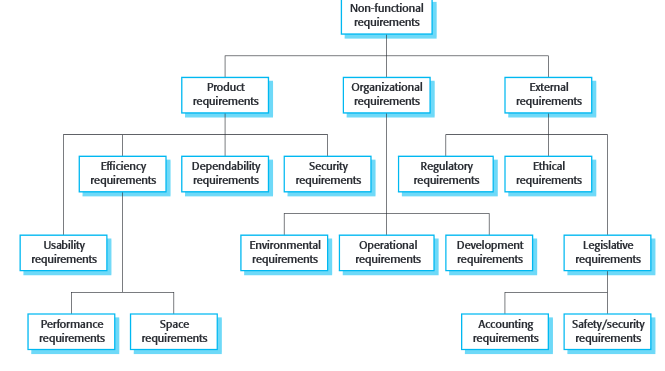
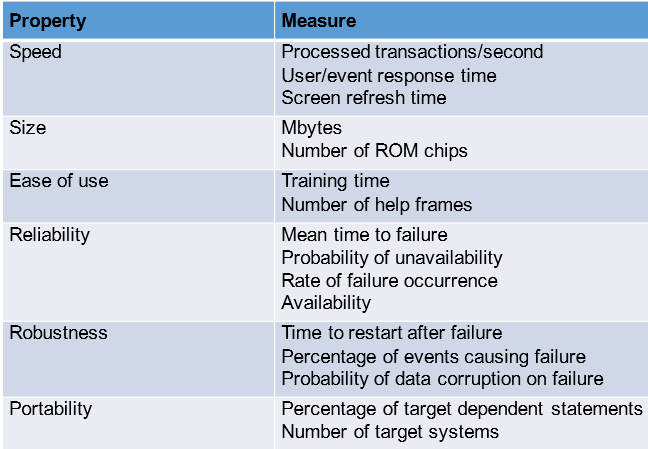
Nonfunctional requirements  
•Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.  
•Often apply to the system as a whole rather than individual features or services



This chart + the following

Speed  
-not too much of an issue, but shouldn’t be able to freeze

-Should not be too slow either , to allow other customers to use the machine without waiting too long.

Size  
-Not an issue either as long as the hardware has enough memory space to accommodate it.

Ease of use  
-organize the UI

-Must be simple and easy to use for people with or without any technical experience.

Reliability  
-Should not fail during a transaction.

Robustness  
-Do not allow for data corruption. Save nothing unless confirmed proper operation.

Portability  
-not required past the single identical line of ATMs

Functional requirements  
•Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.  
•May state what the system should not

Card reading  
-system needs to be able to take card info and find a matching account  
-the provided pin must match the found account  
-no more than 5 attempts at entering correct pin before lockout  
 -display a get assistance message

UI  
-a welcome screen   
-options:  
 -see balance  
 -withdraw funds  
 -request for assistance (if available, should be on all screens)

-deposit funds

-option to print receipt

-option to transfer funds  
 -play music (if we need more)  
 -give a complement (‘cause innovation)  
 -quit/sign off  
-welcome screen needs a space for bank promos  
-each screen needs a back button

Data  
-confirm that cash was dispensed before changing funds when withdrawing  
-be able to use exception handling in case something goes wrong  
 -in which case don’t change the balance or dispense money  
-reduce the users balance by an additional 1% of their withdrawn amount

-verify that there are enough funds available into the machine to satisfy the customer’s needs.(not sure if this belongs in data)

Security  
-protect against system cracking  
-do not allow for any kind of withdrawing without a card, sufficient balance, or unauthorized accounts  
-in the case of insufficient funds, tell this to the user and prompt for a new amount. Don’t withdraw  
 -they can input a new amount or quit  
-in some way implement a protection for forced withdraws by muggers. Add a discrete security button that notifies authorities and security (it should do it without displaying any confirmation)

Records  
-save a copy of each transaction’s information (receipts) to a file showing amount withdrawn, remaining balance, and data and time along with a transaction number  
-receipts printed to the customer should say something nice

Environment, constraints, etc  
•Constraints on the system from the domain of

-phone connection for alert button  
-staff availability for assistance/hours of operation  
-write the code in Java, develop using Git