# FOUR Ws

## PART ONE: WHAT (CH1)

# WHAT ARE SOFTWARE REQUIREMENTS?

Users (or end users) are the ones who will use the software system.

- Terms: user requirement, software requirement, business requirement, functional requirement, system requirement, product requirement, project requirement, user story, feature, constraint, etc.
- Definition: *Requirements are asset of specifications of what should be implemented. They are descriptions of how the system should behave, or about system properties or attributes. They may be a constraint on the development process of the system.*
- Types of requirements (see the list)

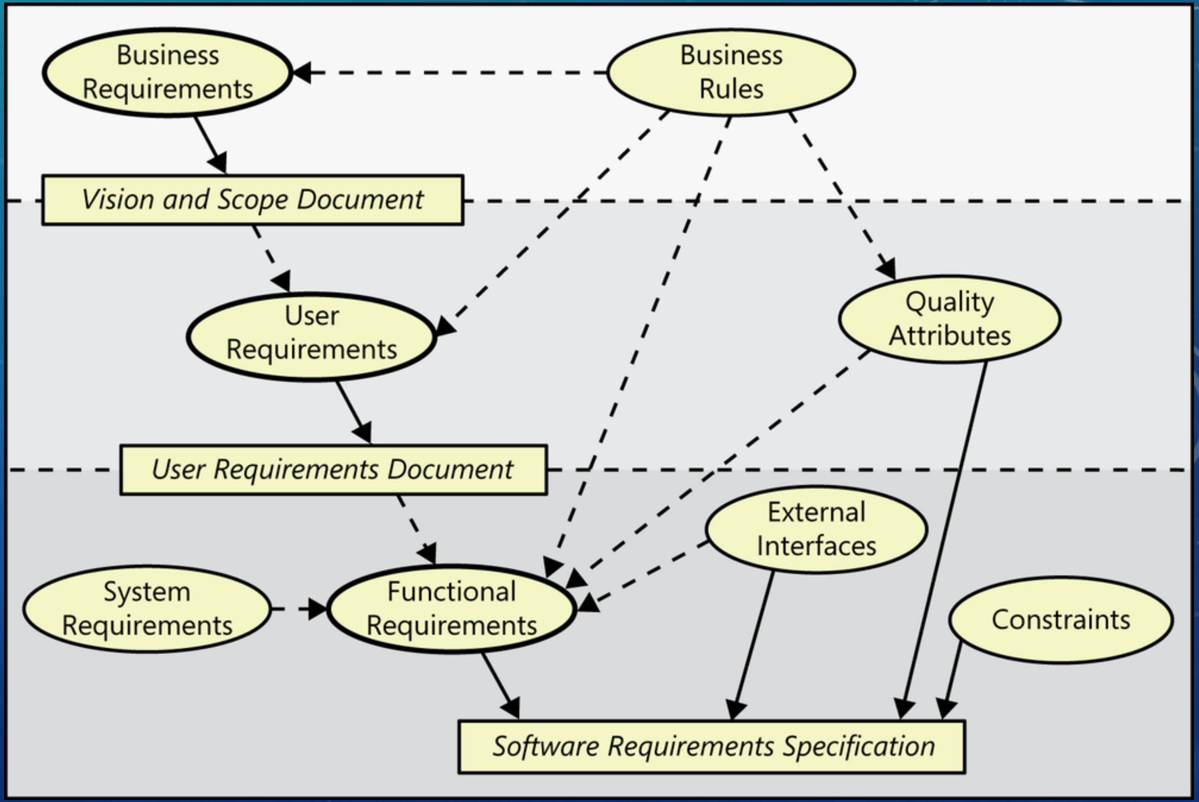
| Term | DefinitionProvides end-to-end IT solutions, Network Support |
|---|---|
| **Business requirement** | A high-level business objective of the organisation that builds a product or of a customer who procures it. |
| **Business rule** | A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements. |
| **Constraint** | A restriction that is imposed on the choices available to the developer for the design and construction of a product. |
| **External interface requirement** | A description of a connection between a software system and a user, another software system, or a hardware device. |
| **Feature** | One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements. |
| **Functional requirement** | A description of a behaviour that a system will exhibit under specific conditions. |
| **Nonfunctional requirement** | A description of a property or characteristic that a system must exhibit or a constraint that it must respect. |
| **Quality attribute** | A kind of non-functional requirement that describes a service or performance characteristic of a product. |
| **System requirement** | A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware. |
| **User requirement** | A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute. |

# WHAT ARE SOFTWARE REQUIREMENTS?

**Software requirements include three distinct levels: business requirements, user requirements, and functional requirements**

- Top level: Vision and scope – business objectives and expected benefit, giving the reasons for developing the software (why)
- 2nd level: User requirements – tasks to perform and software attributes and characteristics (what)
- 3rd level: Software specification --
  - Functional requirements – system's behaviours
  - Systems requirements – structure of the system
  - External interface – connections to other systems
  - Constraints – what developers can do and what cannot

Example: London Ambulance Service (LAS) Computer-Aided Dispatching System (CAD) is provided in the handout. It explains some types of requirements.



# WHAT ARE SOFTWARE REQUIREMENTS?

Example: University of London is now an umbrella of a number of “small” universities in London. One of these small-size universities has been updating its student management system (basically a data system consisting of databases and reporting tools) from Agresso to SITS.

- Business requirements – solving all problems of Agresso (The two systems have the similar functionalities and Agresso fits smaller universities. Agresso’s functionalities were not demonstrated because of the so-called IT personnel do not really understand the business rules of admissions, registry, academics etc.) Reducing human errors in student management by automating the process
- User requirements – the requirements from staff who use the system, including admissions, registry, admin of academic departments, academics, etc.
- System specification or functionalities – students’ profiles creation and maintenance , creating modules of each semester for every students, students’ mark publishing, progression, awards (1st, 2nd upper, 2nd lower, 3rd, and no award), timetables, GUI, etc.
- Constraints – GDPR, security, access, EDI, etc.

# WHAT ARE SOFTWARE REQUIREMENTS?

Two different scenarios (business models):

- A software firm to develop software for clients (Some software firms have their own expert areas and they develop software of the expert areas for their clients.)
    - Manager, marketing personnel and users are all on client side
    - BA is the interface between the firm and its clients
    - Example: excoms is an SME Provides end-to-end IT solutions, computer network support, cybersecurity solutions.
- A software firm to customise their products to fit their clients' specific needs
    - BA is a bridge connecting CEO level of the firms and the development team.
    - It is also the interface between the customers and the development team.
    - Example: Grassroots was a software company. Its main product was a project management software designed based on the company's philosophy of project management. So, it does not only sell its product, but also its management philosophy.

# CW TASKS 1, 2

**CW contains a series tasks on a group work basis.**

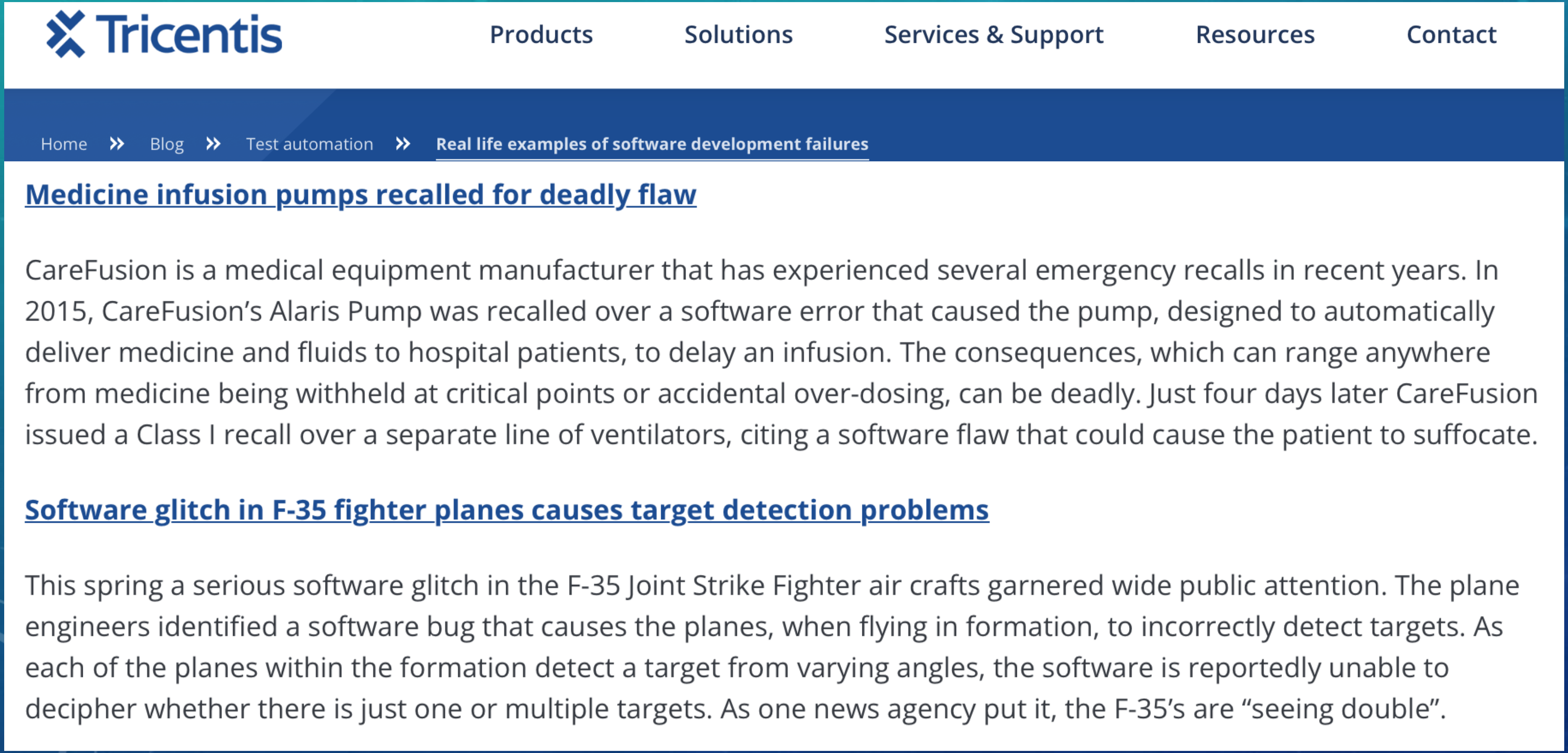
- Tasks 1 to 3:
  - Grouping with group members, a leader and a agreed scenario
  - Detailed description of the project/scenario agreed

| Term | DefinitionProvides end-to-end IT solutions, Network Support |
|---|---|
| **Business requirement** | A high-level business objective of the organisation that builds a product or of a customer who procures it. |
| **Business rule** | A policy, guideline, standard, or regulation that defines or constrains some aspect of the business. Not a software requirement in itself, but the origin of several types of software requirements. |
| **Constraint** | A restriction that is imposed on the choices available to the developer for the design and construction of a product. |
| **External interface requirement** | A description of a connection between a software system and a user, another software system, or a hardware device. |
| **Feature** | One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements. |
| **Functional requirement** | A description of a behaviour that a system will exhibit under specific conditions. |
| **Nonfunctional requirement** | A description of a property or characteristic that a system must exhibit or a constraint that it must respect. |
| **Quality attribute** | A kind of non-functional requirement that describes a service or performance characteristic of a product. |
| **System requirement** | A top-level requirement for a product that contains multiple subsystems, which could be all software or software and hardware. |
| **User requirement** | A goal or task that specific classes of users must be able to perform with a system, or a desired product attribute. |

## PART TWO: WHY (CHS1 AND 2)

## WHY ARE SOFTWARE REQUIREMENTS IMPORTANT?

- Various studies suggest that errors introduced during requirements activities count for 40 to 50 percent of all defects found in a software product. (negative impacts can be found in: https://www.tricentis.com/blog/real-life-examples-of-software-development-failures/)

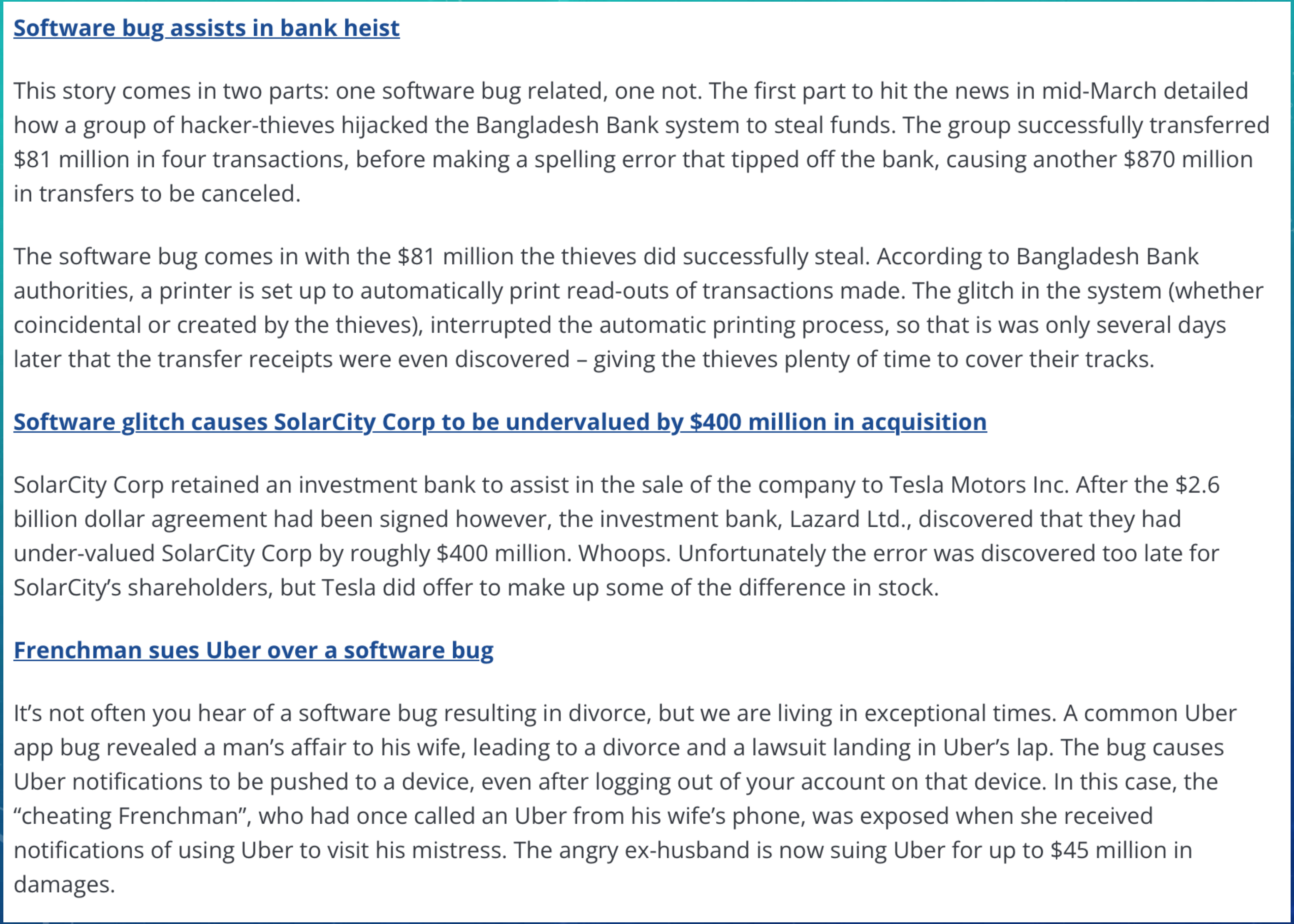

Tricentis
Products
Solutions
Services & Support
Resources
Contact
Home » Blog » Test automation » Real life examples of software development failures

**Medicine infusion pumps recalled for deadly flaw**

CareFusion is a medical equipment manufacturer that has experienced several emergency recalls in recent years. In 2015, CareFusion's Alaris Pump was recalled over a software error that caused the pump, designed to automatically deliver medicine and fluids to hospital patients, to delay an infusion. The consequences, which can range anywhere from medicine being withheld at critical points or accidental over-dosing, can be deadly. Just four days later CareFusion issued a Class I recall over a separate line of ventilators, citing a software flaw that could cause the patient to suffocate.

**Software glitch in F-35 fighter planes causes target detection problems**

This spring a serious software glitch in the F-35 Joint Strike Fighter air crafts garnered wide public attention. The plane engineers identified a software bug that causes the planes, when flying in formation, to incorrectly detect targets. As each of the planes within the formation detect a target from varying angles, the software is reportedly unable to decipher whether there is just one or multiple targets. As one news agency put it, the F-35's are "seeing double".



## Software bug assists in bank heist

This story comes in two parts: one software bug related, one not. The first part to hit the news in mid-March detailed how a group of hacker-thieves hijacked the Bangladesh Bank system to steal funds. The group successfully transferred $81 million in four transactions, before making a spelling error that tipped off the bank, causing another $870 million in transfers to be canceled.

The software bug comes in with the $81 million the thieves did successfully steal. According to Bangladesh Bank authorities, a printer is set up to automatically print read-outs of transactions made. The glitch in the system (whether coincidental or created by the thieves), interrupted the automatic printing process, so that is was only several days later that the transfer receipts were even discovered – giving the thieves plenty of time to cover their tracks.

## Software glitch causes SolarCity Corp to be undervalued by $400 million in acquisition

SolarCity Corp retained an investment bank to assist in the sale of the company to Tesla Motors Inc. After the $2.6 billion dollar agreement had been signed however, the investment bank, Lazard Ltd., discovered that they had under-valued SolarCity Corp by roughly $400 million. Whoops. Unfortunately the error was discovered too late for SolarCity's shareholders, but Tesla did offer to make up some of the difference in stock.

## Frenchman sues Uber over a software bug

It's not often you hear of a software bug resulting in divorce, but we are living in exceptional times. A common Uber app bug revealed a man's affair to his wife, leading to a divorce and a lawsuit landing in Uber's lap. The bug causes Uber notifications to be pushed to a device, even after logging out of your account on that device. In this case, the "cheating Frenchman", who had once called an Uber from his wife's phone, was exposed when she received notifications of using Uber to visit his mistress. The angry ex-husband is now suing Uber for up to $45 million in damages.

## WHY ARE SOFTWARE REQUIREMENTS IMPORTANT?

- Rework often consumes 30 to 50 percent of total development cost, and requirements errors can count for 70 to 85 percent of the rework cost
- Reasons for poor software requirements
  - Insufficient user involvement (This is also addressed in Agile)
  - Inaccurate planning (https://www.entrepreneur.com/article/329019)
  - Creeping user requirements (creeping means enlarge without being noticed.)
  - Ambiguous requirements (e.g. user friendly is not tangible and measurable)
  - Gold plating (meaning any expensive nonessential item, convenience, or features)
  - Overlooked stakeholders (Someone who received little attention. Who will be that ”someone” in your experience? )

# WHY ARE SOFTWARE REQUIREMENTS IMPORTANT?

**Benefits from a high-quality requirements process**

- Fewer defects in requirements and in the delivered product.
- Reduced development rework.
- Faster development and delivery.
- Fewer unnecessary and unused features.
- Lower enhancement costs.
- Fewer miscommunications.
- Reduced scope creep.
- Reduced project chaos.
- Higher customer and team member satisfaction.
- Products that do what they're supposed to do.

**Four principles of high-quality requirements**
https://seilevel.com/requirements/the-four-attributes-of-high-quality-requirements (Requirement Blog)

1. Atomic/ specific
An atom is the "smallest indivisible unit of matter that retains the properties of an element."

2. Unambiguous
When two different people claim they understand the same requirement, they can have two different interpretations of what is being stated.

3 Testable/measurable
The requirement must provide a way to determine if the software built is sufficient. There must be a finite point distinguishing between passing and failing.

4. Necessary
If you ask the stakeholder, every requirement will probably be "necessary".

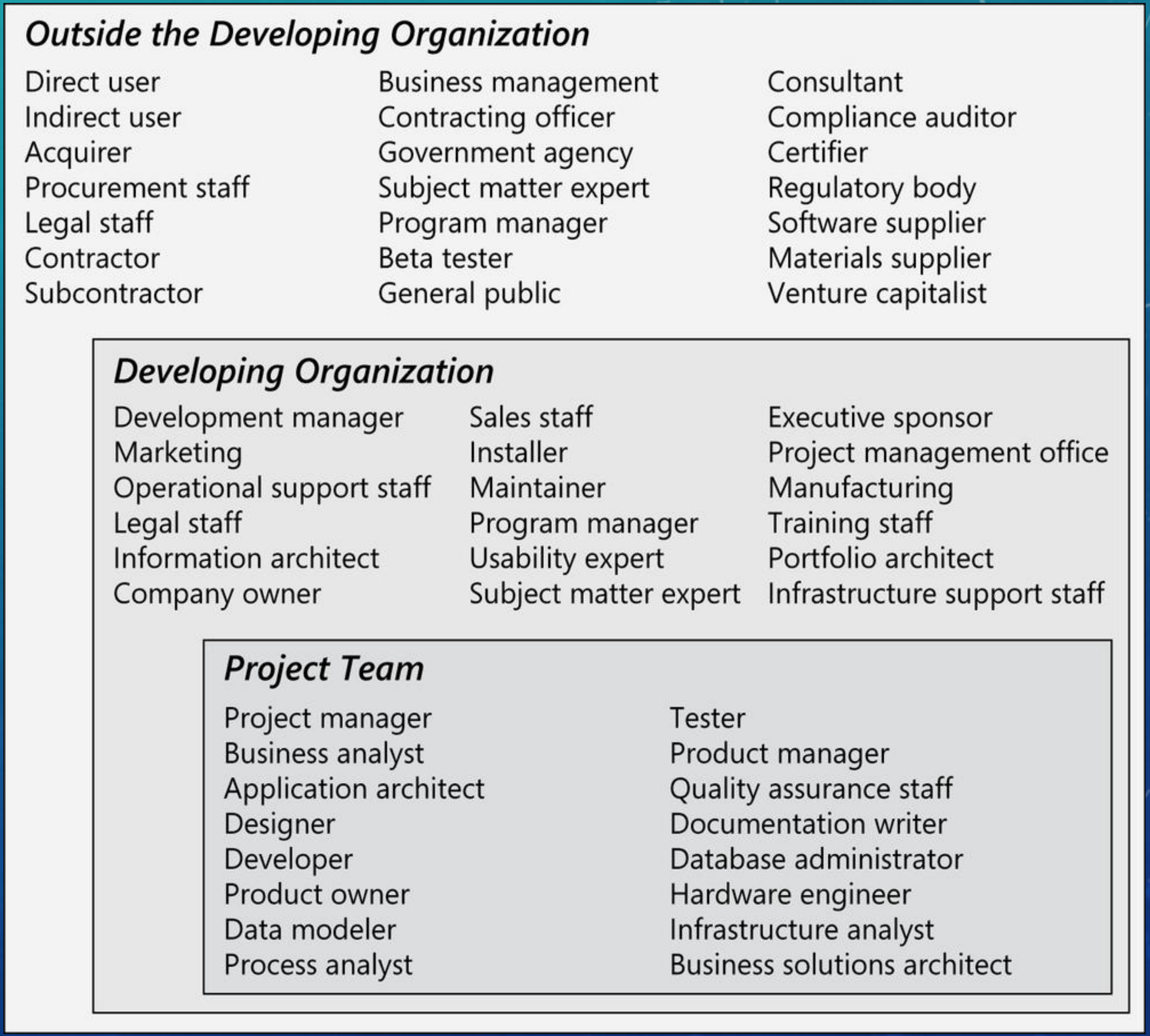
## YOUR PROFESSIONS

- The Rise of GenAI (or AGI)
- Threat or opportunity?
- Current situation and the indication

## SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

A *stakeholder* is a person, group, or organisation that is actively involved in a project, is affected by its process or outcome, or can influence its process or outcome.

Stake – wooden pole driven into ground to mark boundary.
Stakeholder – people who hold the stake when it is driven into ground, i.e. people who decide where to place the stake because of their own interests.



***Outside the Developing Organization***

| | | |
|---|---|---|
| Direct user | Business management | Consultant |
| Indirect user | Contracting officer | Compliance auditor |
| Acquirer | Government agency | Certifier |
| Procurement staff | Subject matter expert | Regulatory body |
| Legal staff | Program manager | Software supplier |
| Contractor | Beta tester | Materials supplier |
| Subcontractor | General public | Venture capitalist |

***Developing Organization***

| | | |
|---|---|---|
| Development manager | Sales staff | Executive sponsor |
| Marketing | Installer | Project management office |
| Operational support staff | Maintainer | Manufacturing |
| Legal staff | Program manager | Training staff |
| Information architect | Usability expert | Portfolio architect |
| Company owner | Subject matter expert | Infrastructure support staff |

***Project Team***

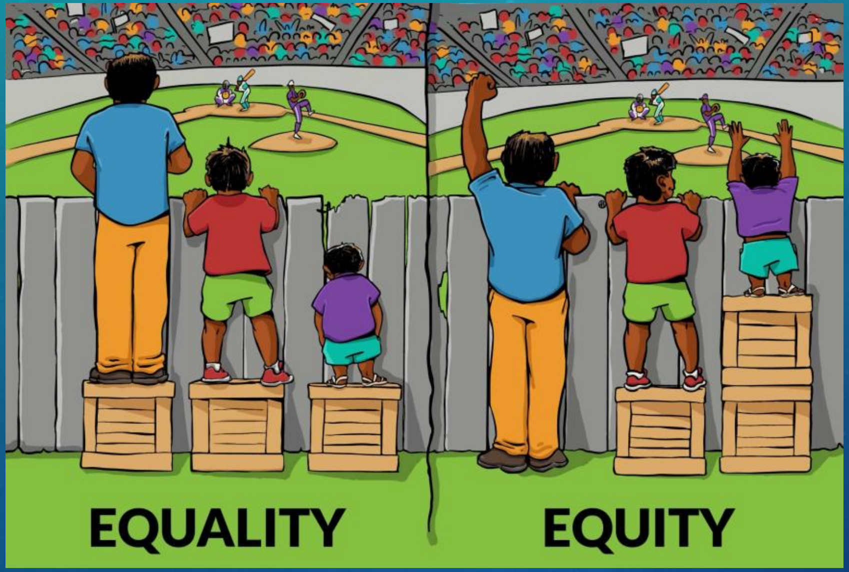
| | |
|---|---|
| Project manager | Tester |
| Business analyst | Product manager |
| Application architect | Quality assurance staff |
| Designer | Documentation writer |
| Developer | Database administrator |
| Product owner | Hardware engineer |
| Data modeler | Infrastructure analyst |
| Process analyst | Business solutions architect |

## CW TASK 3

- Task 3:
  - List of all stakeholders of your projects
  - Explanations of the roles of the stakeholders would play

## SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

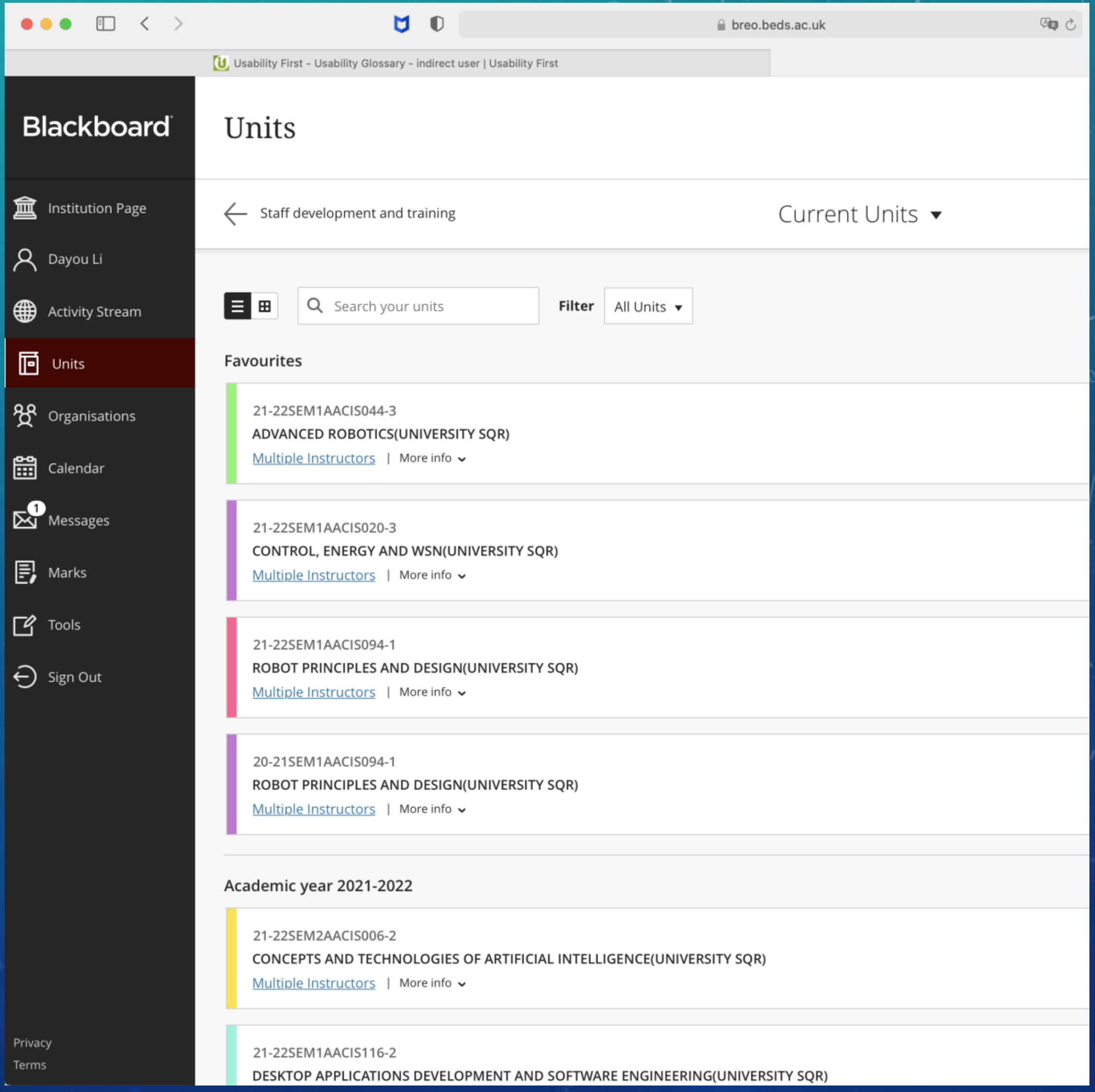
Avoid overlooking some important community but focus on the core set whose input you really need, to make sure you understand all of the project's requirements and constraints so your team can deliver the right solution



## SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

Customers are a subset of stakeholders.

- A *customer* is an individual or organization that derives either direct or indirect benefit from a product. Software customers could request, pay for, select, specify, use, or receive the output generated by a software product.
- The customers include the direct user, indirect user, executive sponsor, procurement staff, and acquirer.
- Some stakeholders are not customers, such as legal staff, compliance auditors, suppliers, contractors, an venture capitalists.
- *End users* are a subset of customers.
- User requirements should come from the end user who will actually use the product, either directly or indirectly.



Units

Staff development and training — Current Units

Search your units — Filter: All Units

Favourites

21-22SEM1AACIS044-3
ADVANCED ROBOTICS(UNIVERSITY SQR)
Multiple Instructors | More info

21-22SEM1AACIS020-3
CONTROL, ENERGY AND WSN(UNIVERSITY SQR)
Multiple Instructors | More info

21-22SEM1AACIS094-1
ROBOT PRINCIPLES AND DESIGN(UNIVERSITY SQR)
Multiple Instructors | More info

20-21SEM1AACIS094-1
ROBOT PRINCIPLES AND DESIGN(UNIVERSITY SQR)
Multiple Instructors | More info

Academic year 2021-2022

21-22SEM2AACIS006-2
CONCEPTS AND TECHNOLOGIES OF ARTIFICIAL INTELLIGENCE(UNIVERSITY SQR)
Multiple Instructors | More info

21-22SEM1AACIS116-2
DESKTOP APPLICATIONS DEVELOPMENT AND SOFTWARE ENGINEERING(UNIVERSITY SQR)

## SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

Excellent requirements result from effective collaboration between developers and customers (in particular, actual users)—a partnership.

Discussion:
Difference between teamwork, cooperation and collaboration?

# SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

Legal reasons (in the USA)

Requirements Bill of Rights for Software Customers

**You have the right to**

1. Expect BAs to speak your language.
2. Expect BAs to learn about your business and your objectives.
3. Expect BAs to record requirements in an appropriate form.
4. Receive explanations of requirements practices and deliverables.
5. Change your requirements.
6. Expect an environment of mutual respect.
7. Hear ideas and alternatives for your requirements and for their solution.
8. Describe characteristics that will make the product easy to use.
9. Hear about ways to adjust requirements to accelerate development through reuse.
10. Receive a system that meets your functional needs and quality expectations.

Requirements Bill of Responsibilities for Software Customers

**You have the responsibility to**

1. Educate BAs and developers about your business.
2. Dedicate the time that it takes to provide and clarify requirements.
3. Be specific and precise when providing input about requirements.
4. Make timely decisions about requirements when asked.
5. Respect a developer's assessment of the cost and feasibility of requirements.
6. Set realistic requirement priorities in collaboration with developers.
7. Review requirements and evaluate prototypes.
8. Establish acceptance criteria.
9. Promptly communicate changes to the requirements.
10. Respect the requirements development process.

## SOFTWARE REQUIREMENTS FROM CUSTOMERS PERSPECTIVE

**Decision making with the development team**

- There can be hundreds of decisions to make on software projects; often, they are on the critical path to being able to move ahead. You might need to resolve some conflict, accept (or reject) a proposed change, or approve a set of requirements for a specific release.
- Agreement between development and customers
  - Reaching agreement on the requirements for the product to be built, or for a specific portion of it, is at the core of the customer-developer partnership. Multiple parties are involved in this agreement:
    - Customers agree that the requirements address their needs.
    - Developers agree that they understand the requirements and that the requirements are feasible (doable, achievable).
    - Testers agree that the requirements are verifiable.
    - Management (from customer side) agrees that the requirements will achieve their business objectives.

## CW TASK 4

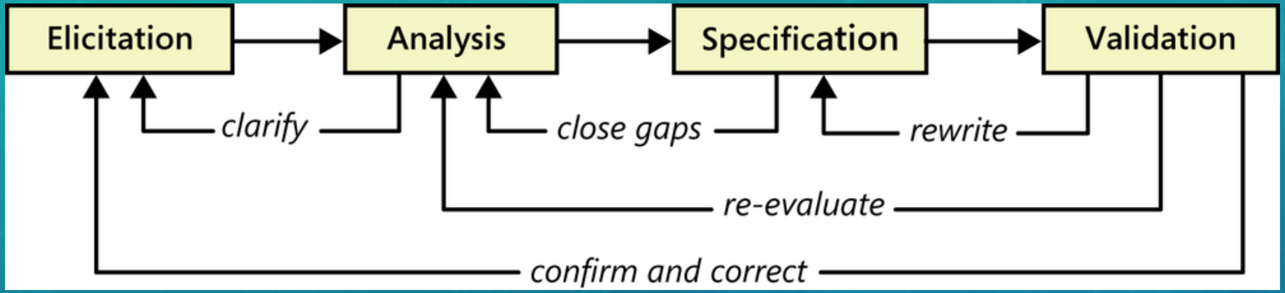
- To identify requirements for all different types from all stakeholders
- To classify the requirements
- To explain the requirements and types.

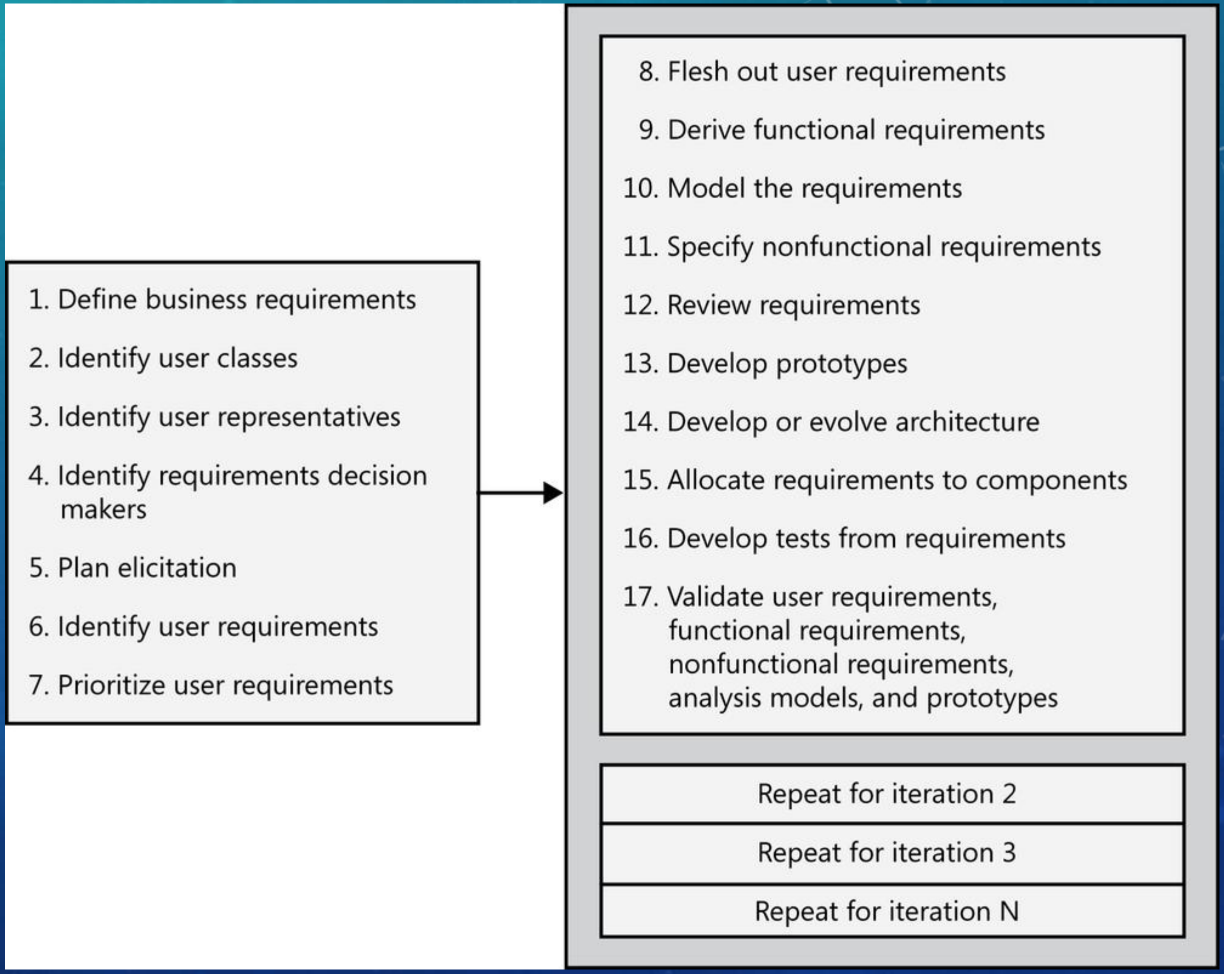
## PART THREE: HOW (CH3)

# HOW TO DEVELOP SOFTWARE REQUIREMENTS?

**Software requirements development process**

Four iterative, interwoven and incremental phases OR, 17 steps





1. Define business requirements
2. Identify user classes
3. Identify user representatives
4. Identify requirements decision makers
5. Plan elicitation
6. Identify user requirements
7. Prioritize user requirements

8. Flesh out user requirements
9. Derive functional requirements
10. Model the requirements
11. Specify nonfunctional requirements
12. Review requirements
13. Develop prototypes
14. Develop or evolve architecture
15. Allocate requirements to components
16. Develop tests from requirements
17. Validate user requirements, functional requirements, nonfunctional requirements, analysis models, and prototypes

Repeat for iteration 2

Repeat for iteration 3

Repeat for iteration N

# HOW TO DEVELOP SOFTWARE REQUIREMENTS?

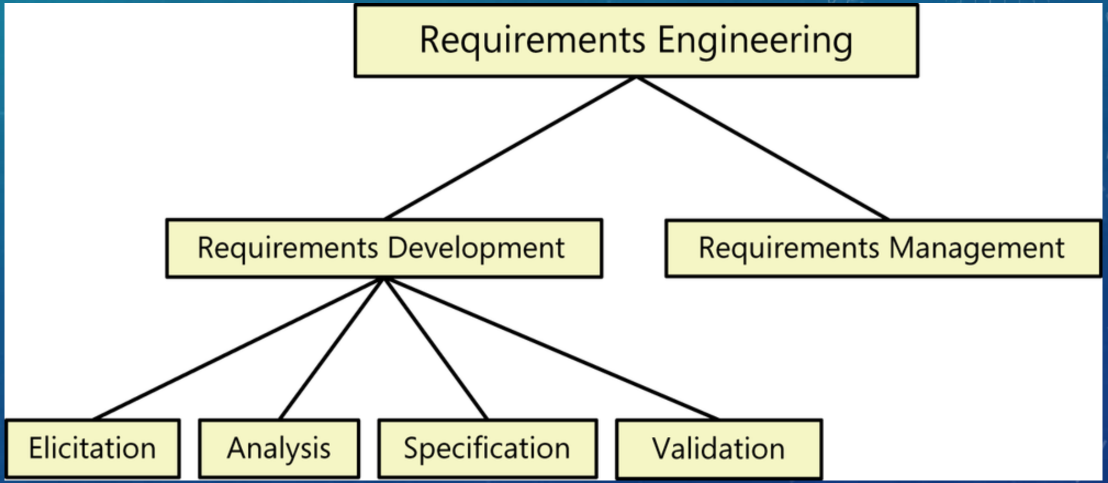
**Elicitation** encompasses all of the activities involved with discovering requirements starting from the top level. The activities can be interviews, workshops, document analysis, prototyping, and others, aiming to:

- Identify the product's expected user classes and other stakeholders (from whom the requirements should come from)

Example of identifying "other stakeholders"

Think about: whether would staff in Finance department use SITS?

They do as they need to record students' fee situation.



(This diagram is not a good one. Requirement engineering also include requirement implementation.)

## HOW TO DEVELOP SOFTWARE REQUIREMENTS?

- Understanding user tasks and goals, and the business objectives with which those tasks align. Not all tasks taken by users is relevant to need for the systems and some can be irrelevant. (what the requirements are about and why)

  Example of “irrelevant task”
  Think about: whether would recording external examiners’ comments be related to the use of SITS?
  No, as the comments has nothing to do with the students.

- Learn about the environment in which the new product will be used (conditions)

  Example of “environment” (may not necessarily be physical environment)
  Think about: what would the admin’s IT skills be regarding SITS development?

- Understand functionality needs and their quality expectations of each user class. (functional and non-functional requirements)

## HOW TO DEVELOP SOFTWARE REQUIREMENTS?

**Analysing** requirements leads requirement progression from one level to another

- It is a process having the requirements as the raw material inputs and a richer and more precise understanding of each requirement as the outputs.
- Analysis activities:
  - Analysing the information received from users to distinguish their task goals from functional requirements, quality expectations, business rules, suggested solutions, and other information
  - Decomposing high-level requirements into an appropriate level of detail
  - Deriving functional requirements from other requirements information
  - Understanding the relative importance of quality attributes
  - Allocating requirements to software components defined in the system architecture (which can be the case in the second type scenario where product is already developed)
  - Negotiating implementation priorities (you know this is important in Agile)
  - Identifying gaps in requirements or unnecessary requirements as they relate to the defined scope

## HOW TO DEVELOP SOFTWARE REQUIREMENTS?

Requirements **specification** involves representing and storing the collected requirements knowledge in a persistent and well-organised fashion. The principal activity is:

"Translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences."

- This is a "house keeping" task and it is a good habit. It helps to find the related items when we need them in the later stages of software development.
- This is equivalent to data pre-processing in ML
- Sorting is very important (classification + sorting according to importance).

| | | | | |
|---|---|---|---|---|
| 5 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 | 3 |
| 8 | 8 | 8 | 8 | 5 |
| 1 | 5 | 5 | 5 | 8 |

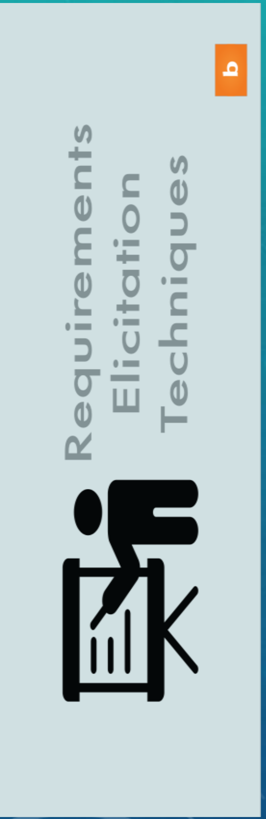
What is the algorithm used? Can you implement?

## HOW TO DEVELOP SOFTWARE REQUIREMENTS?

Requirements **validation** confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives. The central activities are:

- Reviewing the documented requirements to correct any errors/mistakes or anything you are  not very clear about before the development group accepts them.
- Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives. (You must know acceptance tests used in Agile. An acceptance test is of the format of “black-box” test, i.e. you feed with inputs and check whether the output is the one you expected, known as validation.)

# GOOD PRACTICES

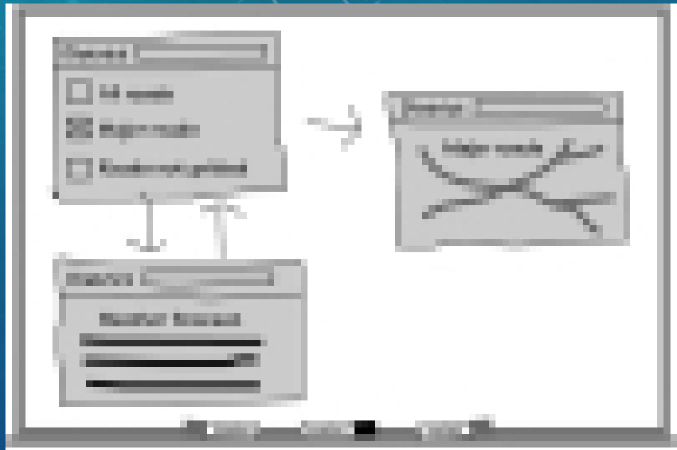


**In the four phases**

- **Elicitation**
  - Define vision and scope
  - Identify user classes (according the way they would use the software)
  - Select product champions (key contact person for each class)
  - Conduct focus groups
  - Identify user requirements
  - Identify system events and responses (to understand the relations between each items of requirements)
  - Hold elicitation interviews (you have to be well-prepared)
  - Hold facilitated elicitation workshops (like what I did for DPS)
  - Observe users performing their jobs (Natural languages can be defective and unclear)
  - Distribute questionnaires (ensure at least 30 to 40% back to you)
  - Perform document analysis
  - Examine problem reports
  - Reuse existing requirements (if you could find one)

## GOOD PRACTICES

- **Analysis**
  - Model the application environment
  - Create prototypes
  - Analyse feasibility
  - Prioritise requirements
  - Create a data dictionary
  - Model the requirement (scenario-based modelling, data modelling, flow-oriented modelling, class-based modelling and behavioural modelling)
  - Analyse interfaces (Between different sets of requirements)
  - Allocate requirements to subsystem (This might be too early to do, but you can classify requirements into different sets.)



## GOOD PRACTICES

- **Specification**
  - Adopt requirement document templates (So requirements can be uniformed and more transparent)
  - Identify requirement origins (whether it comes from the end users or not)
  - Uniquely label each requirement
  - Record business rules
  - Specify non-functional requirements
- **Validation**
  - Review the requirements
  - Test the requirements
  - Define acceptance criteria
  - Simulate the requirements with scenario-based modelling, data modelling, flow-oriented modelling, class-based modelling and behavioural modelling

We think we're real and experience the world as though we're real.

## GOOD PRACTICES

### Good practices from other perspectives

**Requirements management** (to handle changes)

- Establish a change control process
- Perform change impact analysis
- Establish baselines and control versions of requirements sets
- Maintain change history
- Track requirements status
- Track requirements issues
- Maintain a requirements traceability matrix
- Use a requirements management tool

**Knowledge**

- Train business analysts
- Educate stakeholders about requirements
- Educate developers about application domain
- Define a requirements engineering process
- Create a glossary

*"The client kept changing the requirements on a daily basis, so we decided to freeze them until the next release."*

## GOOD PRACTICES

**From other perspectives (continue)**

**Project management**

- Select an appropriate life cycle – which one is suitable for the entire project?
- Plan requirements approach – how and when to start and continue requirements process?
- Estimate requirements effort – what HR needed?
- Project base plans on requirements – renew project base plans along with requirements
- Identify requirements decision makers – who decide?
- Renegotiate commitments – with customer when changes in requirements introduced
- Manage requirements risks – contingent plan for changing conditions/environments
- Track requirements effort – cannot spend endless efforts on requirements
- Review past lessons learned – to avoid the same mistakes

## GOOD PRACTICES

**Trade-off between value and difficulty**

- High value (value of the good practices with respect to the benefit they can bring into requirement engineering process)

| **Value** | **Difficulty** | | |
|---|---|---|---|
| | High | Medium | Low |
| **High** | ▪ Define a requirements engineering process<br>▪ Base plans on requirements<br>▪ Renegotiate commitments | ▪ Train business analysts<br>▪ Plan requirements approach<br>▪ Select product champions<br>▪ Identify user requirements<br>▪ Hold elicitation interviews<br>▪ Specify nonfunctional requirements<br>▪ Prioritize requirements<br>▪ Define vision and scope<br>▪ Establish a change control process<br>▪ Review the requirements<br>▪ Allocate requirements to subsystems<br>▪ Use a requirements management tool<br>▪ Record business rules | ▪ Educate developers about application domain<br>▪ Adopt requirement document templates<br>▪ Identify user classes<br>▪ Model the application environment<br>▪ Identify requirement origins<br>▪ Establish baselines and control versions of requirements sets<br>▪ Identify requirements decision makers |

# GOOD PRACTICES

- Medium value

| Value | Difficulty | | |
|---|---|---|---|
| | High | Medium | Low |
| **Medium** | • Maintain a requirements traceability matrix<br>• Hold facilitated elicitation workshops<br>• Estimate requirements effort<br>• Reuse existing requirements | • Educate stakeholders about requirements<br>• Conduct focus groups<br>• Create prototypes<br>• Analyze feasibility<br>• Define acceptance criteria<br>• Model the requirements<br>• Analyze interfaces<br>• Perform change impact analysis<br>• Select an appropriate life cycle<br>• Identify system events and responses<br>• Manage requirements risks<br>• Review past lessons learned<br>• Track requirements effort | • Create a data dictionary<br>• Observe users performing their jobs<br>• Test the requirements<br>• Track requirements status<br>• Perform document analysis<br>• Track requirements issues<br>• Uniquely label each requirement<br>• Create a glossary |

## GOOD PRACTICES

- Low value

| **Value** | **Difficulty** | | |
|---|---|---|---|
| | High | Medium | Low |
| **Low** | | • Distribute questionnaires<br>• Maintain change history<br>• Simulate the requirements | • Examine problem reports |

- In summary, a balance between value and difficulty is needed in overall project period

- but at certain stages
  the focus can be on either sides.

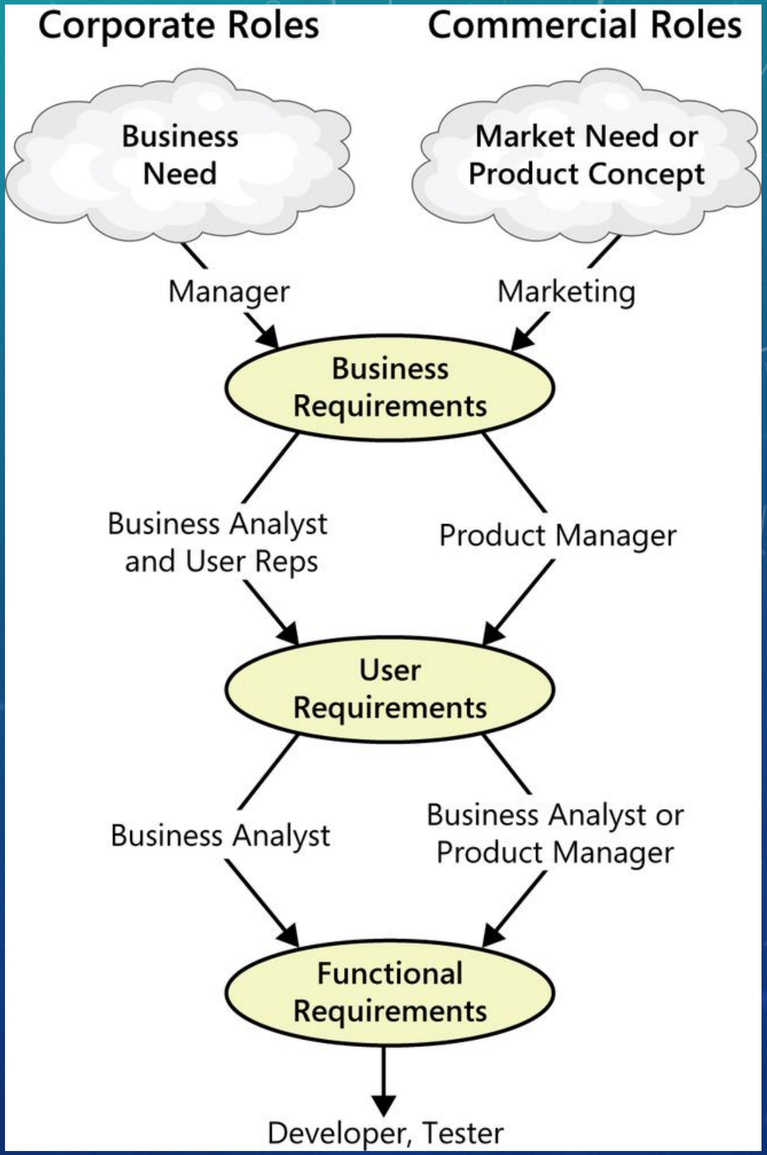
## PART FOUR: WHO (CH4)

## WHOSE JOB?

**The ways to participate**

How various stakeholders might participate in eliciting the three levels of requirements?

- Manager (corporate roles) to produce business needs
- Marketing (commercial roles) to identify market opportunities
- Business analysts (BAs) and User representatives (Reps) are the key personnel to convert business requirements to user requirements
- Project manager is supposed
  - To work with user requirement, and
  - To produce functional requirement (BA and project manager)
- Developers and testers are to carry on in the line to work out system specifications

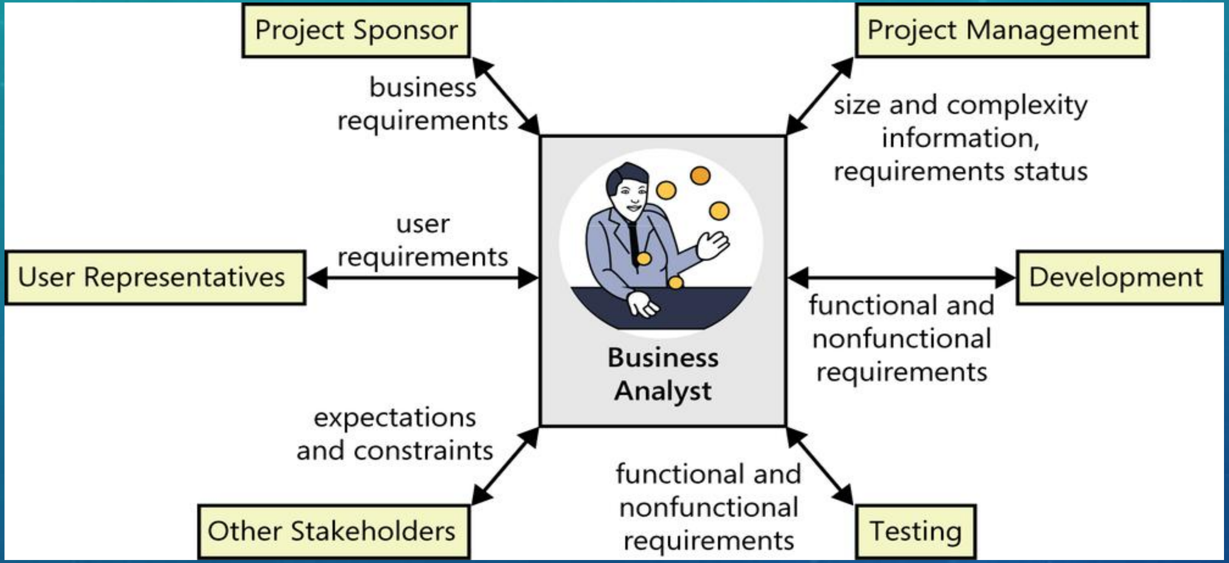


## WHOSE JOB?

**Business analyst (BA)**

**Role**

- Responsibility to elicit, analyse, document, and validate the needs of the project stakeholders
- Principal interpreter between customer community and the software development team


Project Sponsor
business requirements
Project Management
size and complexity information, requirements status
User Representatives
user requirements
Business Analyst
Development
functional and nonfunctional requirements
Other Stakeholders
expectations and constraints
functional and nonfunctional requirements
Testing

## WHOSE JOB?

Difference between planning and scheduling:
Planning is make a plan formed by a sequence of actions
Scheduling is to make a schedule formed with a sequence of action and associated with time.

**Business analyst’s tasks**

**Comprehensively understand applications and pass on the understanding to developers**

- Define business requirements – by BA himself or through BA to find some other people such as sponsor, product manager, or marketing manager, BA might suggest a template for a vision and scope document
- Plan the requirements approach – plans to elicit, analyse, document, validate, and manage requirements throughout the project, BA works with project manager to ensure the plan align with overall project plan (Should this be scheduling rather than planning or not?)
- Identify project stakeholders and user classes – work with the business sponsors to select appropriate representatives for each user class, explain what BA would like to have from them and agree on an appropriate level of engagement from each one
- Elicit requirement – by using a variety of information-gathering techniques
- Analyse requirements – derived requirements as the logical consequence of what the customers requested by using requirements models to recognize patterns, to identify gaps in the requirements, to reveal conflicting requirements, and to confirm that all requirements scope

## WHOSE JOB?

**BA's tasks (continue)**

- Document requirements – writing down requirements "to clearly describes the solution that will address the customer's problem" (really?) by using standard templates
- Communicate requirements – represent requirements by using methods other than text to ensure all parties to understand the information you are communicating (Key in communication is to let others to accept your idea rather than just to understand you.)
- Lead requirements validation -- ensuring all requirements are feasible, in scope, legal, etc. and setting up acceptance criteria for testing possible solutions
- Facilitate requirements prioritisation -- broker collaboration and negotiation among various stakeholders and the developers to ensure that they make sensible priority decisions
- Management requirements -- track the status of those requirements, verify their satisfaction in the product, and manage changes to the requirements baseline (meaning essential and approved-by-all requirements)

## WHOSE JOB?

Thinking on your feet – you got an idea and you make a move based on the idea very quickly. Here, it means that you spot uncertainty quickly and take action to clarify it quickly.

**Essential skills a BA needs**

- Listening -- Active listening involves eliminating distractions, maintaining an attentive posture and eye contact, and restating key points to confirm your understanding
- Interviewing and questioning -- interact with diverse individuals and groups and ask the right questions
- Thinking on your feet -- spot contradictions, uncertainty, vagueness, and assumptions so the issues can be discussed in the moment when they are discovered (uncertainties = vagueness, ambiguity, inaccuracy, un-specification, etc.)
- Analytical -- think at both high and low levels of abstraction and knows when to move from one to another, i.e. specifying and generalising
- System thinking -- see the big picture
- Learning – life-long learner for new material quickly to develop background knowledge to carry on requirement engineering (background, really? Shall it be background of customer?)
- Facilitation – initiate and lead discussion

## WHOSE JOB?

**Essential skills a BA needs (continue)**

- Leadership -- influence a group of stakeholders to move in a certain direction to accomplish a common goal based on the knowledge of various techniques
- Observational -- watching a user perform her job or use a current application helps to detect subtleties that the user might not think to mention
- Communication -- express complex ideas clearly, both in written form and verbally for multiple audiences, including customers who have to validate the requirements and developers who need clear, precise requirements for implementation
- Organisational -- to set up an information architecture (classes, categories, etc) to support the project information as it grows throughout the project
- Modelling -- structured analysis models (data flow diagram, entity-relationship diagram, and similar diagrams, Unified Modelling Language (UML) notations, etc
- Interpersonal -- get people with competing interests to work together as a team and speak the language of the audience she is talking to, not using technical teams with business stakeholders
- Creativity – may offer new ideas (not solutions)

## WHOSE JOB?

**Essential analyst knowledge**

Business analysts need a breadth of knowledge, much of which is gained through experience.

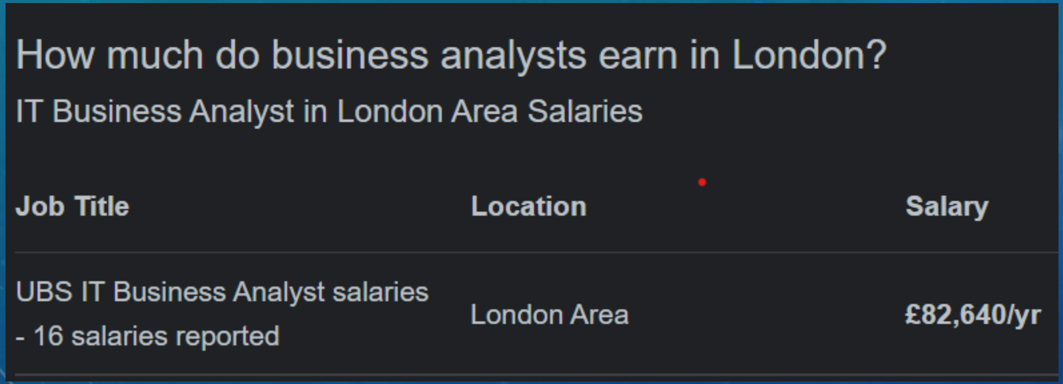
- understand contemporary requirements engineering practices and how to apply them in the context of various software development life cycles
- sound understanding of project management, development life cycles, risk management, and quality engineering
- product management concepts such as new product development, business justification, planning, verification, forecasting, pricing, product launch, and marketing of a product
- basic level of knowledge about the architecture and operating environment, so that they can engage in technical conversations about priorities and non-functional requirements.
- business, the industry, and the organization are powerful assets -- applications' backgrounds

## WHOSE JOB?

Who are suitable to be a business analyst?

- The former user
- The former developer or tester
- The former (or concurrent) project manager
- The subject matter expert
- The rookie (new recruit)

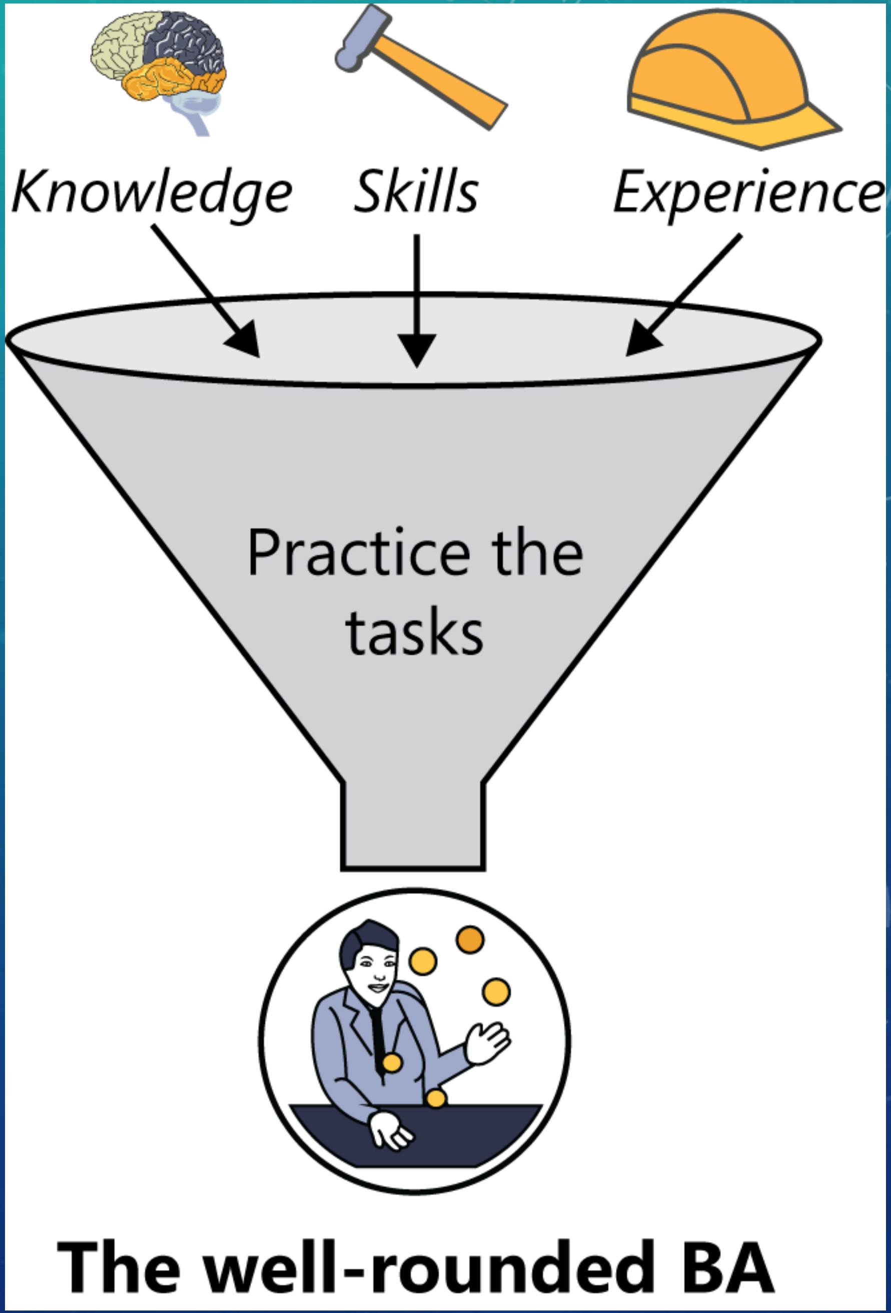
It is a higher ranking and rewarding job



How much do business analysts earn in London?

IT Business Analyst in London Area Salaries

| Job Title | Location | Salary |
|---|---|---|
| UBS IT Business Analyst salaries - 16 salaries reported | London Area | £82,640/yr |



## PART FIVE: CASE STUDIES

## 1. THE NEED FOR SOFTWARE REQUIREMENTS

*"Hello, Phil? This is Maria in Human Resources. We're having a problem with the personnel system you programmed for us. An employee just changed her name to Sparkle Starlight, and we can't get the system to accept the name change. Can you help?"*
*"She married some guy named Starlight?"*
*"No, she didn't get married, just changed her name," Maria replied. "That's the problem. It looks like we can change a name only if someone's marital status changes."*
*"Well, yeah, I never thought someone might just change her name. I don't remember you telling me about this possibility when we talked about the system," Phil said.*
*"I assumed you knew that people could legally change their name anytime they like," responded Maria. "We have to straighten this out by Friday or Sparkle won't be able to cash her paycheck. Can you fix the bug by then?"*
*"It's not a bug!" Phil retorted. "I never knew you needed this capability. I'm busy on the new performance evaluation system. I can probably fix it by the end of the month, but not by Friday. Sorry about that. Next time, tell me these things earlier and please write them down."*
*"What am I supposed to tell Sparkle?" demanded Maria. "She'll be upset if she can't cash her check."*
*"Hey, Maria, it's not my fault," Phil protested. "If you'd told me in the first place that you had to be able to change someone's name at any time, this wouldn't have happened. You can't blame me for not reading your mind."*
*Angry and resigned, Maria snapped, "Yeah, well, this is the kind of thing that makes me hate computers. Call me as soon as you get it fixed, will you?"*

## 2. CUSTOMERS’ THOUGHTS ABOUT SOFTWARE REQUIREMENTS

*Gerhard, a senior manager at Contoso Pharmaceuticals, was meeting with Cynthia, the manager of Contoso’s IT department. “We need to build a chemical tracking information system,” Gerhard began. “The system should keep track of all the chemical containers we already have in the stockroom and in laboratories. That way, the chemists can get some chemicals from someone down the hall instead of always buying a new container. This should save us a lot of money. Also, the Health and Safety Department needs to generate government reports on chemical usage and disposal with a lot less work than it takes them today. Can you build this system in time for the compliance audit in five months?”*

*“I see why this project is important, Gerhard,” said Cynthia. “But before I can commit to a schedule, we’ll need to understand the requirements for the chemical tracking system.”*

*Gerhard was confused. “What do you mean? I just told you my requirements.”*

*“Actually, you described some general business objectives for the project,” Cynthia explained. “That doesn’t give me enough information to know what software to build or how long it might take. I’d like to have one of our business analysts work with some users to understand their needs for the system.”*

*“The chemists are busy people,” Gerhard protested. “They don’t have time to nail down every detail before you can start programming. Can’t your people figure out what to build?”*

*Cynthia replied, “If we just make our best guess at what the users need to do with the system, we can’t do a good job. We’re software developers, not chemists. I’ve learned that if we don’t take the time to understand the problem, nobody is happy with the results.”*

*“We don’t have time for all that,” Gerhard insisted. “I gave you my requirements. Now just build the system, please. Keep me posted on your progress.”*

# 3. SOFTWARE REQUIREMENTS PROCESS

*"Welcome to the group, Sarah," said the project manager, Kristin. "We're looking forward to having you help us with the requirements for this project. I understand that you were a business analyst in your previous job. Do you have some idea of how we should get started here?"*

*"Well," Sarah replied, "I was thinking I should just interview some users and see what they want. Then I'll write up what they tell me. That should give the developers a good place to start. That's mostly what we did before. Do you know some users I could talk to?"*

*"Hmmm. Do you think that will be good enough for this type of project?" Kristin asked. "We tried that approach before, but it didn't work out very well. I was hoping you might have some ideas about best practices from your past BA experiences that might be better than just interviewing a couple of users. Are there any particular techniques that you've found to be especially helpful?"*

*Sarah was rather at a loss. "I don't really know about any specific ways to approach requirements other than talking to users and trying to write clear specifications from what they say. At my last job I just did the best I could based on my business experience. Let me see what I can find out."*

## END OF LECTURE ONE