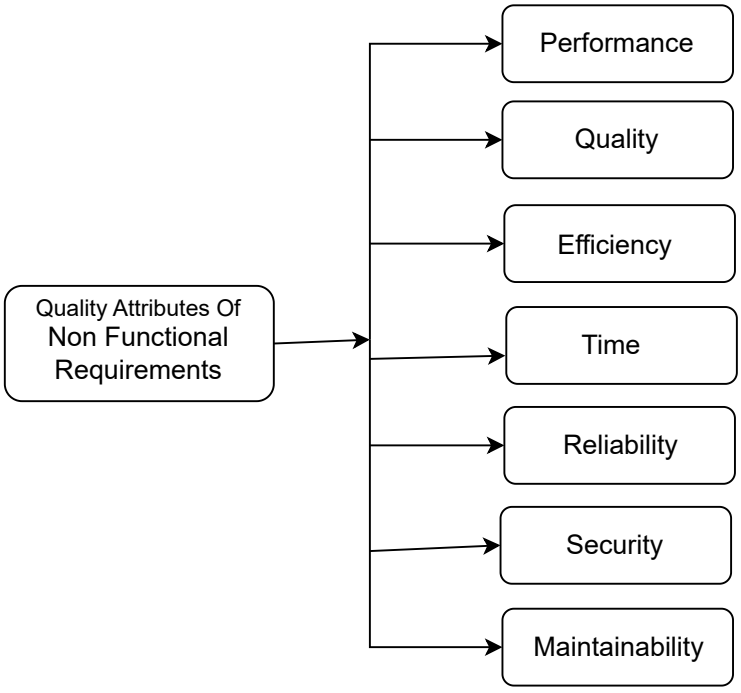
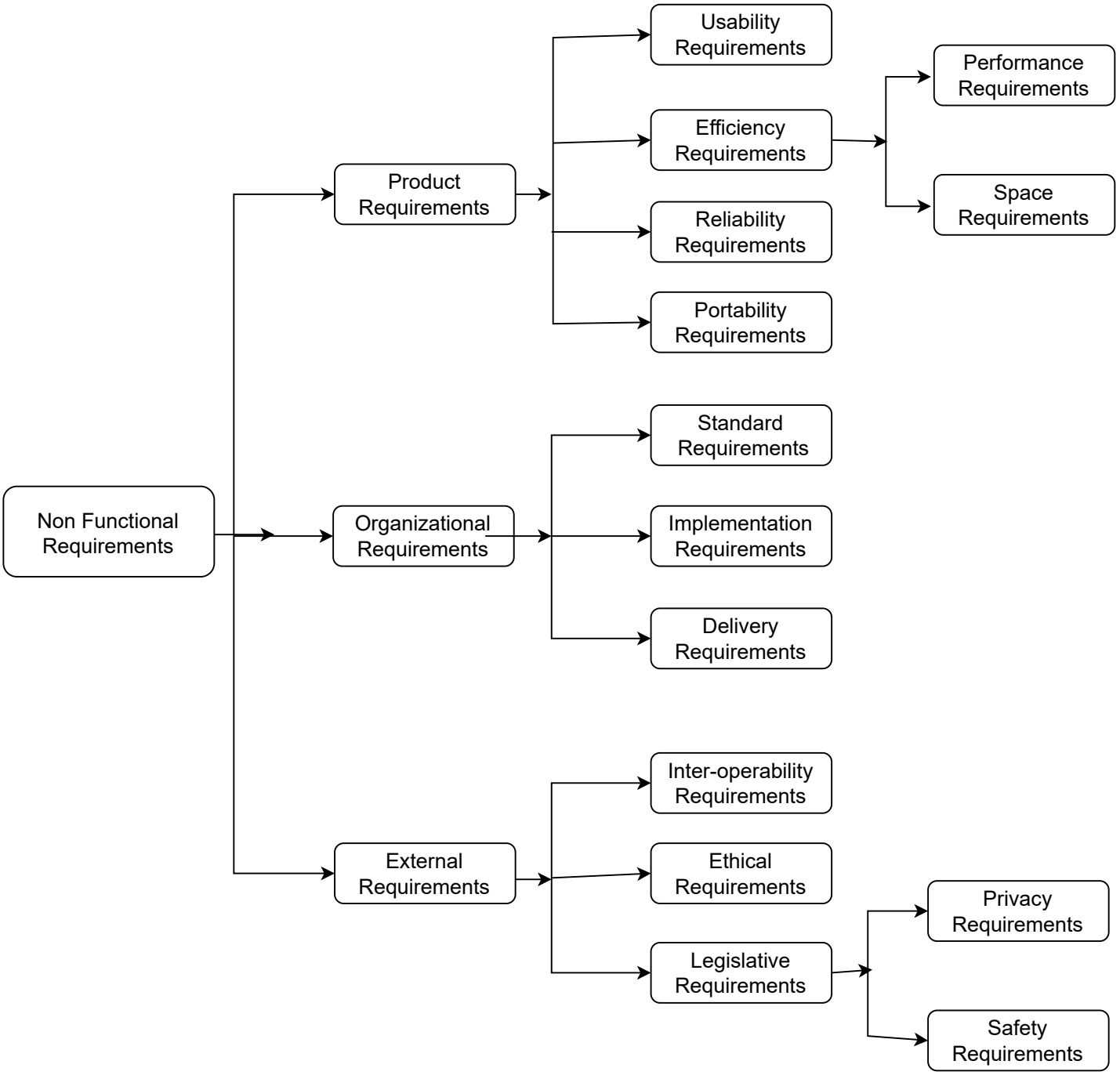
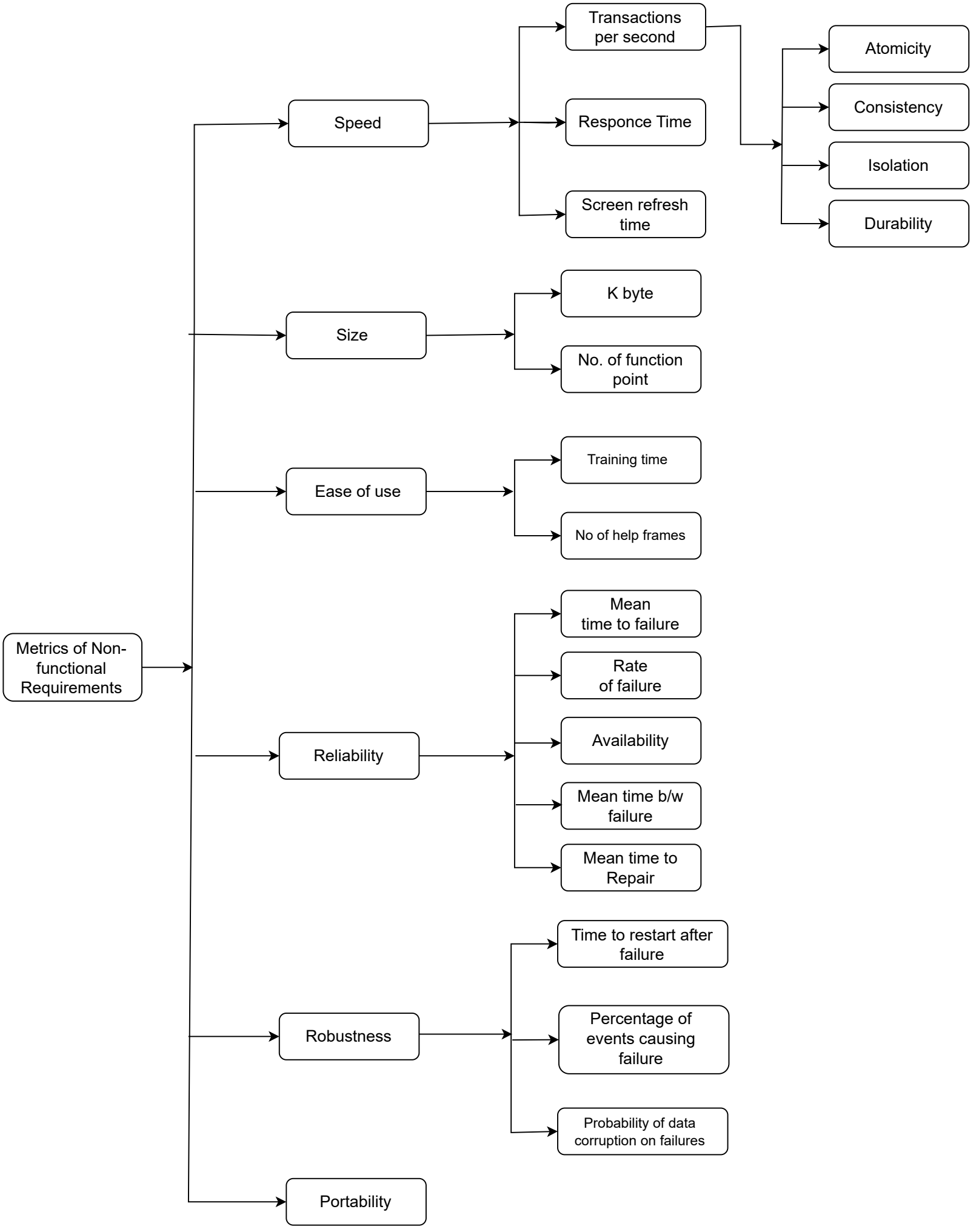


Legislative Requirements
Security: The software shouldn't have ill effects on data / hardware. Proper measures should be taken to keep data secure from external threats. **Safety:** The software should not be hazardous to the environment/life.

Product Requirements
Usability: Time required to learn how to use the software should be less. **Efficiency:** This characteristic relates to the way software uses the available resources. The software should make effective use of the storage space and execute command as per desired timing requirements. **Reliability:** The software product should not have any defects. Not only this, it shouldn't fail while execution. Or **Reliability** is how system wil operate without failure for a defined duration.

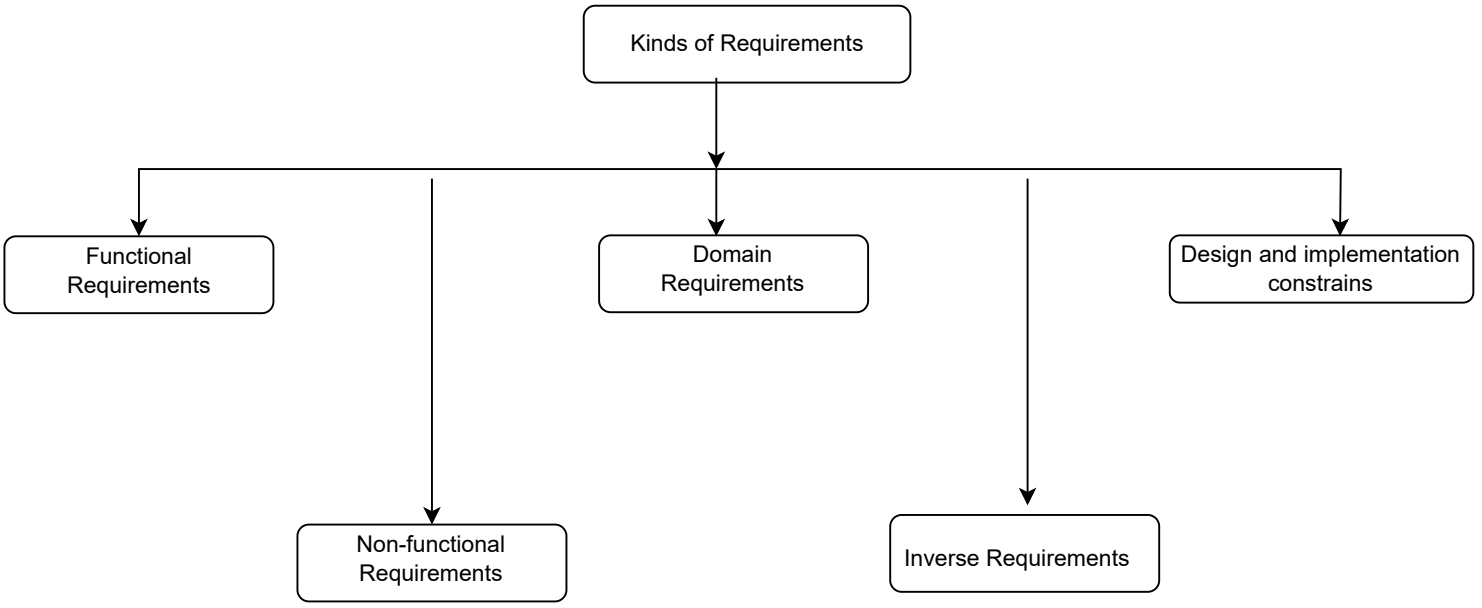

Non Functional Requirements
Product Requirements
Organizational Requirements
External Requirements
Usability Requirements
Efficiency Requirements
Reliability Requirements
Portability Requirements
Performance Requirements
Space Requirements
Standard Requirements
Implementation Requirements
Delivery Requirements
Inter-operability Requirements
Ethical Requirements
Legislative Requirements
Privacy Requirements
Safety Requirements


Metrics of Non-functional Requirements
Speed
Transactions per second
Responce Time
Screen refresh time
Atomicity
Consistency
Isolation
Durability
Size
K byte
No. of function point
Ease of use
Training time
No of help frames
Reliability
Mean time to failure
Rate of failure
Availability
Mean time b/w failure
Mean time to Repair
Robustness
Time to restart after failure
Percentage of events causing failure
Probability of data corruption on failures
Portability

Transactions
Atomicity: **Consistency:** The system ensures that the payment is consistent with the user's account balance and updates the payment records accurately. **Isolation:** The transaction is isolated from other transactions, ensuring that it operates independently. **Durability:** Once the payment is confirmed, the changes are permanent and will not be lost even in the event of a system failure.
Responce Time
Responce Time: After heavy calculation how long to responce.
Screen Refresh Time
Screen Refresh Time: computer games, graphic oriented use this more

Size
K byte: to measure size in kilo bytes of executable source or source code **No. of function point:** Count of function points, which can then be used for estimating project effort, cost and other metrices.
Ease of use
Training time: to use a system **No. of function point:** how much help build in the product
Reliability
Mean time to failure: average life time of product before it fails. formula = Total Opeating Time / No. of Failures **Mean time to repair:** average time it takes to repair a failed system. formula = Total DownTime / No. of Repairs **Mean time b/w failure:** formula = Total opeating time / (No. of Failures-1) **Rate of failure:** Rate at which failure occur in system. formula = No of Failures / Total Opeating Time **Availability:** Probability of product for being unavailable. The remaing time is availability time. Availability depends on performance, speed and responce time

Robustness (Cope up with faults that occurred)
Time to restart after failure: how long a product take to response after failure has occurred **Percentage of events causing failure:** minimize the percentage to 0 of failure occurrence. **Probability of data corruption on failures:**
Types of faults
Hardware faults(disk fail, device timeout) **Software faults**(bugs, error, defects) **User faults**(entering data in different format)



Kinds of Requirements
Domain Requirements: Requirements that come from the application domain eg. Banking domain has its own specific constraints, for example, most banks do not allow over-draw on most accounts, however, most banks allow some accounts to be overdrawn **Inverse Requirements:** These requirements indicate the indecisive nature of customers.e.g.The system shall not use red color in the user interface, whenever it is asking for inputs from the end-user