REQUIREMENT SPECIFICATION

PART ONE: SOFTWARE REQUIREMENT SPECIFICATIONS (SRS) (CHs 10 AND 11)

SRS AS REQUIREMENT SPECIFICATION DOCUMENT

- SRS describes the expected outcomes from analysis stage
  - Functions – what the software can deliver given inputs (?)
  - Capabilities – what the software can do and cannot do
  - Features – what the software looks different from others (Distinctive behaviours? What about other behaviours?)
  - Constraints – what rules/regulations the software must follow (business rules and IT industry standards)
  - System behaviours under various conditions (behaviours other than features)
  - Qualities – containing internal and external qualities
- SRS itself is also the output of specification stage.
- It is the basis for subsequential project planning, design, and coding, as well as the foundation for system testing and user documentation
- Creating SRS is a process of specifying requirements.

This image shows the capability of problem-solving which is supported by four functionalities – questioning, searching, creating and implementing.

SRS AUDIENCES

- Customers (users) – they need to know what product they can expect to be delivered from SRS
- Project managers – they plan/monitor/amend timeline, effort, and resources based on requirements given in SRS
- Software development teams – they need to know what to build as specified in SRS.
- Testers – they use SRS to develop requirements-based tests, test plans, and test procedures.
- Maintenance and support staff – they use SRS to understand what each part of the product is supposed to do and in relation to other parts/systems.
- Documentation writers – they develop user manuals and help screens as stated in the SRS, in particular, the features given by user interface.
- Training personnel – they use the SRS and user documentation to develop training tutorials.
- Legal staff – they need to ensure that the requirements comply with applicable laws and regulations.
- Subcontractors – they are contracted based on the specified requirements

SRS STRUCTURE AND CONTENTS

1. Introduction section

An overview to help the reader understand how the SRS is organized and how to use it.

- Purpose
  - the purpose of the project, i.e. product or application to be developed in the project to which the requirements are associated ("to develop the product of ….")
  - the targeted audiences of the document (specify as mentioned in the previous slide)
- Document conventions – standards/conventions in terms of format used in the document (e.g. IOS/IEC/IEEE)
- Scope – what the project will achieve and what are not its concerns
- References – resources the document refers to

SRS STRUCTURE AND CONTENTS

2. Overall description section

A high-level overview (general, no details is needed) of the product and the environment in which the product will be used, the anticipated users, and known constraints, assumptions, and dependencies

- Product perspective -- the product's context and origin (why it is needed)
- User classes and characteristics – the targeted users and their characteristics (who are going to use it and in which ways they are going to use)
- Operating environment -- including the hardware platform; operating systems and versions; geographical locations of users, servers, and databases; and organisations that host the related databases, servers, and websites (hardly to have a stand-alone system now)
- Design and implementation constraints -- any factors that will restrict the options available to the developers and the rationale for each constraint, such as the limitations of programming language and library to be used, etc. (**technical constraints** rather than business rules)
- Assumptions – functionalities related (e.g. all mobile phones have touch-screen)
- dependencies – external factors that are not controllable, e.g. a certain version of .NET

SRS STRUCTURE AND CONTENTS

3. System features section

Set up sub-headings for each features (required by customers) and describe the features one after another in the corresponding sub-headings

- 3.x System feature x (sub-heading)

  3.x.1. Description

  - What feature x is
  - The priority level in terms of development of feature x

  3.x.2. Functional requirements (raised by customers)

  - Itemised functional requirements associated with feature x, i.e. what functions and capabilities the software should have to realise this feature

(The above should be repeatedly used for all features.)

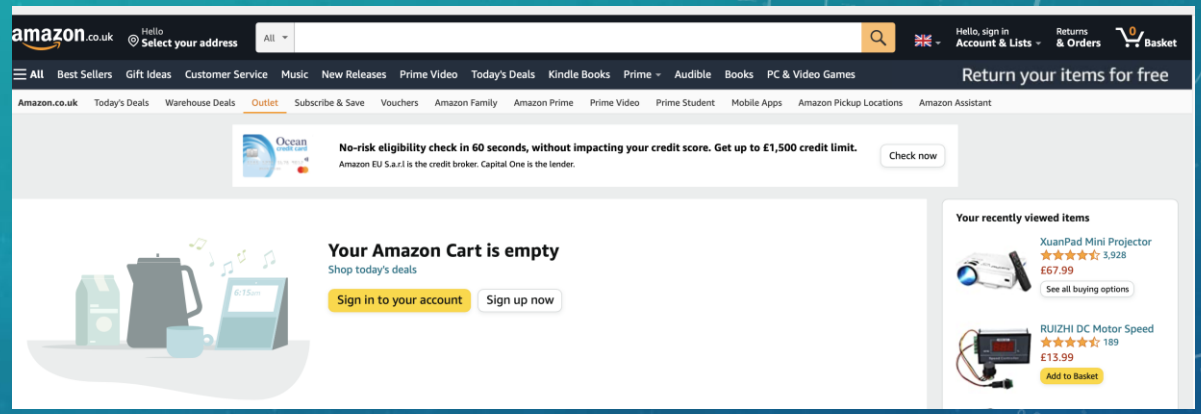
Note, we are talking about requirements here but not system specifications! Therefore, the features and functions are those that customers required.

SRS STRUCTURE AND CONTENTS

Exercise 1: Shopping Cart?

What are the features of a Shopping Cart?

What are functions that support the features?



Exercise 2: Smart Shopping with smart scanner

Imagine and name two features?

Imagine and name functions that support the features?

SRS STRUCTURE AND CONTENTS

4. Data requirements section (more details in Part Two)

Defining the types of data the software will be used as inputs, as intermedium variables and as the outputs

- Logical data model such as entity-relationship diagrams (types of data for each column) and UML class diagrams (date types)
- Data dictionary – defines the composition of data structures and the meaning, data type, length, format, and allowed values for the data elements
- Reports – management reports generated from IT systems, their format, size, language, tables, diagrams, etc
- Data acquisition (all about data protection), including consents (purpose, life-cycle, who to use), integrity (keep quality when transferred), retention (keep in a safe place), and disposal processes (destroy)

SRS STRUCTURE AND CONTENTS

5. External interface requirements section

Specifying how the system communicate the user as well as external software and hardware

6. Non-functional requirements section (more in Part Three)

- Performance – such as speed of response, throughput, execution time and storage capacity
- Security – information security is a big issue now
- Accessibility – for people who have disabilities

7. Internationalisation and localisation section

Internationalisation and localisation requirements ensure that the product will be suitable for use in nations, cultures, and geographic locations other than those in which it was created. (Imagine if JD has had Chinese-language version only, it would not be internationalised.)

8. Other requirements section

- Legal, regulatory, or financial compliance and standards requirements
- Product installation, configuration, start-up, and shutdown; and logging, monitoring, and audit trail requirements.

SRS VALIDATING

The collected individual requirements need to be validated according to the following criteria:

- Complete – Containing all detailed information (e.g. fond – style, size, colour, effects)
- Correct – not saying the wrong things
- Feasible – achievable given capabilities and limitations of the system and its operating environment, as well as time, budget, human resources, etc.
- Necessary – leading to the anticipated business value, differentiating to the product in the marketplace, or being required for conformance to an external standard, policy, or regulation.
- Prioritised – ranking business requirements according to which are most important to achieving the desired value.
- Unambiguous – not misleading or causing misunderstanding (“He is cooking apple”)
- Verifiable – one can check whether the requirement is achieved at the end

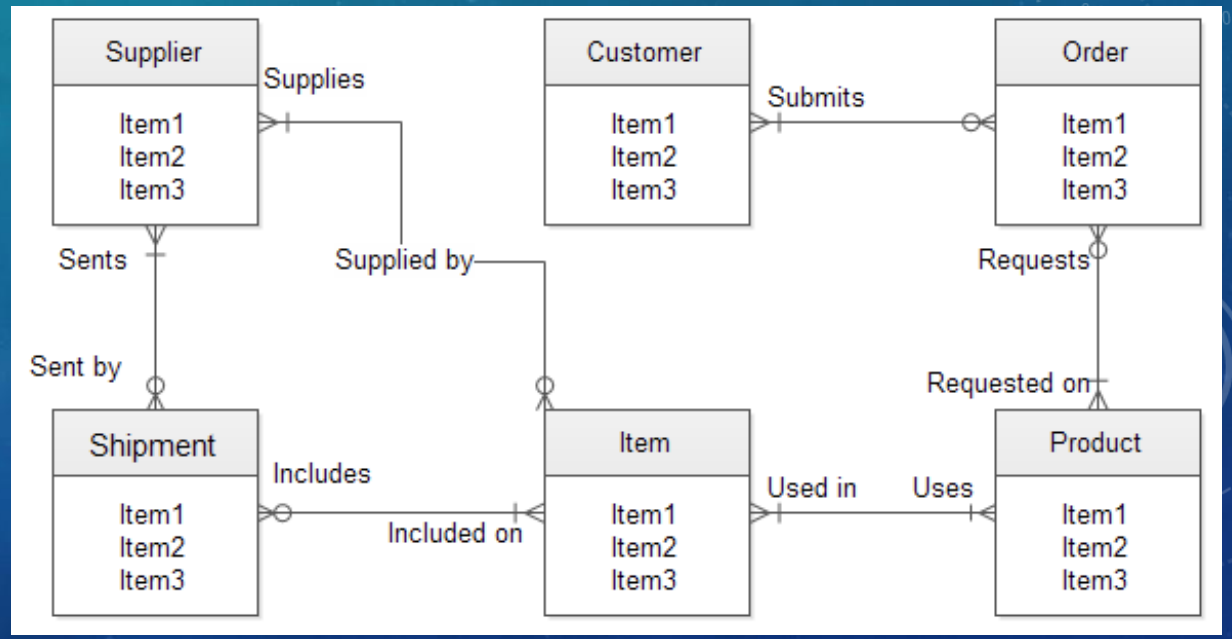
PART TWO: DATA (CH13)

DATA MODELLING

Data specifications are about what data will be manipulated in a software system. It is important as software functionality is all about data – create, read, update and delete data

ER model

- An analysis process aiming to specify data entities themselves and the relationships among them
  - Entity relationship diagram as a common data model (ER model)
  - Entities (data items) – data aggregations or physical items – attributes – features of data items
  - Relationships/associations – logical linkage between data item pair
  - Multiplicity – the number of entity type instances that can be at one end of an association



DATA DICTIONARY

- Date dictionary is a metadata repository – information about data such as meaning, relationships to other data, origin, usage, and format, or collection of tables about metadata.
- Example:

DATA

emlployee_id	first_name	last_name	nin	dept_id
44	Simon	Martinez	HH 45 09 73 D	1
45	Thomas	Goldstein	SA 75 35 42 B	2
46	Eugene	Comelsen	NE 22 63 82	2
47	Andrew	Petculescu	XY 29 87 61 A	1
48	Ruth	Stadick	MA 12 89 36 A	15
49	Barry	Scardelis	AT 20 73 18	2
50	Sidney	Hunter	HW 12 94 21 C	6
51	Jeffrey	Evans	LX 13 26 39 B	6
52	Doris	Berndt	YA 49 88 11 A	3
53	Diane	Eaton	BE 08 74 68 A	1

DATA DICTIONARY (METADATA)

Column	Data Type	Description
emlployee_id	int	Primary key of a table
first_name	nvarchar(50)	Employee first name
last_name	nvarchar(50)	Employee last name
nin	nvarchar(15)	National Identification Number
position	nvarchar(50)	Current postion title, e.g. Secretary
dept_id	int	Employee deparmtnet. Ref: Departmetns
gender	char(1)	M = Male, F = Female, Null = unknown
employment_start_date	date	Start date of employment in organization.
employment_end_date	date	Employment end date.

Datædo

CRUD MATRIX

- CRUD stands for Create, Read, Update, and Delete.
- A CRUD matrix correlates system actions with data entities to show where and how each significant data entity is created, read, updated, and deleted.
- Types of correlations, including the following:
  - Data entities and system events
  - Data entities and user tasks or use cases
  - Object classes and use cases
- CRUD has a lot of more than just the matrix but for requirement specification, the matrix is the focus.
- Example (Data entities and user tasks):

CRUD MATRIX

Calling Item	Item Type	COUNTER	CUSTOMER	EMPLOYEE	PRODUCT	SALES_ORDER_ITEMS	SALES_ORDER
ALL_SALES_ORDER_ITEMS_DETAILS	View			R	R	R	R
ALL_PRODUCTS	View				R		
ALL_ORDERS_BY_EMPLOYEE	View						R
EMPLOYEE_COUNT	Trigger			R			
TRG_ORDER	Trigger				R		
CUSTOMERS.InsertCustomer	Procedure		C				
CUSTOMERS.UpdateCustomerName	Procedure		U				
CUSTOMERS.DeleteCustomerById	Procedure		D				
EMPLOYEES.OrdersByEmployee	Procedure		R	R		R	R
PRODUCTS.ProductsByCustomer	Procedure		R		R	R	R
PRODUCTSBYCUSTOMER	Procedure		R		R	R	R
pfc_updateprep	Event	RU					
itemchanged	Event				R		
demopfc.pbl.d_tab_customer	Datawindow Object		RU				
demopfc.pbl.d_tab_sales_order	Datawindow Object		R	R	R	R	R
demopfc.pbl.d_ff_sales_order	Datawindow Object		R	R			RU
demopfc.pbl.d_tab_employee	Datawindow Object			RU			

Table or Column Accessed

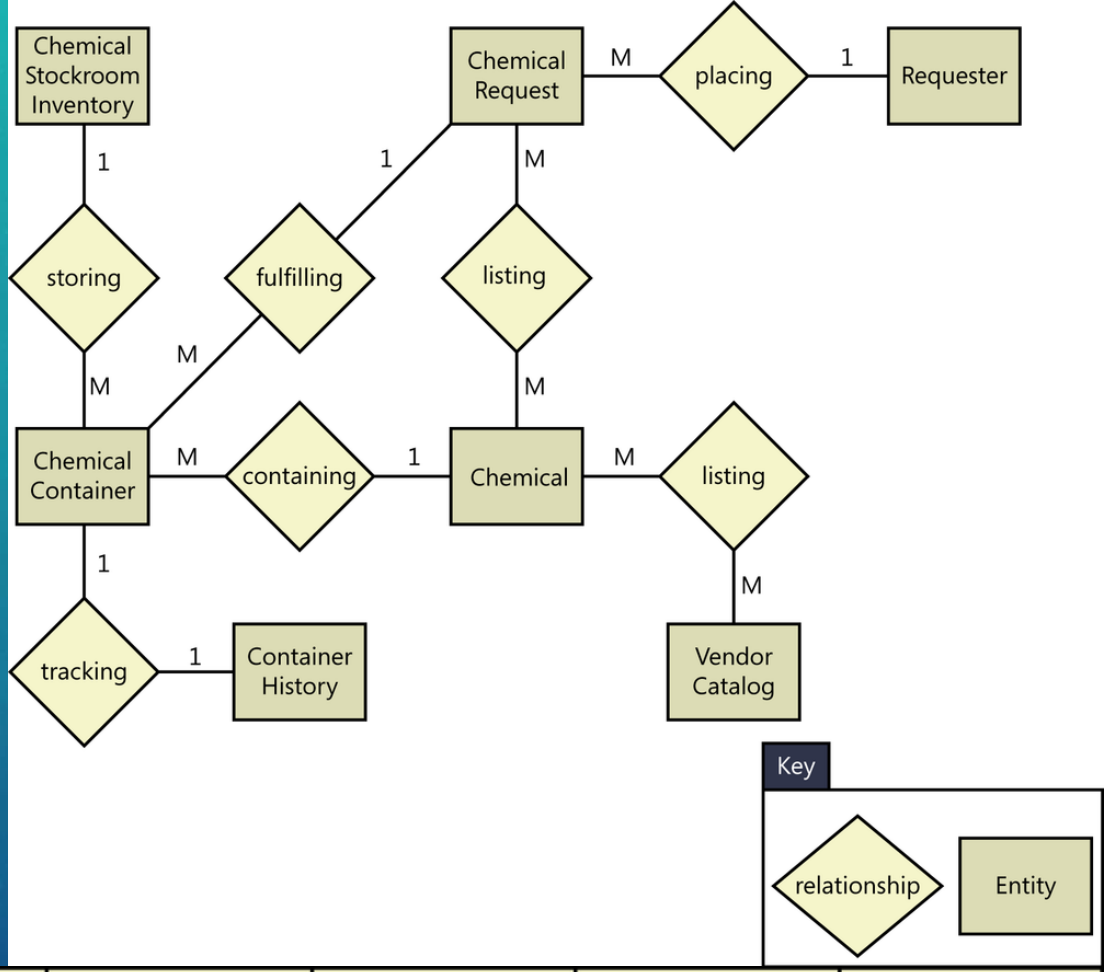
Components Accessing the table/column

Type of Access (Create, Read, Update, Delete)

DATA SPECIFICATION

Case study – Chemical tracking system

- "ER" model
- CRUD matrix, e.g. The first row means: Place order use case: an order is C'ed, data about chemical, requester and supplier's catalogue are R'ed
- CRUD also tells the methods each use case should have, e.g. The third row means: Chemical inventory has methods of C's, U's and D's chemical data in the DB. If data is not manual manipulated, a class of Inventory can be defined.



Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

REPORT

IT systems generate reports

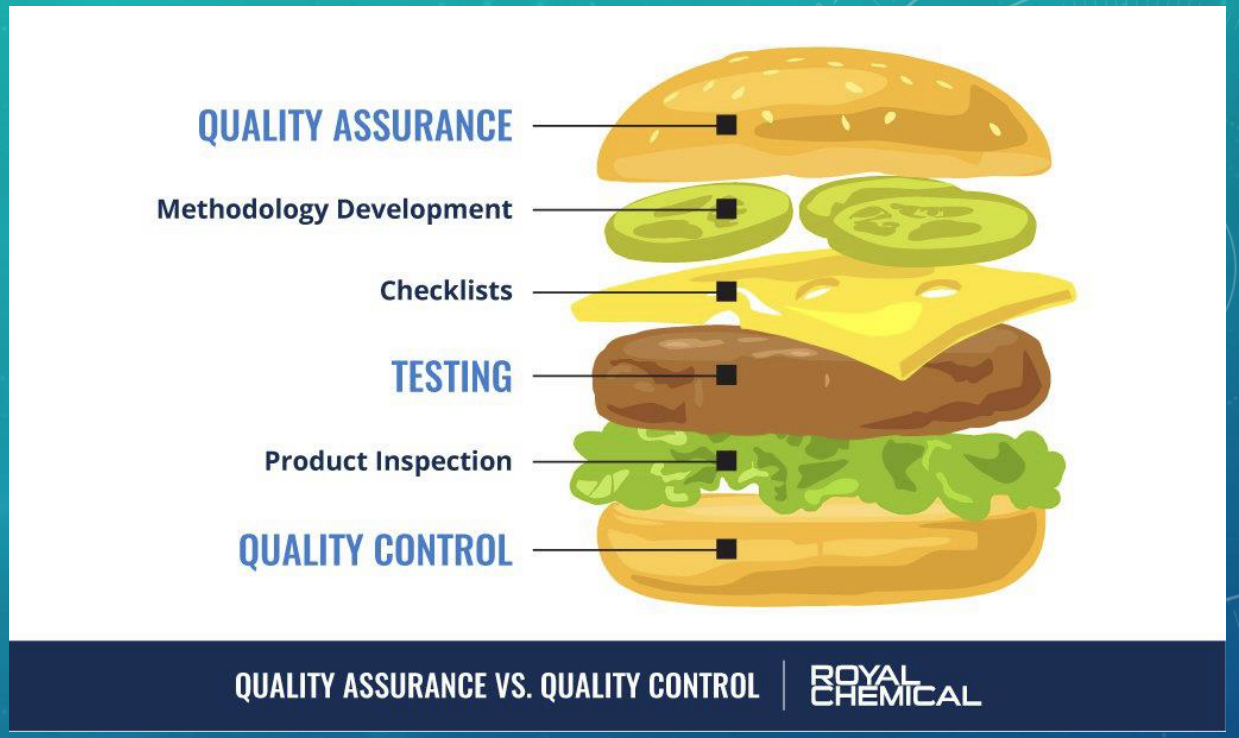
- What are the sources of the data and the selection criteria for pulling data from the repository?
- What parameters are selectable by the user?
- What calculations or other data transformations are required?
- What are the criteria for sorting, page breaks, and totals?
- How should the system respond if no data is returned in response to a query when attempting to generate this report?
- Should the underlying data of the report be made available to the user for ad hoc reporting?
- Can this report be used as a template for a set of similar reports?

PART THREE: QUALITY (CH14)

QUALITY ATTRIBUTES

Reasons to discuss non-functional requirements

- In addition to functionalities, users also have expectations, often unstated, about how well the product will work
- Excellent products reflect an optimal balance of competing quality characteristics – so there is a need to identify and prioritise quality attributes. (Think about .NET IDE. It is so convenient for us to develop a GUI, but the IDE has gone to too complex. Whilst Java allow you to program on any text editor, but it is not easy to implement a GUI compared with .NET – my experience 10 yeas ago.)



Classification

- External quality factors are what the user can see and hence are primarily important to users,
- Internal qualities factors refer to what developer can see, therefore are more significant to development and maintenance staff.
- (More on the next slide)

QUALITY ATTRIBUTES

Two classes:

- User interests
- Developer interests

External quality	Brief description
Availability	The extent to which the system's services are available when and where they are needed
Installability	How easy it is to correctly install, uninstall, and reinstall the application
Integrity	The extent to which the system protects against data inaccuracy and loss
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system
Internal quality	Brief description
Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent components can be used in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, etc.
Verifiability	How readily developers and testers can confirm that the software was implemented correctly

EXTERNAL QUALITY ATTRIBUTES

Availability = MTBF/(MTBF+MTTR) where MTBF is the mean time between failures and MTTR is the mean time in repairing. Availability is closely related to reliability and is strongly affected by the maintainability subcategory of modifiability. Availability can be:

- System up time between XX to YY, down time from AA to BB
- Warmings or alerts before maintenance

Increasing a system's **Installability** reduces the time, cost, user disruption, error frequency, and skill level needed for an installation operation. A measure of a system's installability is the mean time to install the system. Examples are:

- An untrained user shall be able to successfully perform an initial installation of the application in an average of 10 minutes.
- When installing an upgraded version of the application, all customizations in the user's profile shall be retained and converted to the new version's data format if needed.
- The installation program shall verify the correctness of the download before beginning the installation process.
- Installing this software on a server requires administrator privileges.
- Following successful installation, the installation program shall delete all temporary, backup, obsolete, and unneeded files associated with the application.

EXTERNAL QUALITY ATTRIBUTES

Data **Integrity** also addresses the accuracy, completeness and proper formatting of the data.

- Example of requirements:
  - After performing a file backup, the system shall verify the backup copy against the original and report any discrepancies.
  - The system shall protect against the unauthorised addition, deletion, or modification of data.
  - The Chemical Tracking System shall confirm that an encoded chemical structure imported for third-party structure-drawing tools represents a valid chemical structure.
  - The system shall confirm daily that the application executables have not been modified by the addition of unauthorized code.

Interoperability indicates how readily the system works with others, e.g. exchanging data and services with other software systems and being integrated with external hardware devices.

- Example:
  - A web-based health care information system should be able to Import sensing data from IoT, to search information from the Internet and to use predictive models from cloud.

EXTERNAL QUALITY ATTRIBUTES

Performance

Performance dimension	Example
Response time	Number of seconds to display a webpage
Throughput	Credit card transactions processed per second
Data capacity	Maximum number of (records stored in a database -- really? Do we see any user ask the size of a DB?) info can be saved 9 gb. (Google mail says you can save infinity number of emails)
Dynamic capacity	Maximum number of concurrent users of a social media website
Predictability in real-time systems	Foresee road condition of auto car
Latency (I think this is a fabrication. Latency is related to networking. User only sees response time)	Time delays in music recording and production software
Behaviour in degraded modes or overloaded conditions	A natural disaster leads to a massive number of emergency telephone system calls (My MacBook pro cannot cope with MS Teams meeting when I switched on recording.)

EXTERNAL QUALITY ATTRIBUTES

Ways to specify and measure software **Reliability** include the percentage of operations that are completed correctly, the average length of time the system runs before failing (MTBF), and the maximum acceptable probability of a failure during a given time period. Examples:

- One failure in 1000 runs
- MTBF of the card reader component shall be at least 90 days (how many hours does Karl use the card reader in 90 days?)
- Fault rate is 3 in 1000 hours

Robustness is the degree to which a system continues to function properly when confronted with invalid inputs, defects in connected software or hardware components, external attack, or unexpected operating conditions. Good example is the text editor, such as

- If the text editor fails before the user saves the file, it shall recover the contents of the file being edited as of, at most, one minute prior to the failure the next time the same user launches the application. (MS office)

Safety – If a software system is to control a hardware, injuries and damages are possible.

EXTERNAL QUALITY ATTRIBUTES

Security is a major issue with Internet software and cloud computing

Some considerations to examine when eliciting security requirements

- User authorisation or privilege levels (ordinary user, guest user, administrator) and user access controls (the roles and permissions matrix can be a useful tool)
- User identification and authentication (password construction rules, password change frequency, security questions, forgotten logon name or password procedures, biometric identification, account locking after unsuccessful access attempts, unrecognized computer)
- Data privacy (who can create, see, change, copy, print, and delete what information)
- Deliberate (consider) data destruction, corruption, or theft
- Protection against viruses, worms, Trojan horses, spyware, rootkits, and other malware
- Firewall and other network security issues
- Encryption of secure data
- Building audit trails of operations performed and access attempts

EXTERNAL QUALITY ATTRIBUTES

Possible ways to improve **Usability** (user-friendliness, ease of use, and human engineering)

Ease of learning	Ease of use
Verbose prompts	Keyboard shortcuts
Wizards	Rich, customisable menus and toolbars
Visible menu options	Multiple ways to access the same function
Meaningful, plain-language messages	Autocompletion of entries
Help screens and tooltips	Autocorrection of errors
Similarity to other familiar systems	Macro recording and scripting capabilities
Limited number of options and widgets displayed	Ability to carry over information from a previous transaction
	Automatically fill in form fields
	Command-line interface

Does the image below ring the bell for you? People joked about MS windows about how to start the system years ago. Now, even electric cars adopt the same symbol.

INTERNAL QUALITY ATTRIBUTES (???)

Efficiency is closely related to the external quality attribute of performance. Efficiency is a measure of how well the system utilizes processor capacity, disk space, memory, or communication bandwidth. Efficiency - and hence performance - is a driving factor in systems architecture, influencing how a designer elects to distribute computations and functions across system components.

- Examples:
  - At least 30 percent of the processor capacity and memory available to the application shall be unused at the planned peak load conditions.
  - The system shall provide the operator with a warning message when the usage load exceeds 80 percent of the maximum planned capacity.

Ways to measure **Modifiability** include the average time required to add a capability or fix a problem, and the percentage of fixes that are made correctly. It is closely related to maintenance.

- Examples:
  - A certified repair technician shall be able to replace the scanner module in 10 minutes.
  - The printer shall display an error message if replacement ink cartridges were not inserted in the proper slots.

INTERNAL QUALITY ATTRIBUTES

Portability has become increasingly important as applications must run in multiple environments, such as Windows, Mac, and Linux; iOS and Android; and PCs, tablets, and phones. Some practitioners include the ability to internationalize and localize a product under the heading of portability.

- Examples:
- Modifying the iOS version of the application to run on Android devices shall require changing no more than 10 percent of the source code.
- The user shall be able to port browser bookmarks to and from Firefox, Internet Explorer, Opera, Chrome, and Safari.

Reusable software must be modular, well documented, independent of a specific application and operating environment, and somewhat generic in capability.

Scalability address the ability of the application to grow to accommodate more users, data, servers, geographic locations, transactions, network traffic, searches, and other services without compromising performance or correctness. Scalability has both hardware and software implications.

Designing software for **Verifiability** means making it easy to place the software into the desired pre-test state, to provide the necessary test data, and to observe the result of the test.

IDENTIFICATION AND RANKING

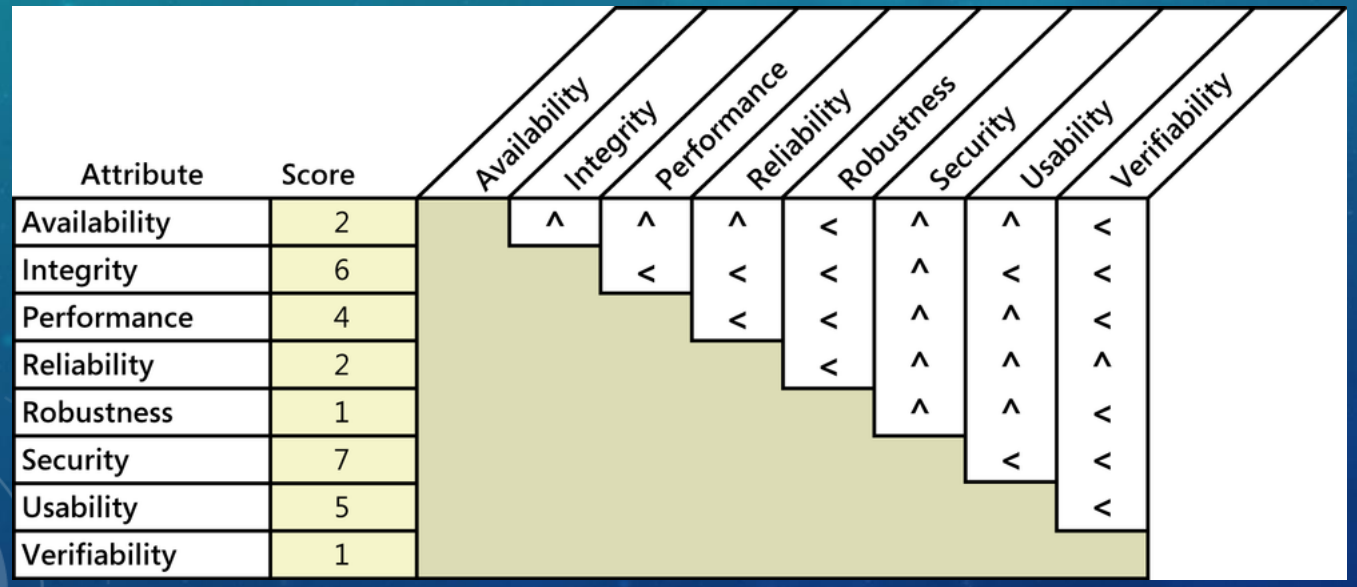
The need for ranking

- Not all software system need to address all the internal and external quality attributes.
- Think about internal alone. It is a 6-d space. Some dimensions can be competing. It is hard to balance.
- Examples about ranking:
  - Embedded systems: performance, efficiency, reliability, robustness, safety, security, usability,
  - Internet and corporate applications: availability, integrity, interoperability, performance, scalability, security, usability
  - Desktop and mobile systems: performance, security, usability

IDENTIFICATION AND RANKING

Process

- Step 1: Start with a broad taxonomy -- Begin with a rich set of quality attributes to consider, to reduce the likelihood of overlooking an important quality dimension.
- Step 2: Reduce the list -- Engage a cross-section of stakeholders to set up quality goals by assessing which of the attributes are likely to be important to the project.
- Step 3: Prioritise the attributes -- Prioritising the pertinent attributes and setitng up the focus for future elicitation discussions



Attribute	Score
Availability	2
Integrity	6
Performance	4
Reliability	2
Robustness	1
Security	7
Usability	5
Verifiability	1

	Availability	Integrity	Performance	Reliability	Robustness	Security	Usability	Verifiability
Availability		^	^	^	<	^	^	<
Integrity			<	<	<	^	<	<
Performance				<	<	^	^	<
Reliability					<	^	^	^
Robustness						^	^	<
Security							<	<
Usability								<
Verifiability								

In, Pe, Re, Se, Us, Av. Ro, Ve
Se, In, Pe, Re, Us, Av, Ro, Ve
Se, In, Pe, Us, Re, Av, Ro, Ve
……..

IDENTIFICATION AND RANKING

- Step 4: Elicit specific expectations (assigning attribute with a value) – Questions to ask are:
  - What would be a reasonable or acceptable response time for retrieval of a typical patent application in response to a query?
  - What would users consider an unacceptable response time for a typical query?
  - How many simultaneous users do you expect on average?
  - What's the maximum number of simultaneous users that you would anticipate?
  - What times of the day, week, month, or year have much heavier usage than usual?
- Step 5: Specify well-structured quality requirements -- using smart (Specific, Measurable, Achievable, Relevant, and Time-sensitive.)

QUALITY ATTRIBUTE TRADE-OFFS

This is to balance quality attributes requirements

- Example:
  - + means indicates that increasing the attribute in the corresponding row usually has a positive effect on the attribute in the column.
  - - means that increasing the attribute in that row generally adversely affects the attribute in the column.

	Availability	Efficiency	Installability	Integrity	Interoperability	Modifiability	Performance	Portability	Reliability	Reusability	Robustness	Safety	Scalability	Security	Usability	Verifiability
Availability									+		+					
Efficiency	+				−	−	+	−			−		+		−	
Installability	+								+					+		
Integrity			−		−		−			−		+		+	−	−
Interoperability	+		−	−			−	+	+		+	−		−		
Modifiability	+		−				−		+	+			+			+
Performance		+			−	−		−			−		−		−	
Portability		−			+	−	−			+				−	−	+
Reliability	+	−		+		+	−				+	+		+	+	+
Reusability		−		−	+	+	−	+						−		+
Robustness	+	−	+	+	+		−		+			+	+	+	+	
Safety		−		+	+		−				+			+	−	−
Scalability	+	+		+			+	+	+		+					
Security	+			+	+		−	−	+		+	+			−	−
Usability		−	+				−	−	+		+	+				−
Verifiability	+		+	+		+			+	+	+	+		+	+	

PART FOUR: RISK MIGRATION (CH15)

RISKS AND PROTOTYPING

Risk, what risk?

- Requirements can be unclear because of uncertainties caused by
  - vagueness
  - ambiguity
  - incompleteness
  - inaccuracy
  - **changes**
- These uncertainty can risk intended software leading to a system that does not satisfy users true requirements
- There are a lots more than user requirements, such as GDPR, IP, ethics issues such as the consequences of a “malfunctional” software, etc.
- Prototypes can realise and then reduce risks. They show how intended software would work and the parts that do not work as well as the consequences.

RISKS AND PROTOTYPING

Types or purposes of prototypes

- From scope: mock-up (for users) and proof-of-concept (for developers)
- Frome whether you will use it in future: throwaway and evolutionary
- From form: paper-based and electronic

Mock-up: focuses on a portion of the interface.

- For the purpose of exploring some specific behaviours of the intended system, with the goal of refining the requirements. The mock-up helps users to judge whether a system based on the prototype will let them do their job in a reasonable way.
- Mock-ups can demonstrate the available functional options the user will have, the look and feel of the user interface (colours, layout, graphics, controls), and the navigation structure.
- Mock-up doesn't perform any useful work, although it looks as if it should, in the sense of "interface", i.e. it only show appearance but not interactions.

RISKS AND PROTOTYPING

A **proof of concept** implements a slice of application functionality from the user interface through all the technical services layers.

- A proof-of-concept prototype works like the real system
- Develop a proof-of-concept when you're uncertain whether a proposed architectural approach is feasible and sound, or when you want to optimize algorithms, evaluate a proposed database schema, confirm the soundness of a cloud solution, or test critical timing

Throwaway prototype is exploratory only

- For the purpose of answering questions, resolve uncertainties, and improve requirements quality
- Because it will be discarded, system quality (internal and external) are ignored. So, make sure you must not allow low-quality code from a throwaway prototype to migrate into a production system

Evolutionary prototype provides a solid architectural foundation for building the product incrementally as the requirements become clear over time. It is of high-quality in terms of software system quality.

RISKS AND PROTOTYPING

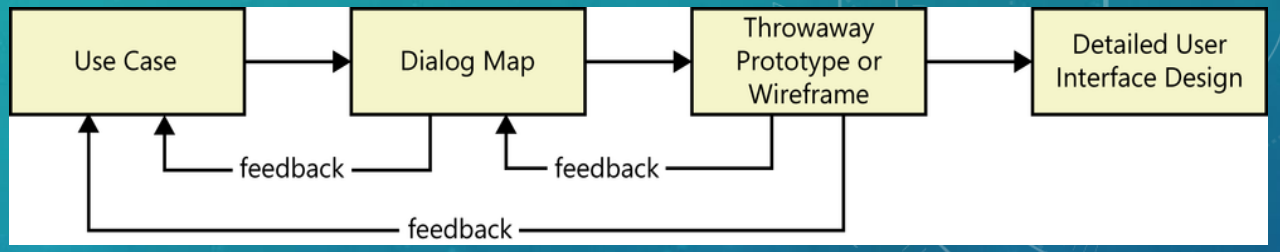
	Throwaway	Evolutionary
Mock-up	Clarify and refine user and functional requirements. Identify missing functionality. Explore user interface approaches.	Implement core user requirements. Implement additional user requirements based on priority. Implement and refine websites. Adapt system to rapidly changing business needs.
Proof of concept	Demonstrate technical feasibility. Evaluate performance. Acquire knowledge to improve estimates for construction.	Implement and grow core multi-tier functionality and communication layers. Implement and optimise core algorithms. Test and tune performance.

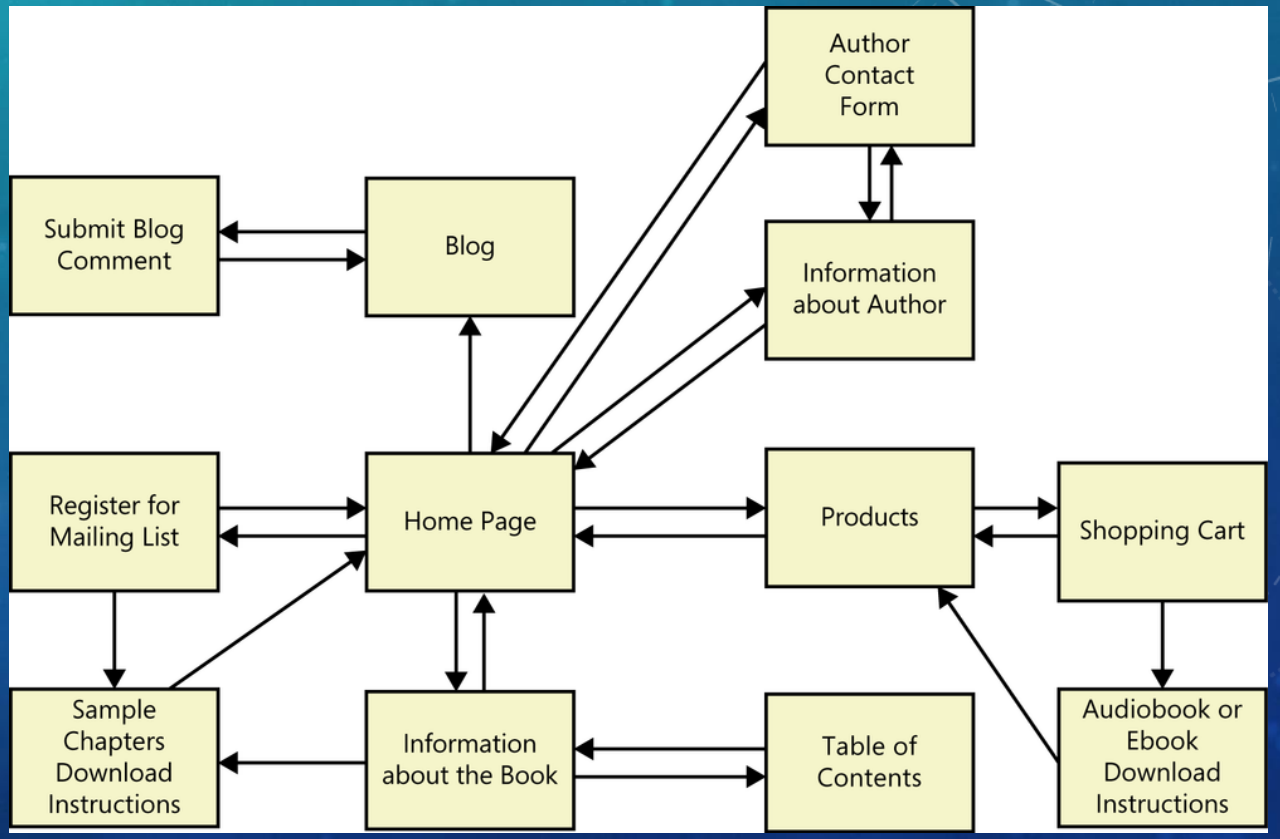
RISKS AND PROTOTYPING

Example:

Digitalising the book of “Peals from Sand”, so the book can be read online. Plus blog, email, etc.

User class (Actors)	Use case
Visitor	Get Information about the Book Get Information about the Author Read Sample Chapters Read the Blog Contact the Author
Customer	Order a Product Download an Electronic Product Request Assistance with a Problem
Administrator	Manage the Product List Issue a Refund to a Customer Manage the Email List





RISKS AND PROTOTYPING