиентами должны быть зашифрованы.

III. ***Надежность***. Система должна иметь доступность 999/1000 или 99,9%. Это требование надежности, которое предполагает, что из каждых 1000 запросов на обслуживание должно быть выполнено 999.

IV. ***Производительность.*** Система будет обрабатывать минимум X транзакций в секунду. -Система будет работать 24 часа в сутки, 7 дней в неделю.

- ***Требования соответствия.***
- **I. Законодательные и нормативные требования:** все изменения, внесенные в пользовательские данные, должны храниться не менее 5 лет.
- **II. Лицензионные требования:** Процесс «обновление заказа» будет лицензироваться для N одновременных пользователей;
- **III. Архитектурные требования:** Постоянство данных будет обеспечиваться с помощью реляционной базы данных;

2.Interface Requirements:

- Buttons
- Images
- Colors
- Labels
- Panels
- Icons
- Short cut keys.

- **References**

- Achimugu, P., Selamat, A., Ibrahim, R., and Mahrin, M.N. (2014) ‘A Systematic Literature Review of Software Requirements Prioritization Research’. *Information and Software Technology* [online] 56 (6), 568–585. available from <https://linkinghub.elsevier.com/retrieve/pii/S0950584914000354> [7 January 2019]
- Grady, R.B. (1987) ‘Measuring and Managing Software Maintenance’. *IEEE Software* [online] 4 (5), 35–45. available from <http://ieeexplore.ieee.org/document/1695821/> [7 January 2019]
- Stapleton, J. (1998) *DSDM, Dynamic Systems Development Method: The Method in Practice*. Reprint. Harlow: Addison-Wesley
- Zimmer, B. (1989) ‘Software Quality and Productivity Analysis at Hewlett-Packard’. in *[1989] Proceedings of the Thirteenth Annual International Computer Software & Applications Conference*, ‘[1989] Thirteenth Annual International Computer Software & Applications Conference’ [online] held 1989 at Orlando, FL, USA. IEEE Comput. Soc. Press, 628–632. available from <http://ieeexplore.ieee.org/document/65157/> [7 January 2019]