

Basic questions that apply to everything (in no particular order):

Who	What	Where	When	How	Why
-----	------	-------	------	-----	------------

Time based aspects:

- Is this the first time (new) or a repeat (existing)?
- How likely is reuse: how soon, how often, by whom?

Structure	Behaviour
-----------	-----------

Performance Influence Matrix: factors relevant to performance

	Users	Transactions (Handled by a package)	Data-Flow (between packages)
Frequency (peaks, etc)			
Volumes (totals, etc)			

Throughput	Transactions per unit of time.
Performance	Elapsed time per transaction.
Latency	Wait time for a response.
Capacity	Number of users / entities the system can support for a given configuration at a fixed level of performance.
Scalability	Ability to increase capacity.
Reliability	Length of time a system can operate without failure.
Response Time	Performance as perceived by a human.

Security:

- Protection of data, both at rest and in-flight
- Protection of resources (misuse of services / functionality)
- Protection of systems (DOS, Out-of-band attacks, etc)
- Boundary defence / encryption
- Authentication
- Authorization
- How will Authentication & Authorization be enforced, be managed?

Logical Layers (not including hardware)

- User Interface
- Application Layer (Service Layer or Controller Layer)
- Domain Layer (Business Layer, Business logic Layer or Model Layer)
- Infrastructure Layer (data access, logging, network access, security, email, file system, and so on)

Lifecycle

- Design, Prototyping
- Development / Refactoring
- Testing (suitability, integration, performance, ...)
- Deployment / Use

Management Matrix: Who owns and controls what?

	Ownership	Control
The process (that the system implements)		
The hardware		
Platform (OS)		
The system(s) (software / services)		
The data		

Business / Project Management

- What are the Critical Success Factors
- Is there a common terminology across all involved parties

Information Architecture

- Organization / Structure
- Labeling
- Navigation, Browsing
- Searching

Information Lifecycle

- Search, Create, Read, Update, Delete

Transactions

- Atomic, Consistent, Isolated, Durable

Architecture -> Capabilities -> Features

- **Execution qualities:** such as security and usability, are observable at run time.
- **Evolution qualities:** such as testability, maintainability, extensibility and scalability, are embodied in the static structure of the software system.

Accuracy: Indicates proximity to the true value (how close are you).

Precision: The repeatability of the measurement (consistency within certain bounds).

Modifiability: Define the effort, time & cost required to make changes in the software. Often, the measurement is personnel effort (person - months). Related to Compatibility & Dependencies (the degree of impact on other systems, etc).

Portability: The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Robustness: the quality of being able to withstand stresses, pressures, or changes in procedure or circumstance.

Performance:

- See "Performance Influence Matrix"
- Expected transaction & response times (average, worst case)
 - response times
 - transaction rates
 - throughput

Availability: Service Level Agreements:

- Percentage of time available and/or hours of availability
- Expected recovery time

Compatibility: Backwards compatibility. Dependencies - do they change between versions?

Extensibility: New capabilities can be added to the software without major changes to the underlying architecture.

Modularity: Well defined components, separation of concerns.

Information Architecture:

- Organization / Structure
- Labeling
- Navigation
- Search

Operating constraints:

- System resources (storage, memory)
- People (hours of business, location)
- Software (dependencies, minimum requirements)

Deployment

- Physical location(s)
- Political location (e.g. hosted internally or externally)
- Ease of access (maintenance) vs. Security
- Dependencies
- Communication between locations/servers/etc (how, management matrix, etc)

Maintainability: the ease with which a software product can be modified in order to:

- correct defects
- meet new requirements
- make future maintenance easier, or
- cope with a changed environment

Levels of Granularity

- Systems
 - Components
 - Classes

Some possible Views:

- Logical (typically coarse grained view)
- Functional (fine grained logical view)
- Deployment / Physical
- Business Process (and specific roles within that)
- Specific scenarios (both business and technical)
- Class / Module / Package / Component / Service
- Concurrency / processes / threading
- The User(s)
- Various stakeholders
- Data
- Security / Attack Surface
- Public / private, internal / external

Others to consider:

- Alignment with current and future technology stacks, industry trends
- Support: strength of community (and its location)
- User Interfaces
- Backend Interfaces
- Context
- Responsibilities
- Patterns
- Platform(s)
- Appropriate Granularity
- Supportability
- Maintainability (evolvment – extension – refactoring)
- Dependencies (on other parties, systems, services, standards)
- Interoperability adherence to standards, or: is achieved when the coherent, electronic exchange of information and services between systems takes place.
- Portability
- Resilience / Robustness: the quality of being able to withstand stresses, pressures, or changes in procedure or circumstance.
- Resource constraints (processor speed, memory, disk space, network bandwidth etc.)
- Scalability (horizontal, vertical / out, up)
- Security
- Usability by target user community