

# Non-Functional Requirements (NFRs)

- What are non-functional requirements
- Product-oriented qualities
- Process-oriented qualities
- Traceability between requirements

## 非功能性需求 (NFR)

- 什么是非功能性需求
- 以产品为导向的品质
- 以过程为导向的品质
- 需求之间的可追溯性

# What are Non-functional Requirements?

## → Functional vs. Non-Functional

### ↳ **Functional requirements describe what the system should do**

- Things that can be captured in use cases
- Things that can be analyzed by drawing sequence diagrams, statecharts, etc.
- Functional requirements will probably trace to individual chunks of a program

### ↳ **Non-functional requirements are global constraints on a software system**

- e.g. development costs, operational costs, performance, reliability, maintainability, portability, robustness etc.
- Often known as the “-ilities”
- Usually cannot be implemented in a single module of a program

# 什么是非功能性需求？

## → 功能性与非功能性

功能需求描述了系统应该做什么

- Ø 可以在用例中捕获的东西 Ø 可以通过绘制序列图、状态图等进行分析的东西
- Ø 功能需求可能会追溯到程序的各个块

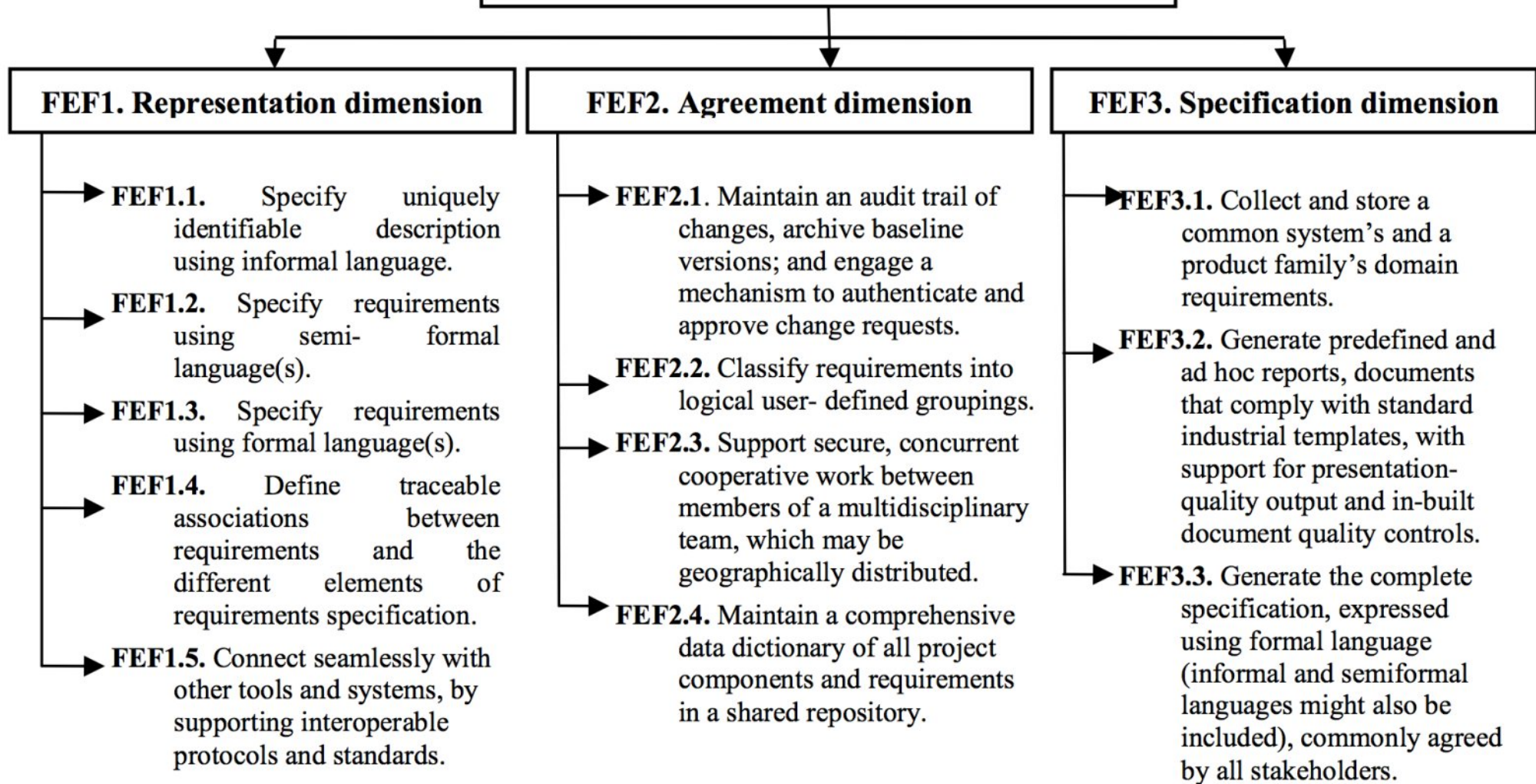
非功能性需求是对软件系统的全局约束

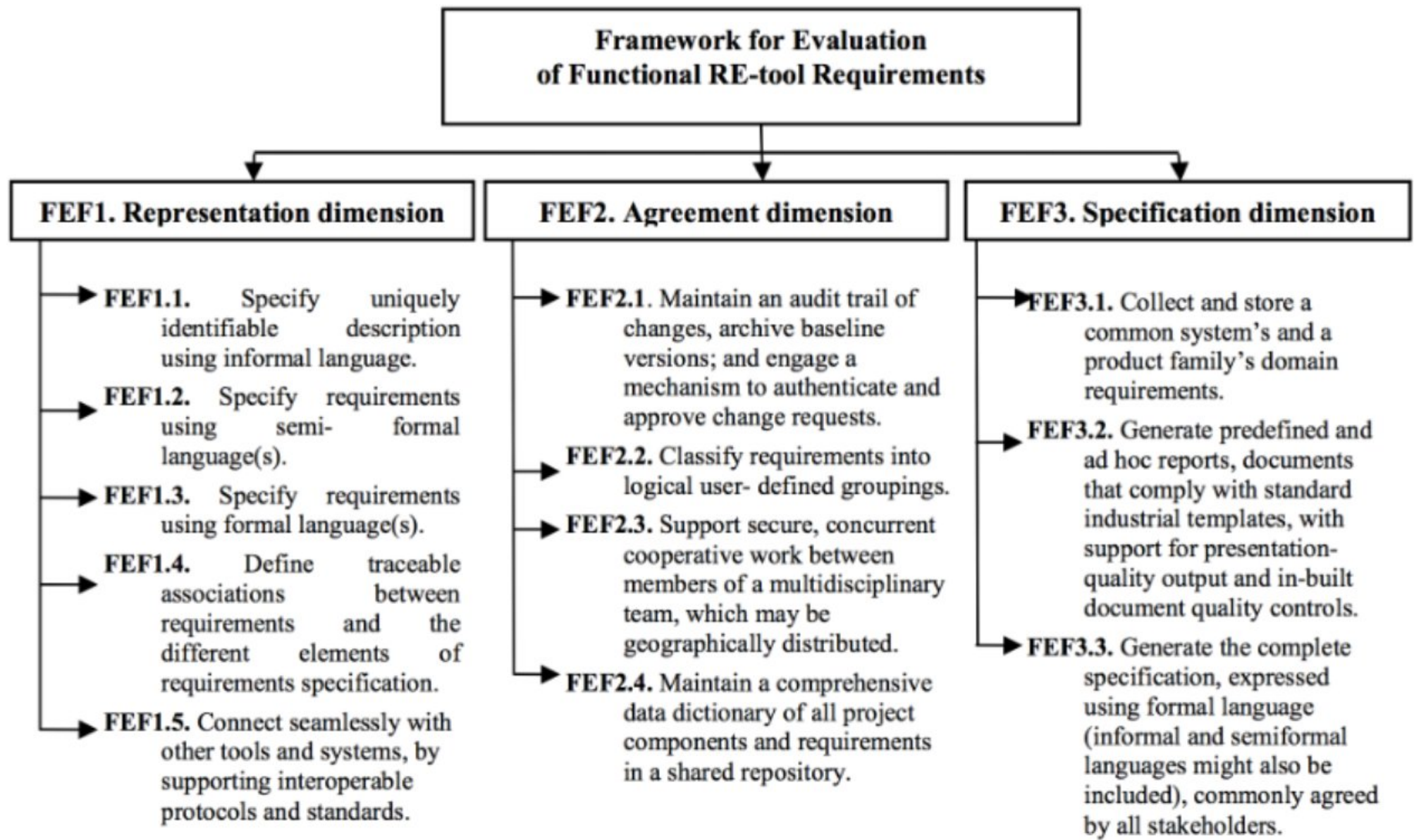
- Ø 例如开发成本、运营成本、性能、可靠性、可维护性、可移植性、稳健性等。

Ø 通常被称为“ -ilities ”

- Ø 通常不能在程序的单个模块中实现

## Framework for Evaluation of Functional RE-tool Requirements





# What are Non-functional Requirements?

## → Functional vs. Non-Functional

↳ **Functional requirements describe what the system should do**

- things that can be captured in use cases
- things that can be analyzed by drawing sequence diagrams, statecharts, etc.
- Functional requirements will probably trace to individual chunks of a program

↳ **Non-functional requirements are global constraints on a software system**

- e.g. development costs, operational costs, performance, reliability, maintainability, portability, robustness etc.
- Often known as the “-ilities”
- Usually cannot be implemented in a single module of a program



# 什么是非功能性需求？

## → 功能性与非功能性

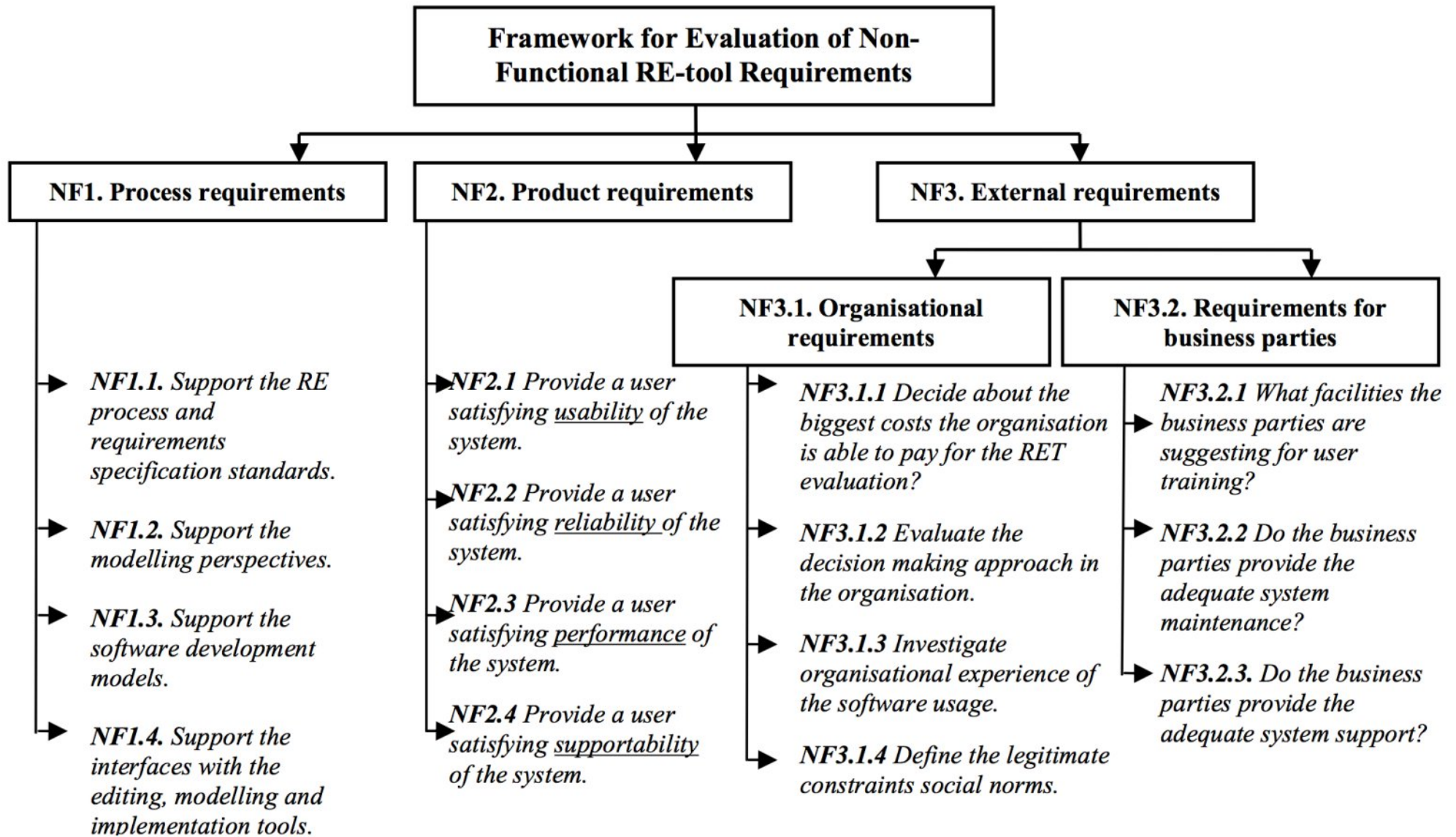
功能需求描述了系统应该做什么

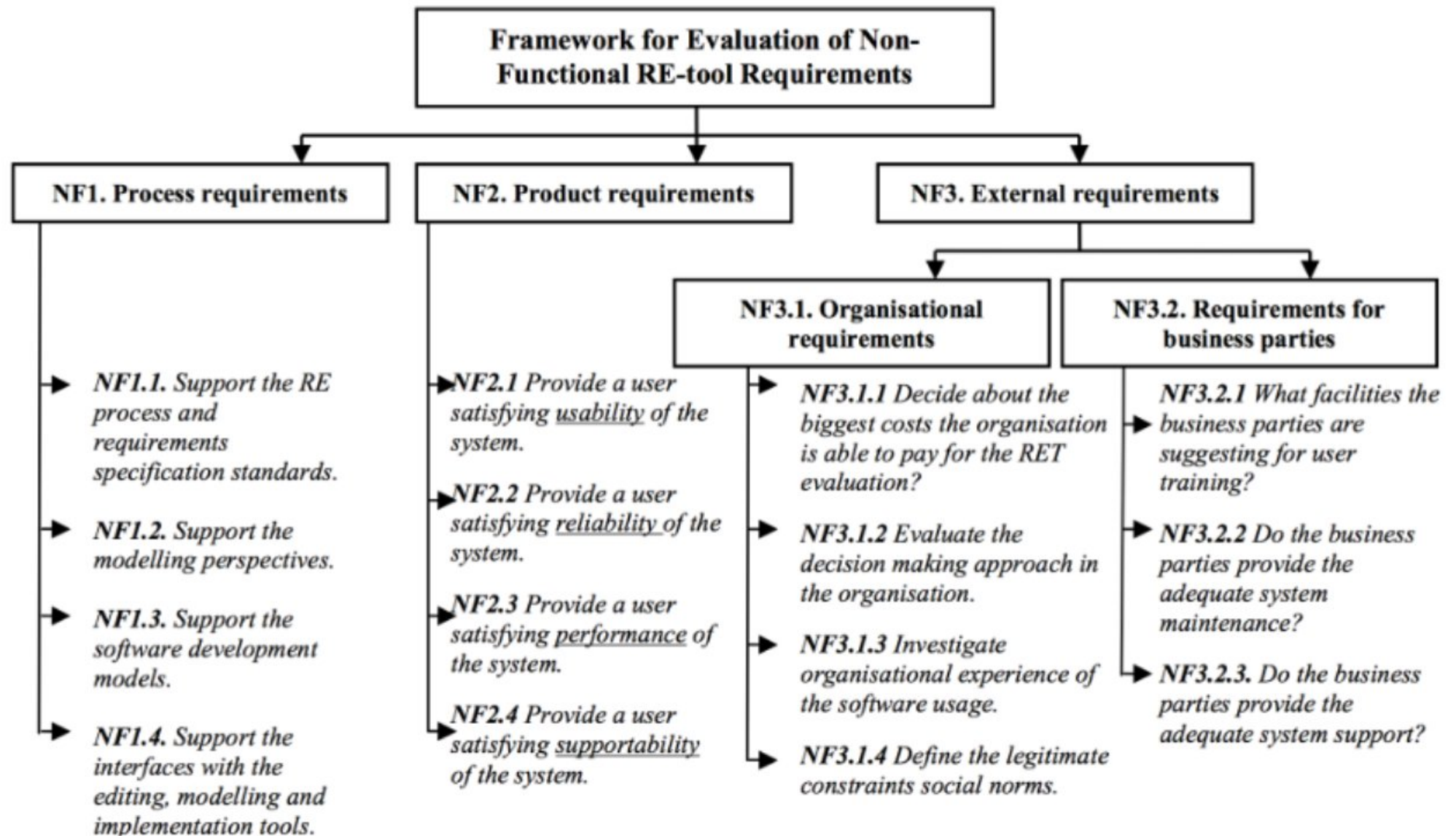
- Ø 可以在用例中捕获的东西 Ø 可以通过绘制序列图、状态图等进行分析的东西
- Ø 功能需求可能会追溯到程序的各个块

非功能性需求是对软件系统的全局约束

- Ø 例如开发成本、运营成本、性能、可靠性、可维护性、可移植性、稳健性等。
- Ø 通常被称为“ -ilities ”
- Ø 通常不能在程序的单个模块中实现







# The challenge of NFRs

↳ **Hard to model**

↳ **Usually stated informally, and so are:**

- often contradictory
- difficult to enforce during development
- difficult to evaluate for the customer prior to delivery

↳ **Hard to make them measurable requirements**

- We'd like to state them in a way that we can measure how well they've been met

## NFR 的挑战

很难建模

通常非正式地表述，例如：

- Ø 经常矛盾
- Ø 开发过程中难以执行
- Ø 交付前难以为客户进行评估

很难让它们成为可衡量的要求

- Ø 我们希望以一种我们可以衡量他们的满足程度的方式来陈述它们

# Example NFRs

## → Interface requirements

↳ **how will the new system interface with its environment?**

- User interfaces and “user-friendliness”
- Interfaces with other systems

## → Performance requirements

↳ **time/space bounds**

- workloads, response time, throughput and available storage space
- e.g. “the system must handle 1,000 transactions per second”

↳ **reliability**

- the availability of components
- integrity of information maintained and supplied to the system
- e.g. “system must have less than 1hr downtime per three months”

↳ **security**

- E.g. permissible information flows, or who can do what

↳ **survivability**

- E.g. system will need to survive fire, natural catastrophes, etc

# NFR 示例

## → 接口要求

ä新系统将如何与其环境交互?

Ø 用户界面和“ 用户友好性” Ø 与其他系统的接口

## → 性能要求

时间/空间界限

Ø 工作负载、响应时间、吞吐量和可用存储空间“ 系统每秒必须处理 1,000 笔交易”

可靠性

Ø 组件的可用性 Ø 维护和提供给系统的信息的完整性 Ø 例如“ 系统每三个月的停机时间必须少于 1 小时”

安全

Ø 例如允许的信息流，或者谁可以做什么

生存能力

Ø 例如系统需要能够抵御火灾、自然灾害等

# Example NFRs

## → Operating requirements

- ↳ physical constraints (size, weight),
- ↳ personnel availability & skill level
- ↳ accessibility for maintenance
- ↳ environmental conditions

## → Lifecycle requirements

- ↳ “Future-proofing”
  - Maintainability
  - Enhanceability
  - Portability
  - *expected market or product lifespan*
- ↳ limits on development
  - development time limitations,
  - resource availability
  - methodological standards

## → Economic requirements

- ↳ e.g. restrictions on immediate and/or long-term costs.



## NFR 示例

### → 操作要求

ä 物理限制 (尺寸、重量), ä 人员可用性和技能水平 ä 维护的可达性 ä 环境条件

### → 生命周期要求 ä “面向未来”

- Ø 可维护性
- Ø 增强性
- Ø 便携性
- Ø 预期的市场或产品寿命
- Ø 发展的限制
  - Ø 开发时间限制, Ø 资源可用性Ø
- 方法标准

### → 经济要求

例如对即时和/ 或长期成本的限制。

# Approaches to NFRs

## → Product vs. Process?

### ↳ **Product-oriented Approaches**

- Focus on system (or software) quality
- Aim is to have a way of measuring the product once it's built

### ↳ **Process-oriented Approaches**

- Focus on how NFRs can be used in the design process
- Aim is to have a way of making appropriate design decisions

## → Quantitative vs. Qualitative?

### ↳ **Quantitative Approaches**

- Find measurable scales for the quality attributes
- Calculate degree to which a design meets the quality targets

### ↳ **Qualitative Approaches**

- Study various relationships between quality goals
- Reason about trade-offs etc.

## NFR 的方法

### → 产品与流程?

以产品为导向的方法

Ø 关注系统（或软件）质量 Ø 目标是在产品构建后有一种测量产品的方法  
Ø 面向过程的方法 Ø 关注如何在设计过程中使用NFR Ø 目标是有一种方法进行适当的设计决定

### → 定量与定性?

定量方法

Ø 找到质量属性的可测量尺度 Ø 计算设计满足质量目标的程度 Ø 定性方法  
Ø 研究质量目标之间的各种关系 Ø 权衡等的原因

# Software Qualities

## → Think of an everyday object

↳ e.g. a chair

↳ **How would you measure it's "quality"?**

- construction quality? (e.g. strength of the joints,...)
- aesthetic value? (e.g. elegance,...)
- fit for purpose? (e.g. comfortable,...)

## → All quality measures are relative

↳ there is no absolute scale

↳ **we can sometimes say A is better than B...**

- ... but it is usually hard to say how much better!

## → For software:

↳ **construction quality?**

- software is not manufactured

↳ **aesthetic value?**

- but most of the software is invisible
- aesthetic value matters for the user interface, but is only a marginal concern

↳ **fit for purpose?**

- need to understand the purpose

# 软件质量

## → 想想一个日常物品

例如一把椅子

您如何衡量它的“质量”？

Ø 施工质量？（例如关节的强度.....） Ø 美学价值？  
（例如优雅，.....） Ø 适合目的吗？（例如舒适，.....）

## → 所有质量指标都是相对的

Ø 没有绝对的尺度

我们有时可以说 **A** 比 **B** 更好.....

Ø .....但通常很难说好多少！

## → 对于软件：

ü 施工质量？

Ø 软件不是制造出来的 Ø 审美价值？

Ø 但大多数软件是看不见的 Ø 审美价值对用户界面很重要，但这只是一个边缘问题

Ø 适合目的吗？

Ø 需要了解目的

# Fitness

## → **Software quality is all about fitness to purpose**

- ↳ does it do what is needed?
- ↳ does it do it in the way that its users need it to?
- ↳ does it do it reliably enough? fast enough? safely enough? securely enough?
- ↳ will it be affordable? will it be ready when its users need it?
- ↳ can it be changed as the needs change?

## → **Quality is not a measure of software in isolation**

- ↳ **it measures the relationship between software and its application domain**
  - cannot measure this until you place the software into its environment...
  - ...and the quality will be different in different environments!
- ↳ **during design, we need to *predict* how well the software will fit its purpose**
  - we need good quality predictors (design analysis)
- ↳ **during requirements analysis, we need to *understand* how fitness-for-purpose will be measured**
  - What is the intended purpose?
  - What quality factors will matter to the stakeholders?
  - How should those factors be operationalized?

## 健康

### → 软件质量关键在于是否符合目的——它是否满足了需要？

它是否按照用户需要的方式进行操作？

ä 是否足够可靠？够快吗？足够安全吗？足够安全吗？ ä 价格实惠吗？当用户需要时它会准备好了吗？是否可以随着需求的变化而改变？

### → 质量不是孤立地衡量软件的标准

ä 它衡量软件与其应用领域之间的关系

Ø 除非您将软件放入其环境中，否则无法测量这一点... Ø ...并且不同环境中的质量会有所不同！

ä 在设计过程中，我们需要预测软件将如何满足其目的

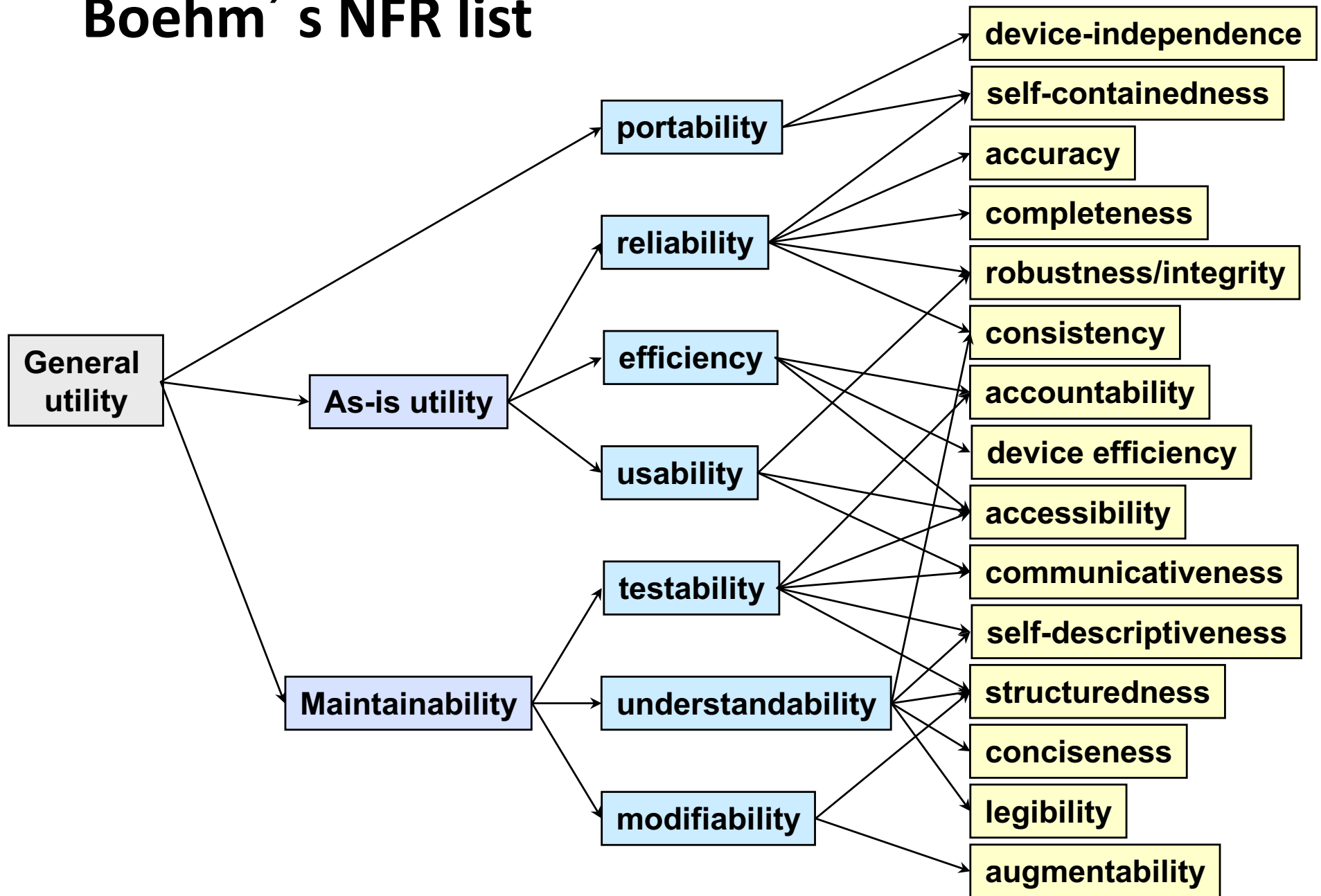
Ø 我们需要高质量的预测器（设计分析）

Ø 在需求分析过程中，我们需要了解如何衡量目的的适用性 Ø 预期目的是什么？

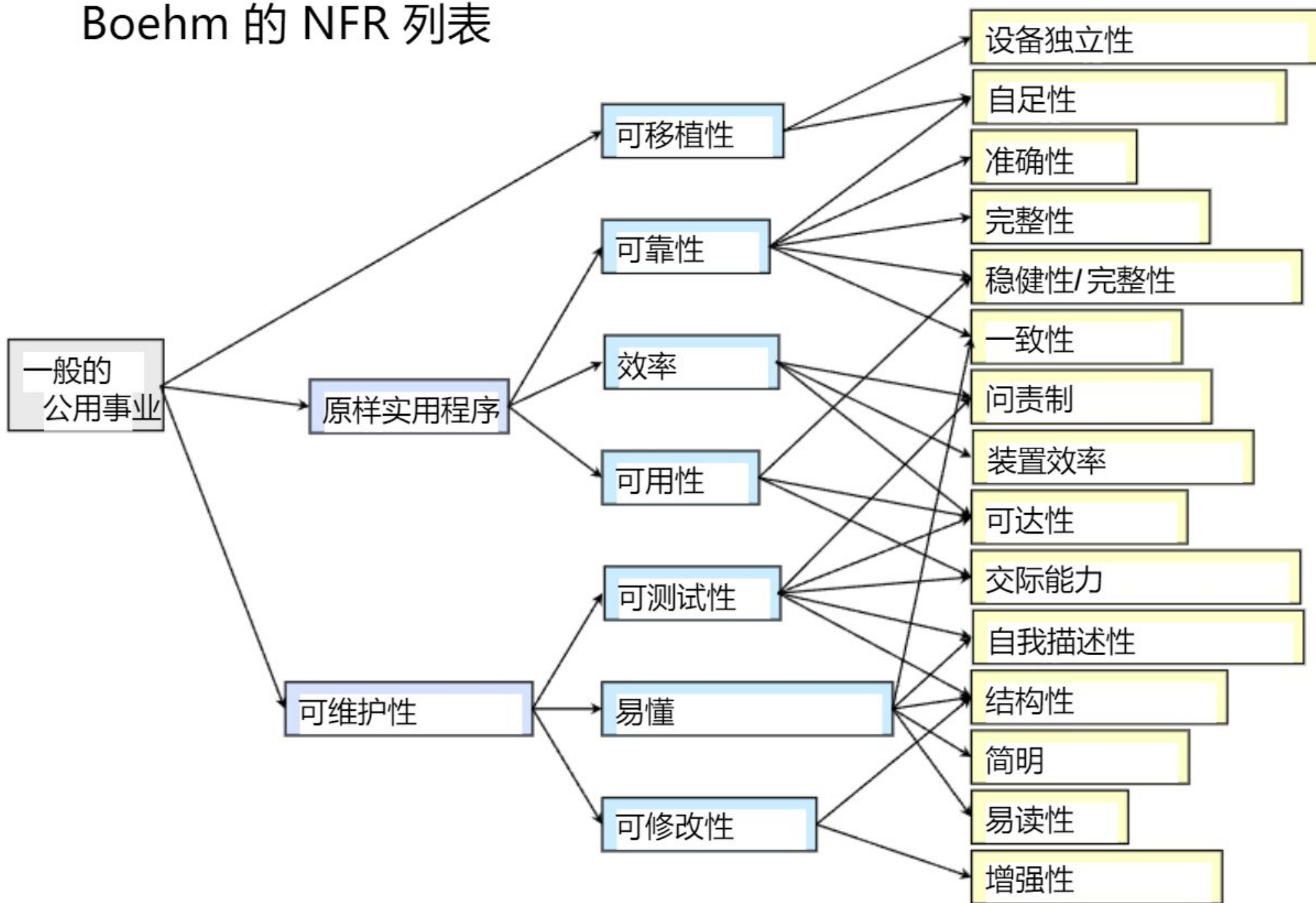
Ø 哪些质量因素对利益相关者来说很重要？ Ø 这些因素应该如何运作？



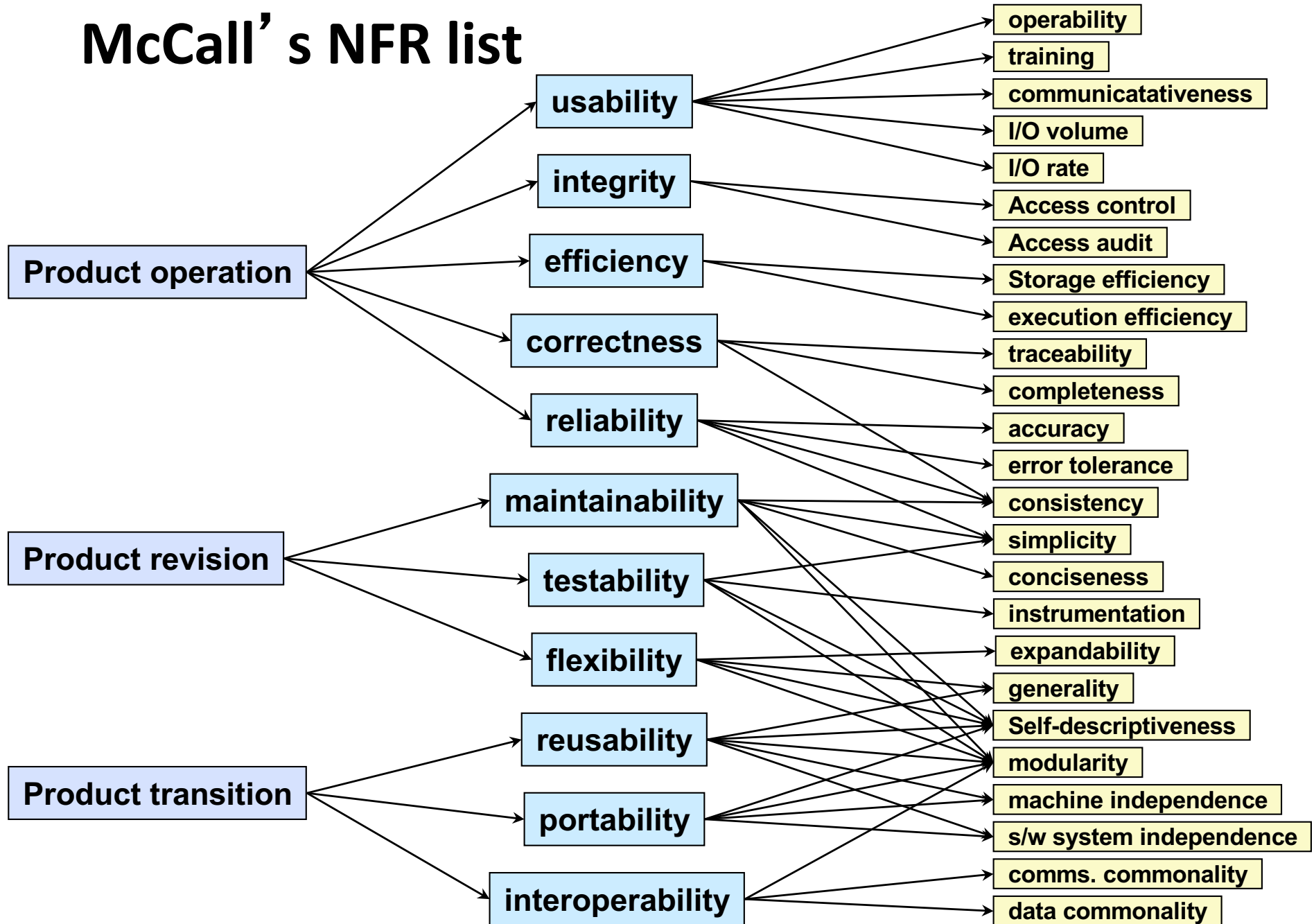
# Boehm's NFR list



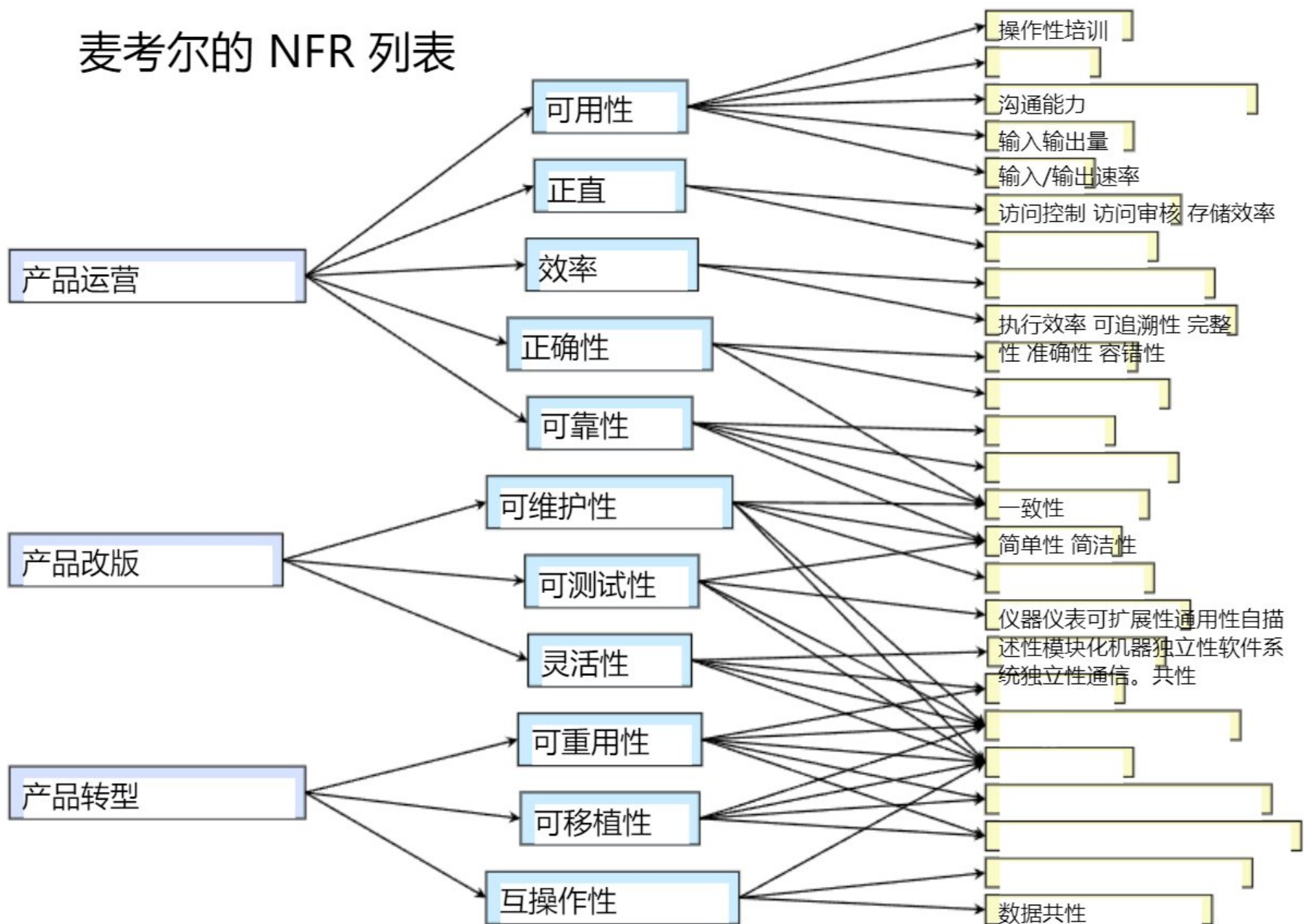
## Boehm 的 NFR 列表



# McCall's NFR list

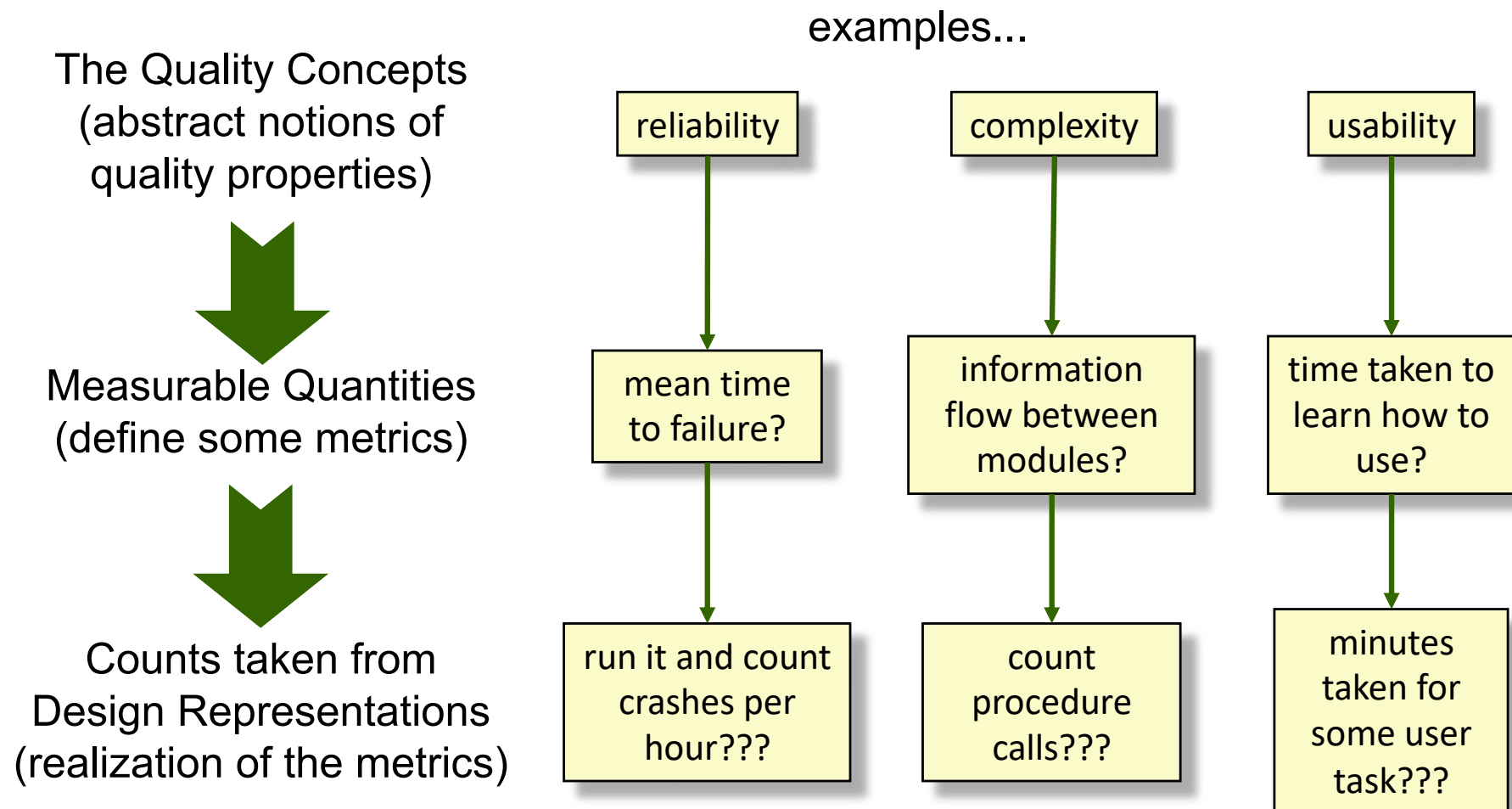


# 麦考尔的 NFR 列表



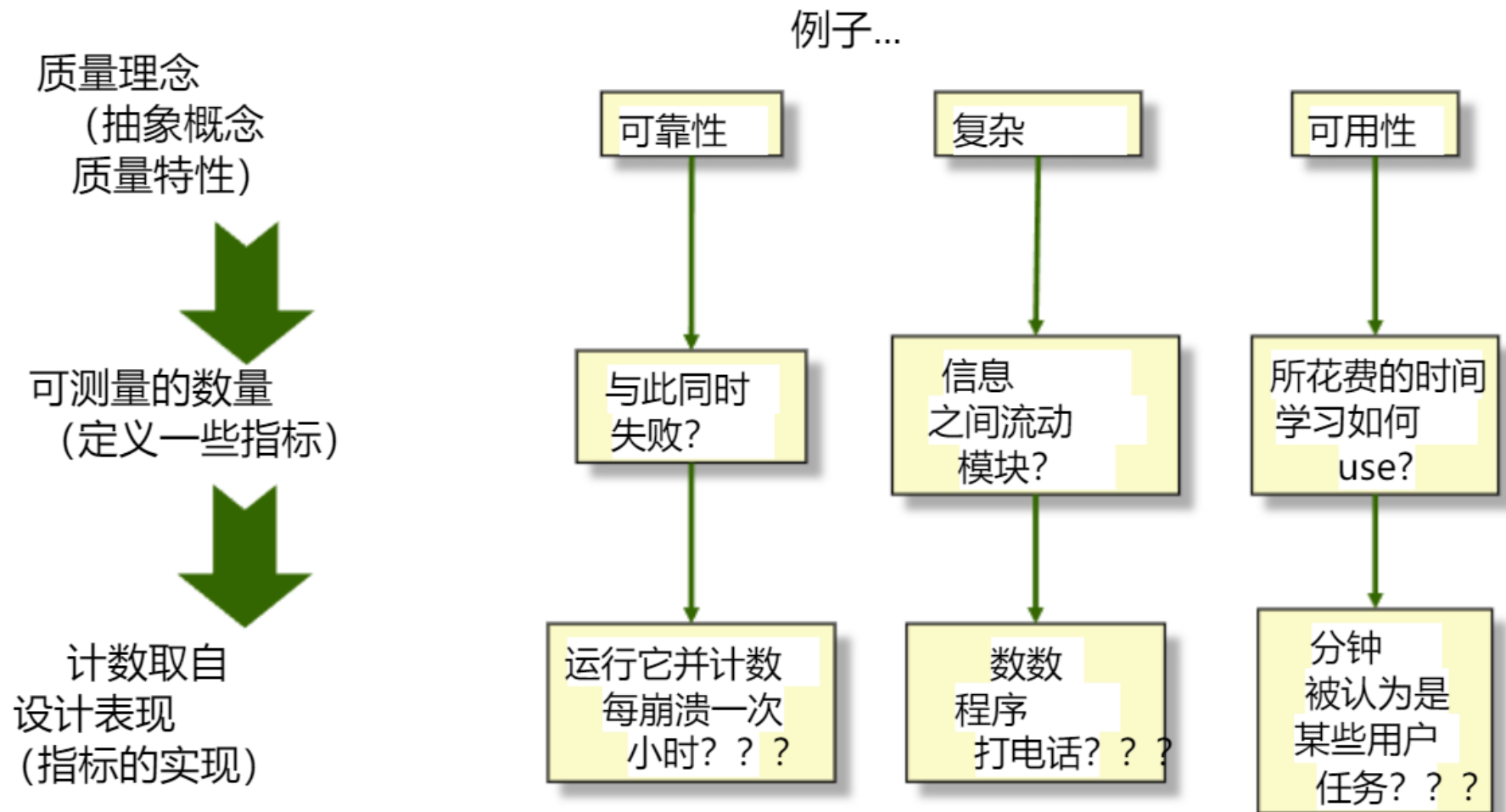
# Making Requirements Measurable

→ We have to turn our vague ideas about quality into measurables



## 使需求可衡量

→ 我们必须将关于质量的模糊想法转变为可衡量的指标



# Example Metrics

Quality	Metric
Speed	transactions/sec response time screen refresh time
Size	Kbytes number of RAM chips
Ease of Use	training time number of help frames
Reliability	mean-time-to-failure, probability of unavailability rate of failure, availability
Robustness	time to restart after failure percentage of events causing failure
Portability	percentage of target-dependent statements number of target systems



## 指标示例

质量	公制
速度	交易/秒 响应时间 屏幕刷新时间
Size	千字节 RAM芯片数量
使用方便	训练时间 帮助帧的数量
可靠性	平均无故障时间, 不可用概率 故障率、可用性
鲁棒性	失败后重新启动的时间 导致失败的事件 百分比
可移植性	目标相关语句的百分比 目标系统的数量

# Making Requirements Measurable

## → Define ‘fit criteria’ for each requirement

- ↳ Give the ‘fit criteria’ alongside the requirement

- ↳ E.g. for new ATM software

  - Requirement: “The software shall be intuitive and self-explanatory”

  - Fit Criteria: “95% of existing bank customers shall be able to withdraw money and deposit cheques within two minutes of encountering the product for the first time”

## → Choosing good fit criteria

- ↳ Stakeholders are rarely this specific

- ↳ The right criteria might not be obvious:

  - Things that are easy to measure aren’t necessarily what the stakeholders want

  - Standard metrics aren’t necessary what stakeholders want

- ↳ Stakeholders need to construct their own mappings from requirements to fit criteria

## 使需求可衡量

### → 为每个要求定义“适合标准”

给出“适合标准”以及要求新的 ATM 软件

Ø 要求：“软件应该直观、不言自明” Ø 适合标准：“95%的现有银行客户在第一次接触该产品后两分钟内能够取款和存入支票”

### → 选择合适的标准

利益相关者很少如此具体正确的标准可能并不明显：

Ø 容易衡量的东西不一定是利益相关者想要的 Ø 标准指标不一定是利益相关者想要的

利益相关者需要构建自己的从需求到符合标准的映射

# Usability requirements

Usability = Fit for use + Ease of use

## → Ease of learning

↳ How easy is the system to learn for various groups of users?

## → Task efficiency

↳ How efficient is it for the frequent user?

## → Ease of remembering

↳ How easy is to remember for the occasional user?

## → Subjective satisfaction

↳ How satisfied is the user with the system?

## → Understandability

↳ How easy is it to understand what the system does?

Satisfaction		
Response time (sec)	Rating	Quality score
< 2	Exceeds expectation	3
2-5	Within the target range	2
6-10	Minimally acceptable	1
>10	Unacceptable	0

# 可用性要求

可用性 = 适合使用 + 易于使用

## → 易于学习

系统对于不同用户群体来说是否容易学习？

## → 任务效率

Ä 它的效率如何？  
经常使用的用户？

## → 易于记忆

Ø 记忆有多容易  
对于偶尔的用户？

## → 主观满意度





ä 用户对系统的满意度如何？

## → 易懂

ä 理解系统功能的难易程度如何？

满意		
回复时间 (秒)	评分	质量分数
< 2	超出预期	3
2-5	在目标范围内范围	2
6-10	最低限度可以接受	1
> 10	不可接受	0

# Usability requirements

- Q1: It should be easy for novice users to do tasks Q and R.
  - Q2: Novice users should perform tasks Q and R in a short time.
  - Q3: Experienced users complete tasks Q, R, and S quicker than novice users
  - Q4: Recording breakfast shall be easy using keyboard
- Problem counts  

  - Task time  
  

  - Keystroke counts  


## 可用性要求

- Q1: 应该很容易  
新手用户做任务Q  
和 R。
- Q2: 新手用户应该  
在 a 中执行任务 Q 和 R  
短时间。
- Q3: 有经验的用户  
完成任务 Q、R 和 S  
比新手用户更快
- Q4: 记录早餐应  
轻松使用键盘
- 问题很重要
  - ä Q1: 20 个新手中至多 1 个在任务  
Q 和 R 中会遇到严重问题。
- 任务时间
  - ä Q2: 新手用户应在 15 分钟内完成  
任务 Q 和 R。 ä Q3: 有经验的用户  
在 2 分钟内完成任务 Q、R 和 S
- 击键次数
  - ä Q4: 每位客人按 5 次按键即可记录  
早餐



# Usability requirements

- Q1: It should be easy for novice users to do tasks Q and R.
  - Q2: Novice users should perform tasks Q and R in a short time.
  - Q3: Experienced users complete tasks Q, R, and S quicker than novice users
  - Q4: Recording breakfast shall be easy using keyboard
- Problem counts
    - ↳ Q1: At most 1 of 20 novices shall encounter critical problems during tasks Q and R.
  - Task time
    - ↳ Q2: Novice users shall perform tasks Q and R in 15 minutes.
    - ↳ Q3: Experienced users complete tasks Q, R and S in 2 minutes
  - Keystroke counts
    - ↳ Q4: Recording breakfast shall be possible within 5 keystrokes per guest

## 可用性要求

- Q1: 应该很容易  
新手用户做任务Q  
和 R。
- Q2: 新手用户应该  
在 a 中执行任务 Q 和 R  
短时间。
- Q3: 有经验的用户  
完成任务 Q、R 和 S  
比新手用户更快
- Q4: 记录早餐应  
轻松使用键盘
- 问题很重要
  - ä Q1: 20 个新手中至多 1 个在任务  
Q 和 R 中会遇到严重问题。
- 任务时间
  - ä Q2: 新手用户应在 15 分钟内完成  
任务 Q 和 R。 ä Q3: 有经验的用户  
在 2 分钟内完成任务 Q、R 和 S
- 击键次数
  - ä Q4: 每位客人按 5 次按键即可记录  
早餐

# Efficiency

- **Efficiency** - the capability of the software to provide the required performance relative to the amount of resources used, under stated conditions

# 效率

- 效率 - 在规定的条件下，软件相对于所使用的资源量提供所需性能的能力

# Efficiency

- **Efficiency** - the capability of the software to provide the required performance relative to the amount of resources used, under stated conditions

Which statement is objectively defined?

1: Product shall be able to process a lot of payment transactions in a short time even during peak load

2: Product shall be able to process 100 payment transactions per second in peak load.

# 效率

- 效率 - 在规定的条件下，软件相对于所使用的资源量提供所需性能的能力

哪种说法是客观定义的？

- 1：即使在高峰负载期间，产品也应能够在短时间内处理大量支付交易
- 2：产品应能够在峰值负载下每秒处理100 笔支付交易。

# Efficiency

- **Efficiency** - the capability of the software to provide the required performance relative to the amount of resources used, under stated conditions

Which statement is objectively defined?

1: Scrolling one page up or down in a 200 page document shall take at most 1s. Searching for a specific keyword shall take at most 5s.

2: Scrolling one page up or down in a large document shall take an efficient time. Searching for a specific keyword shall a reasonable time

# 效率

- 效率 - 在规定的条件下，软件相对于所使用的资源量提供所需性能的能力

哪种说法是客观定义的？

1: 200页文档中向上或向下滚动一页最多需要1秒。搜索特定关键词最多需要5秒。

2: 在大文档中向上或向下滚动一页需要高效的时间。搜索特定关键词应有合理的时间



# Efficiency

- **Efficiency** - the capability of the software to provide the required performance relative to the amount of resources used, under stated conditions

Which statement is objectively defined?

1: A simple report shall take a short waiting time. None shall take the long waiting time

2: None shall take longer than the simple report preparation in the majority of the cases.

# 效率

- 效率 - 在规定的条件下，软件相对于所使用的资源量提供所需性能的能力

哪种说法是客观定义的？

1：简单的报告等待时间短。没有人会花费漫长的等待时间

2：在大多数情况下，任何时间都不会比简单的报告准备时间更长。

# Maintainability

## → Maintenance performance

- ↳ Q1: Supplier's hotline shall analyse almost all reports in a short period
- ↳ Q2: When repairing a defect, a number of related non-repaired defects should be very low

## → Development process

- ↳ Q3: Every program module must be assessed for maintainability according to organisation's standards OST-1.12.x. Majority of the modules have to be "High maintainable" (as defined in the standard) and none "poor" (as defined in the standard)
- ↳ Q4: Development must use regression test allowing full re-testing in a short period

## → Program complexity

- ↳ Q5: No method in any object may contain a lot of code lines

## 可维护性

### → 维护性能

ä Q1: 供应商热线应在短时间内分析几乎所有报告 ä Q2: 修复缺陷时, 相关未修复缺陷的数量应该很低

### → 开发流程

ä Q3: 每个程序模块都必须根据组织的标准 OST-1.12.x 进行可维护性评估。大多数模块必须是“高可维护”（如标准中定义的）并且没有“差”（如标准中定义的）。  
Q4: 开发必须使用回归测试, 允许在短时间内进行全面的重新测试

### → 程序复杂度

ä Q5: 任何对象中的方法都不能包含大量代码行

# Maintainability

## → Maintenance performance

↳ Q1: Supplier's hotline shall analyse 95% of reports within 2 work hours.

↳ Q2: When repairing a defect, related non-repaired defects shall be less than 0.5 coverage

## → Development process

↳ Q3: Every program module must be assessed for maintainability according to organisation's standards OST-1.12.x. 70% of modules must obtain "High maintainable" (as defined in the standard) and none "poor" (as defined in the standard).

↳ Q4: Development must use regression test allowing full re-testing in 12 hours.

## → Program complexity

↳ Q5: No method in any object may exceed 200 lines of code

## 可维护性

### → 维护性能

ä Q1: 供应商热线应在2个工作日内分析95%的报告。 ä Q2: 修复缺陷时, 相关未修复缺陷应小于0.5覆盖率

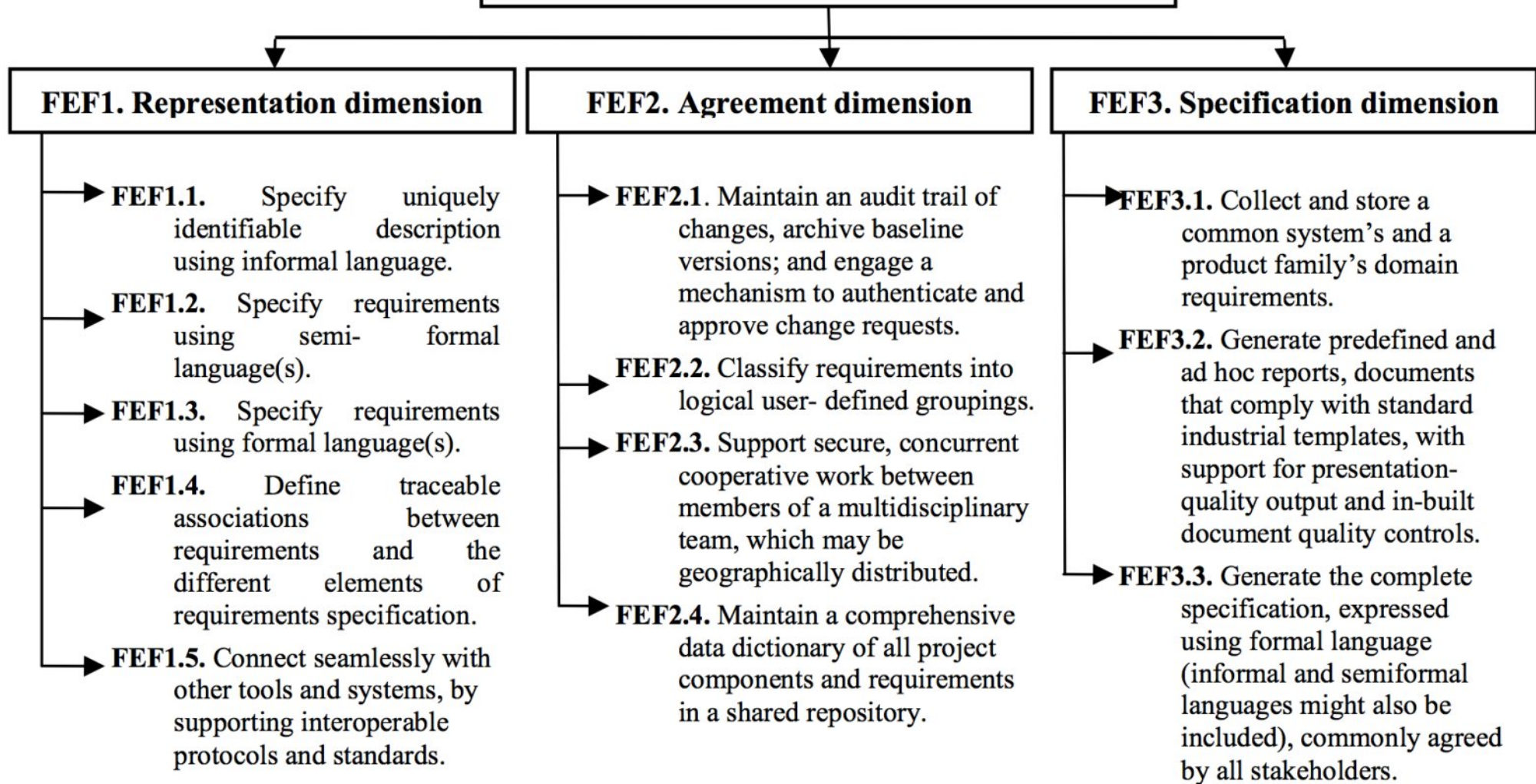
### → 开发流程

ä Q3: 每个程序模块都必须根据组织的标准 OST-1.12.x 进行可维护性评估。 70% 的模块必须获得“高可维护性” (如标准中定义), 并且没有“差” (如标准中定义)。 ä Q4: 开发必须使用回归测试, 允许在 12 小时内进行全面重新测试。

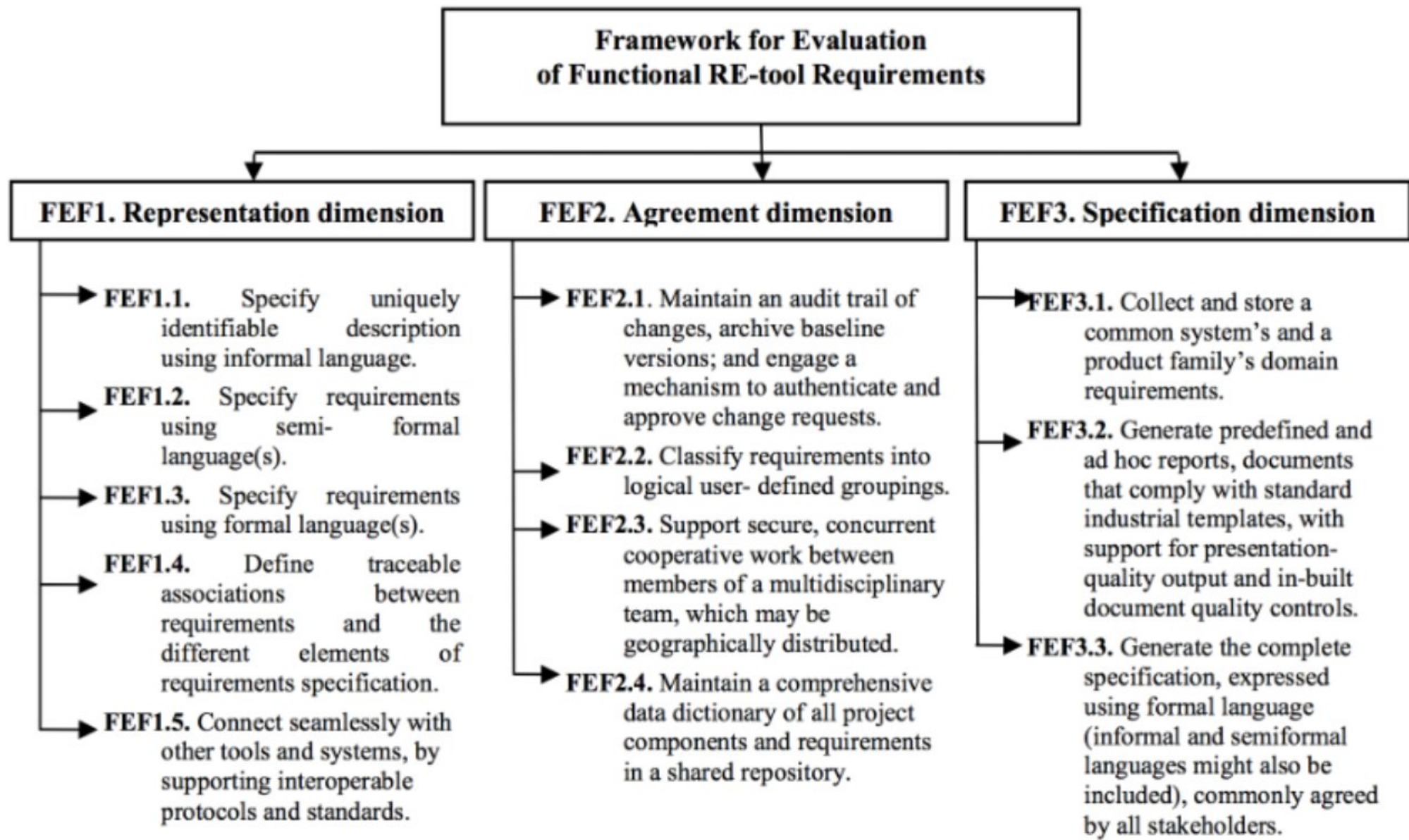
### → 程序复杂度

ä Q5: 任何对象中的方法都不能超过 200 行代码

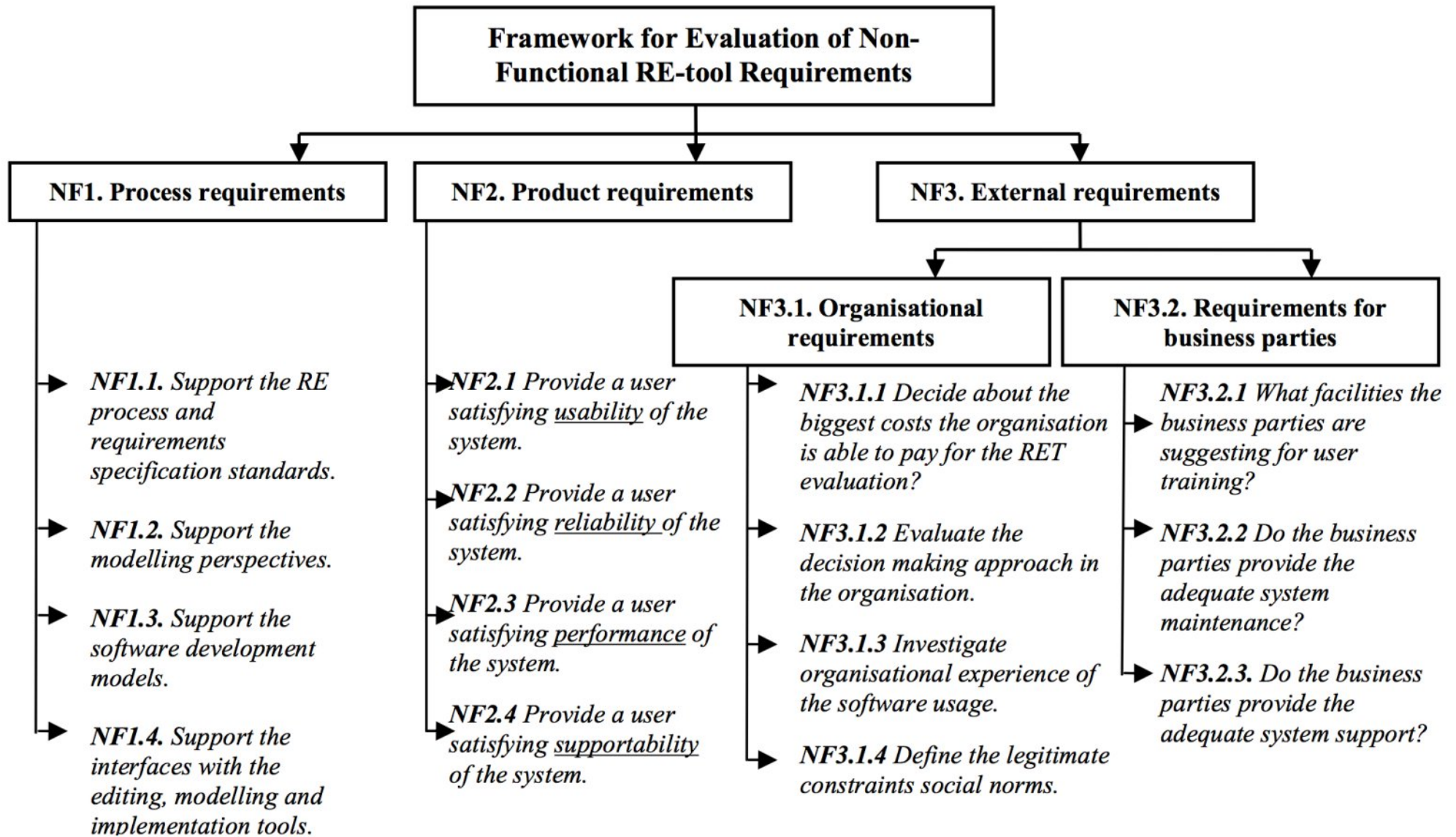
## Framework for Evaluation of Functional RE-tool Requirements

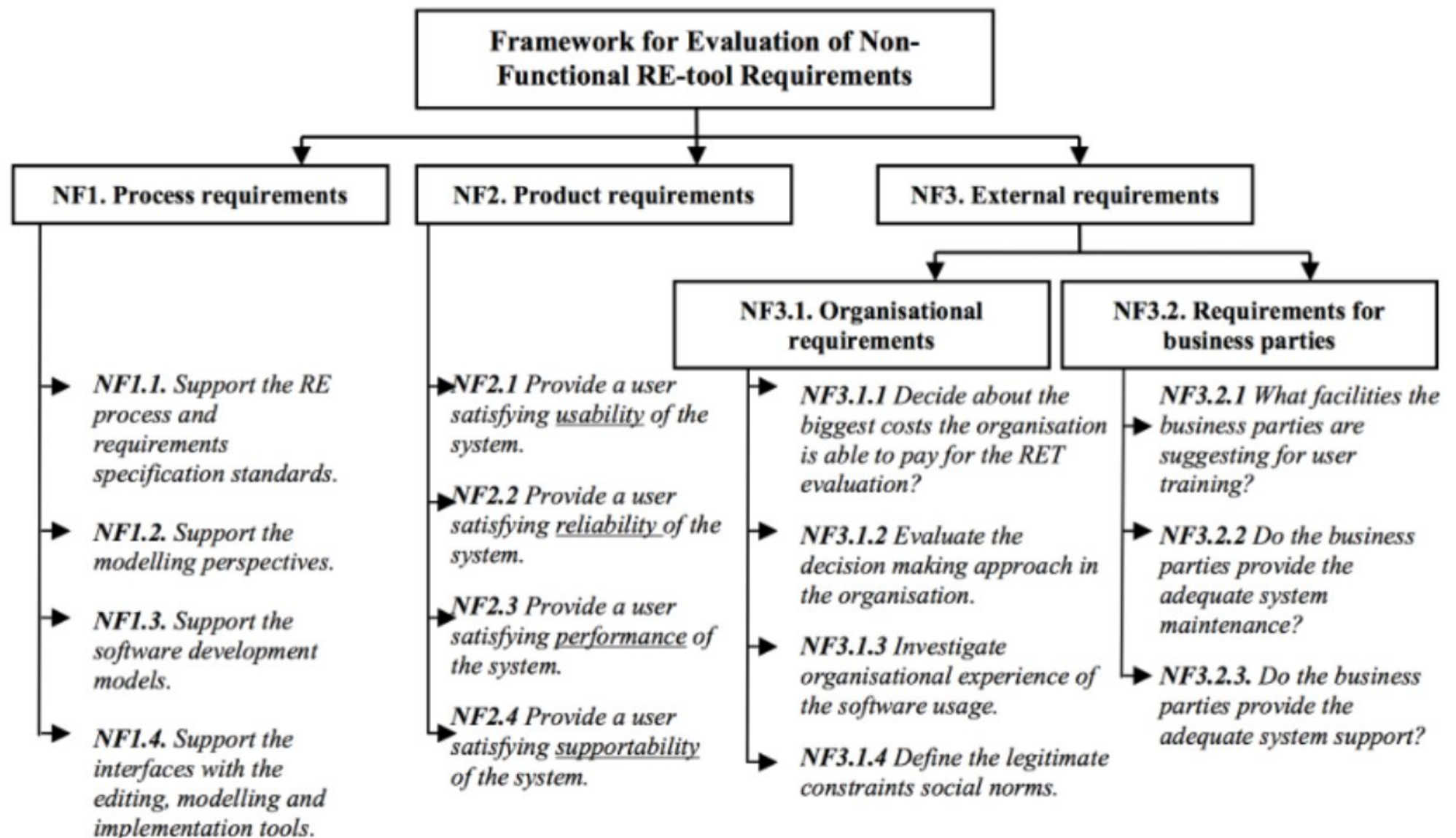


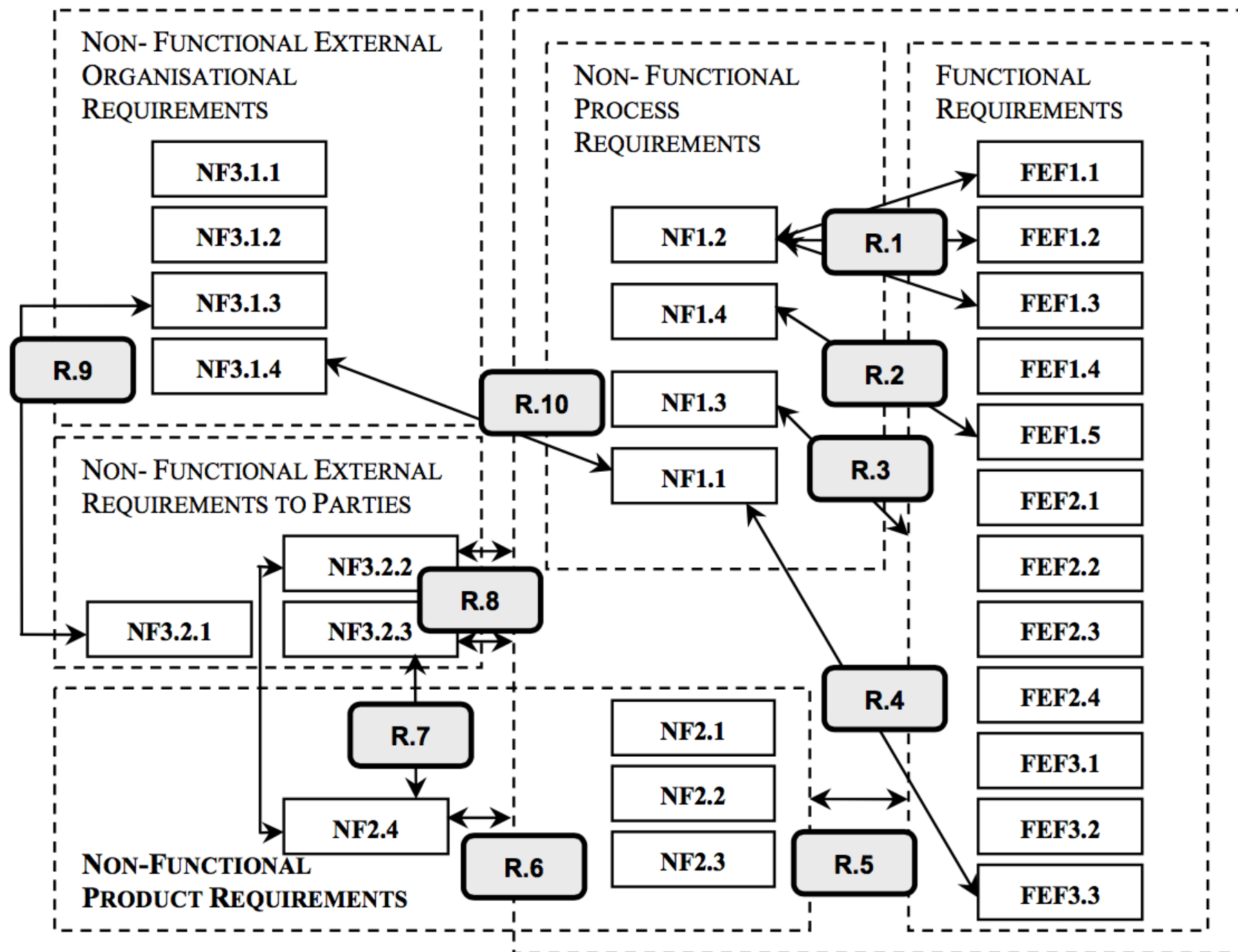




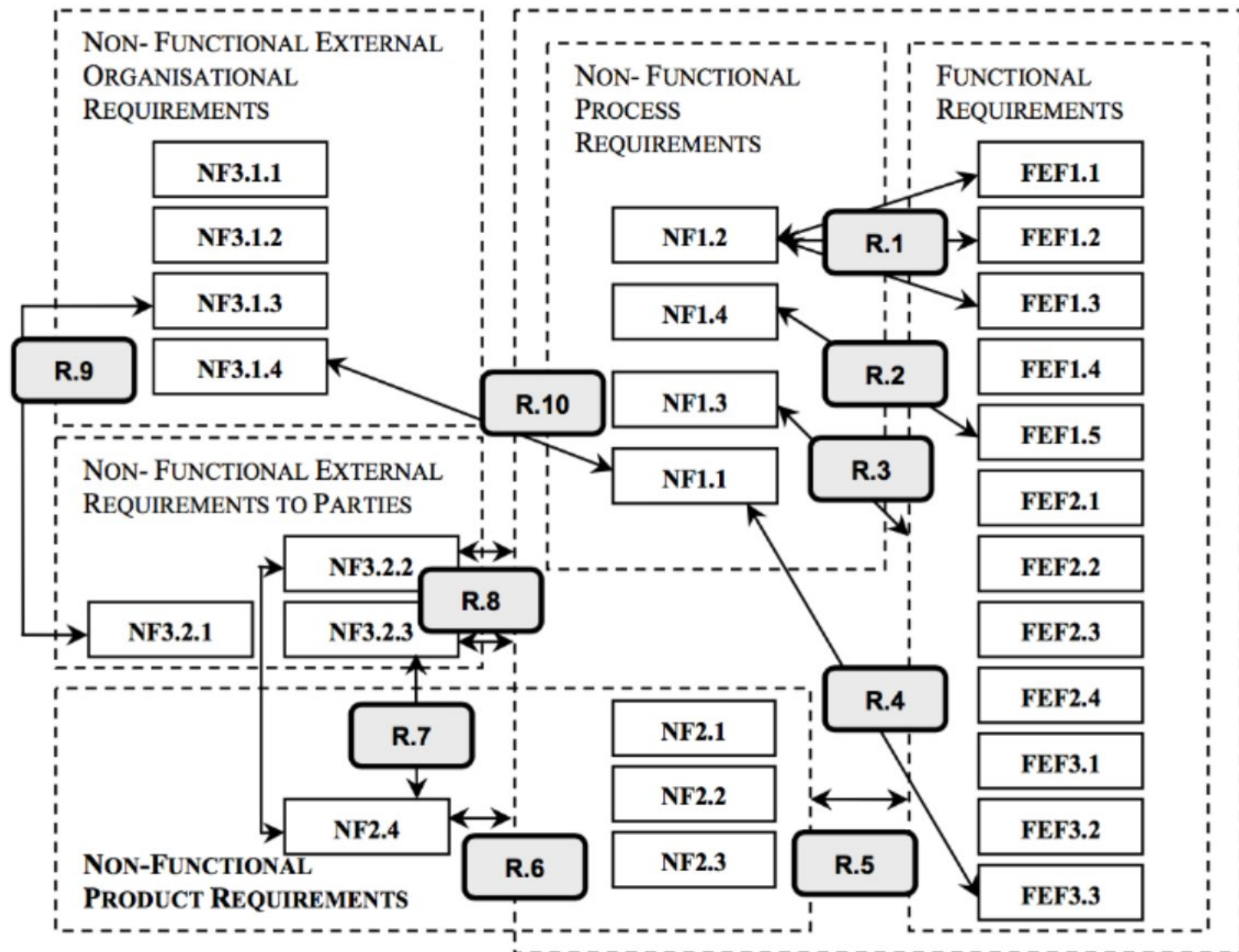












NON- FUNCTIONAL EXTERNAL  
ORGANISATIONAL

NON- FUNCTIONAL

FUNCTIONAL

**R.1.** FEF1.1, FEF1.2, FEF1.3 and NF1.2 – relationship between the modelling perspective and the languages supported by the RE-tool.

**R.2.** NF1.4 and FEF1.5 – relationship between the tools already used and the RE-tool.

**R.3.** NF1.3 and RE-tool functional features consider the software development model, used by user.

**R.4.** NF1.1 and FEF3.3 – relationship between the RE process and specification standards.

**R.5.** Non-functional RE-tool features (NF2.1, NF2.2, and NF2.3) and functional RE-tool features - relationship between the RE-tool usability, efficiency, reliability and functionality.

**R.6.** NF2.4 and the functionality consider the need to maintain the non- functional RE-tool features against the functional characteristics by the customer.

**R.7.** NF3.2.2, NF3.2.3 and NF2.4 – relationship of maintenance and support between the customer and the business parties.

**R.8.** NF3.2.2, NF3.2.3 and the functionality – relationship of the RE-tool maintenance and support by the business parties.

**R.9.** NF3.1.3 and NF3.2.1 – relationship between the customer's knowledge and the need for training.

**R.10.** NF1.1 and NF3.1.4 – relationship between the standards and the social, organisational, law and cultural factors.

NON- FUNCTIONAL  
PRODUCT REQUIREMENTS

R.6

NF2.3

R.5

FEF3.3

NON- FUNCTIONAL EXTERNAL  
ORGANISATIONAL

NON- FUNCTIONAL

FUNCTIONAL

R.1。 FEF1.1、 FEF1.2、 FEF1.3 和 NF1.2 – 建模视角与 RE 工具支持的语言之间的关系。

R.2。 NF1.4 和 FEF1.5 – 已使用的工具和 RE 工具之间的关系。 R.3。 NF1.3和RE-tool功能特性考虑了用户使用的软件开发模型。

R.4。 NF1.1 和 FEF3.3 – RE 过程和规范标准之间的关系。 R.5。非功能性 RE 工具功能 (NF2.1、NF2.2 和 NF2.3) 和功能性 RE 工具功能 - RE 工具可用性、效率、可靠性和功能之间的关系。

R.6。 NF2.4 和功能考虑了根据客户的功能特性维护非功能性 RE 工具特性的需要。 R.7。 NF3.2.2、 NF3.2.3 和 NF2.4 – 客户和业务方之间的维护和支持关系。 R.8。 NF3.2.2、 NF3.2.3 和功能——业务方维护和支持 RE 工具的关系。 R.9。 NF3.1.3 和 NF3.2.1 – 客户知识与培训需求之间的关系。 R.10。 NF1.1 和 NF3.1.4 – 标准与社会、组织、法律和文化因素之间的关系。

NON-FUNCTIONAL  
PRODUCT REQUIREMENTS

R.6

NF2.3

R.5

FEF3.3

# Non-Functional Requirements (NFRs)

- What are non-functional requirements
- Product-oriented qualities
- Process-oriented qualities
- Traceability between requirements

## 非功能性需求 (NFR)

- ！ 什么是非功能性需求
- ！ 以产品为导向的品质
- ！ 以过程为导向的品质
- ！ 需求之间的可追溯性