Good Practices for Requirements Engineering

Dr. Zeinab Teimoori
SENG 3130
Thompson Rivers University

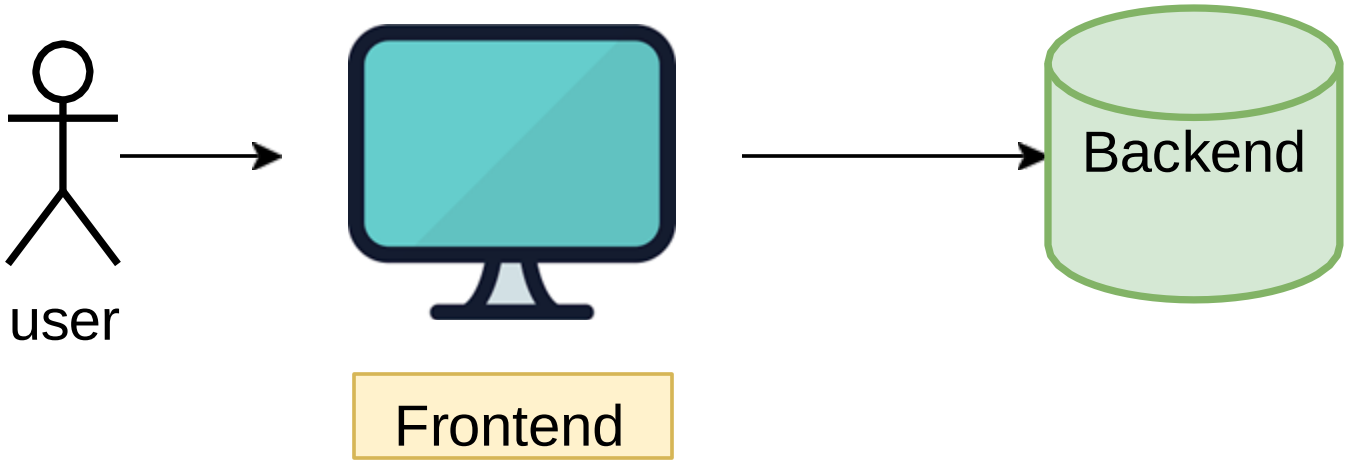
Objectives

- RE best-practices tool kit
- A requirements development process framework
- Good practices

What we have learned about requirements engineering (RE)

- RE is very important
- RE process is incremental
- We should contact the customer early and periodically to close the expectations gap
- Customers have their rights and responsibilities

Online purchasing system



What we have learned about RE

Should I conduct a **survey**?

What we have learned about RE

Should I conduct a **survey**?

Should we make a **prototype** first?

What we have learned about RE

Should I conduct a **survey**?

Should we make a **prototype** first?

Should I **interview** every stakeholder, or is it better to conduct a large brainstorm **meeting**?

What we have learned about RE

Should I conduct a **survey**?

Should we make a **prototype** first?

Do we need to identify the impacted **interfaces**?

Should I **interview** every stakeholder, or is it better to conduct a large brainstorm **meeting**?

What we have learned about RE

Should I conduct a **survey**?

Should we make a **prototype** first?

Do we need to identify the impacted **interfaces**?

Should I **interview** every stakeholder, or is it better to conduct a large brainstorm **meeting**?

Should we write the **use case** diagrams?

There is no single solution that fits all projects

- We cannot handle all projects with the same technique. Projects differ in the customers' knowledge about their needs.
- Software professionals need to acquire a tool kit of techniques (best practices) that are used to handle each project. [1]
- Software professionals apply suitable techniques based on their needs.

RE best practices tool kit

Apply the set of best-practices that is useful for your project

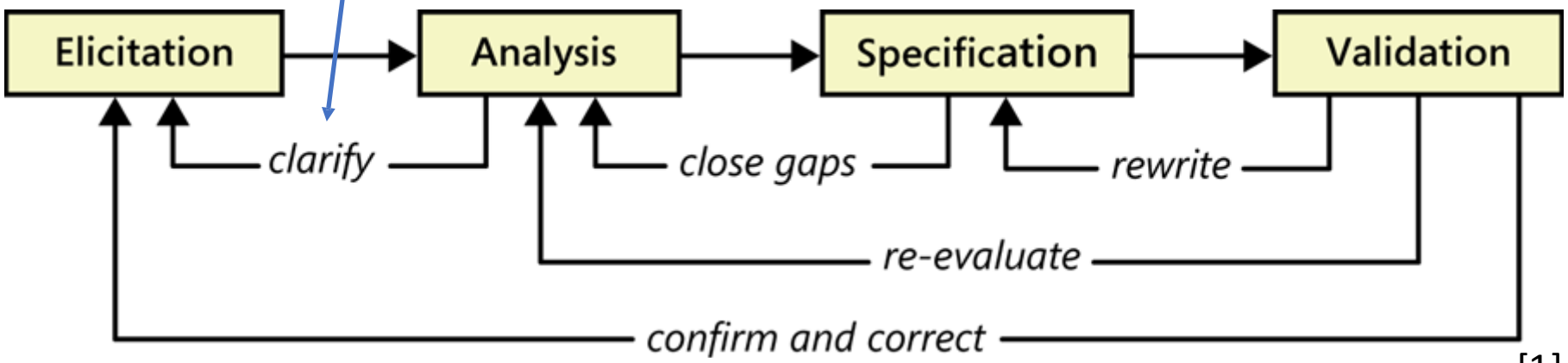
For example,
use cases are useful for *front- end* projects;
prototypes are helpful when the customer *does not have a clear idea* about her/his needs;
interface analysis is helpful for *back-end* projects.

Objectives

- RE best-practices tool kit
- **A requirements development process framework**
- Good practices

The requirements development process (RDP) framework

When analyzing the requirements, BA discovered contradicting/unclear requirements.

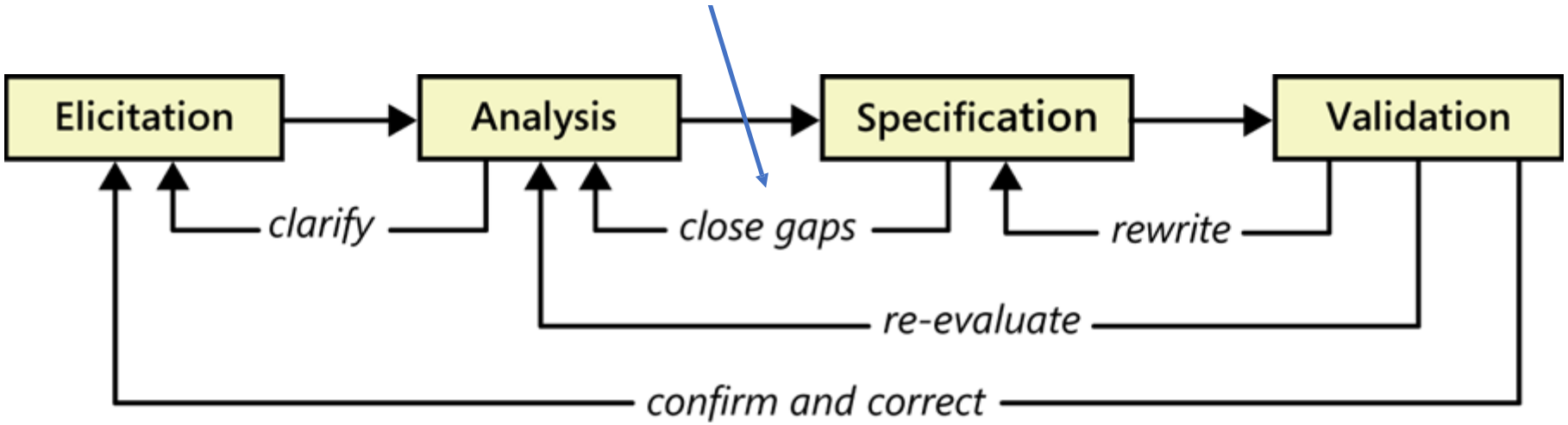


[1]

The process is an iterative process

The requirements development process (RDP) framework

Writing makes you rethink about the collected requirements. For example, when writing the requirements, BA discovered requirements that need more analysis.

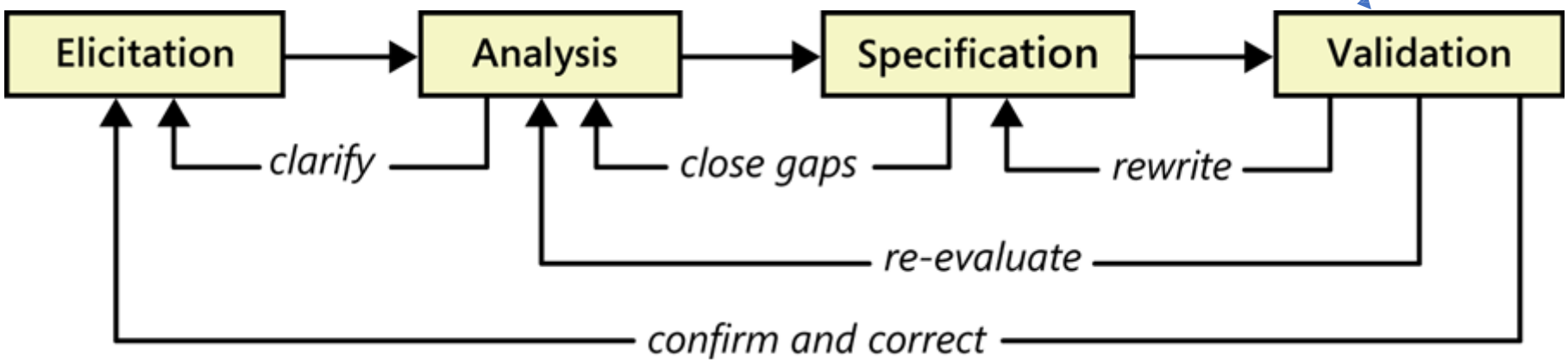


[1]

The process is an iterative process

The Requirements Development Process (RDP) framework

Validation makes BA go back to any previous step. For example, BA may discover (a) conflicting requirements (analysis), (b) missing details (specification), or (3) missing requirements/stakeholders (elicitation).

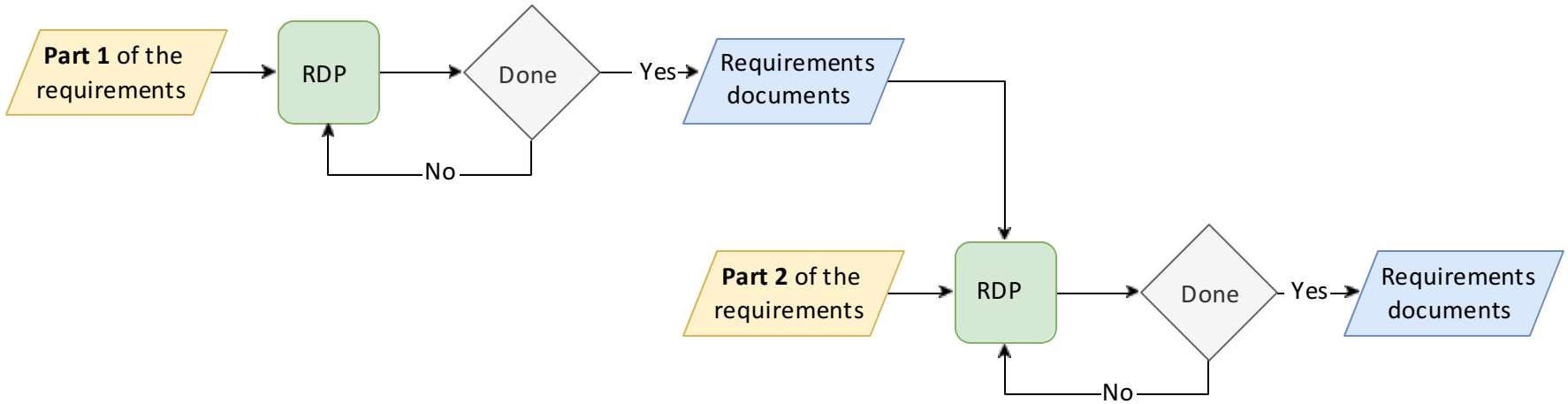


[1]

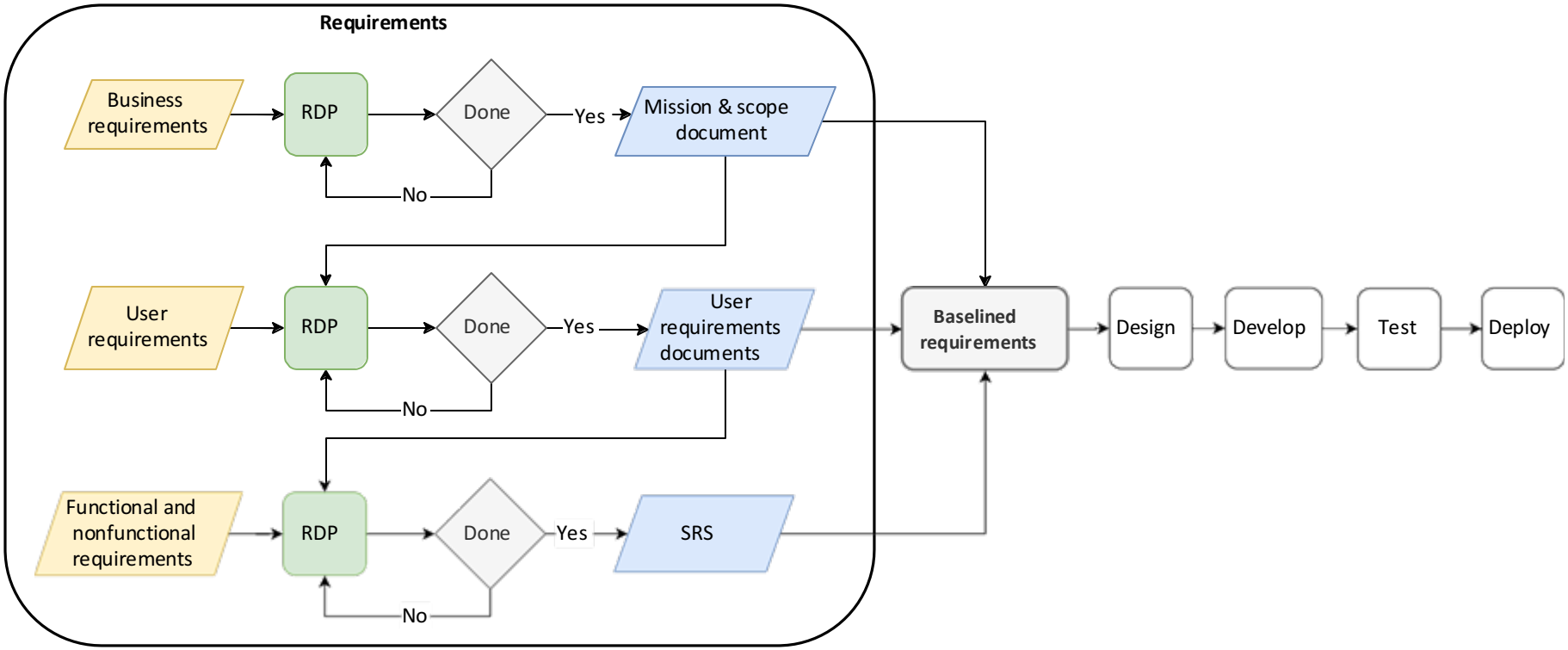
The process is an iterative process

The big picture of employing the RDP

- Start with the most important requirements (i.e., the requirements with the highest priority)



The big picture of employing the RDP (waterfall model)



The big picture of employing the RDP (waterfall model)

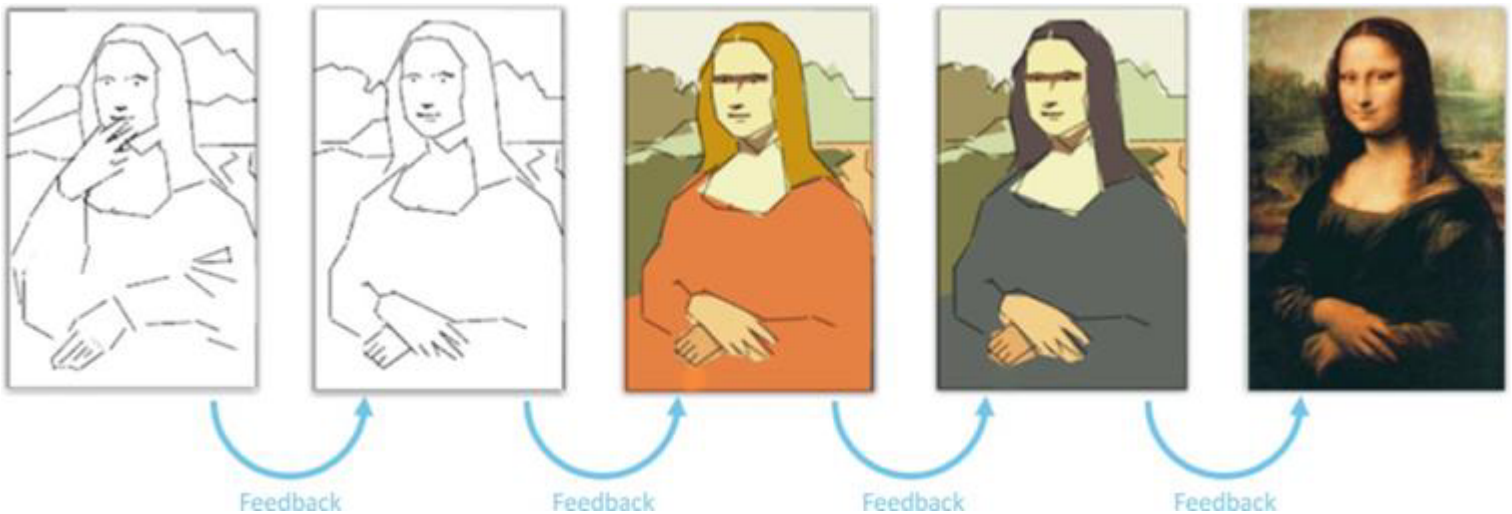
Strengths:

- Customers and developers agree on all requirements in the early phase of the project
- The sequential methodology is easy to manage

Weaknesses:

- It is not straightforward for customers to list all the needed specifications from the beginning. Consequently, the waterfall model may lead to user frustrations

The iterative model



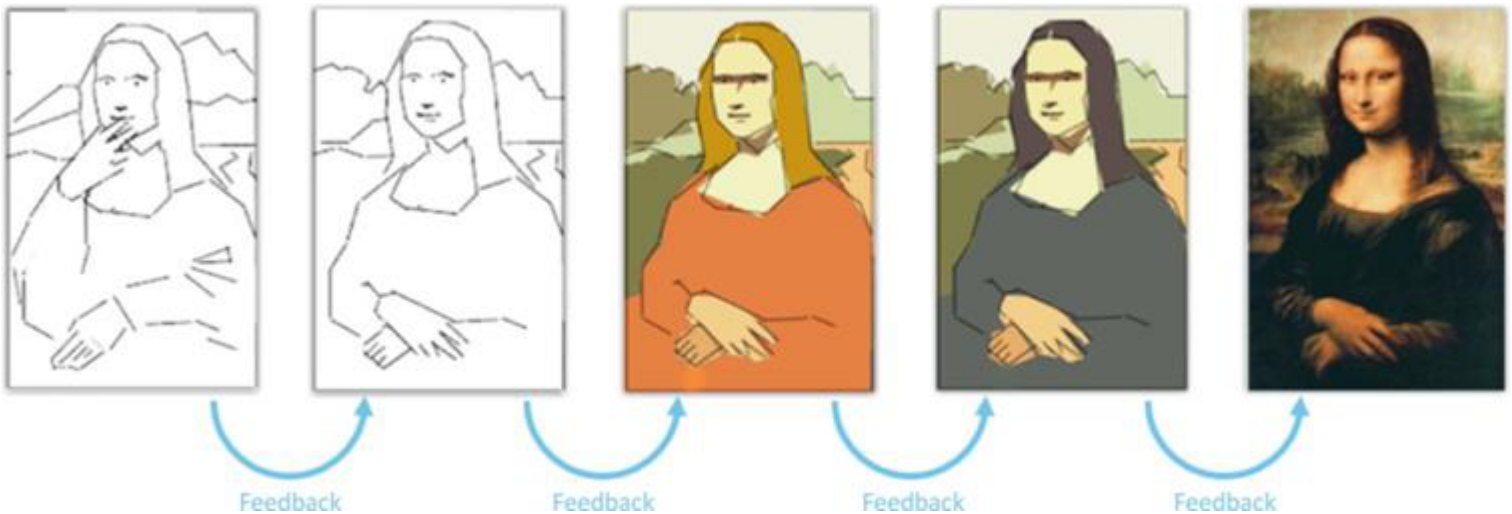
[1]

In the iterative methodology, the software system is delivered through iterations. In every iteration, a **valuable product** is delivered to the customers to test it and get the feedback. Developers learn from the feedback while working on the next iteration.

[1] https://medium.com/@bezerranaiana/enlightening-the-product-continuous-discovery-part-iv-917386caed21

The iterative model

Do you notice the difference between the **first** and the **last** iteration?

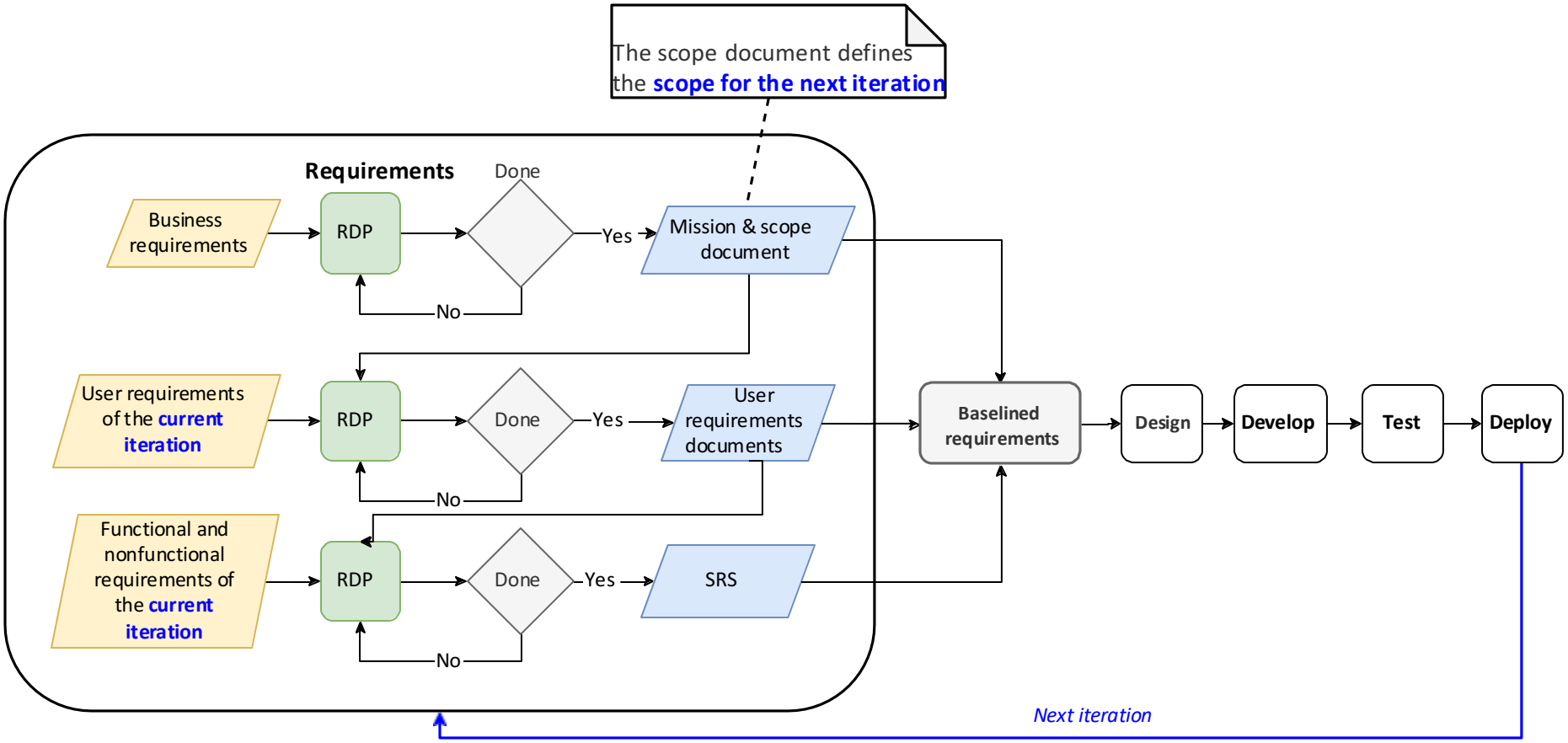


[1]

In the iterative methodology, the software system is delivered through iterations. In every iteration, a **valuable product** is delivered to the customers to test it and get the feedback. Developers learn from the feedback while working on the next iteration.

[1] https://medium.com/@bezerranaiana/enlightening-the-product-continuous-discovery-part-iv-917386caed21

The big picture of employing the RDP (iterative model)



The big picture of employing the RDP (iterative model)

Strengths:

- Customers and developers focus on the most important features[1]
- There is always a ready product

Weaknesses:

- The system needs to be designed in a way that facilitates continuous changes.

[1] https://coachjarkko.com/2013/01/25/with-agile-methods-you-get-to-focus-on-the-most-important- features-of-your-product/

Requirement Development Process - Framework

Per each iteration

Once per project

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

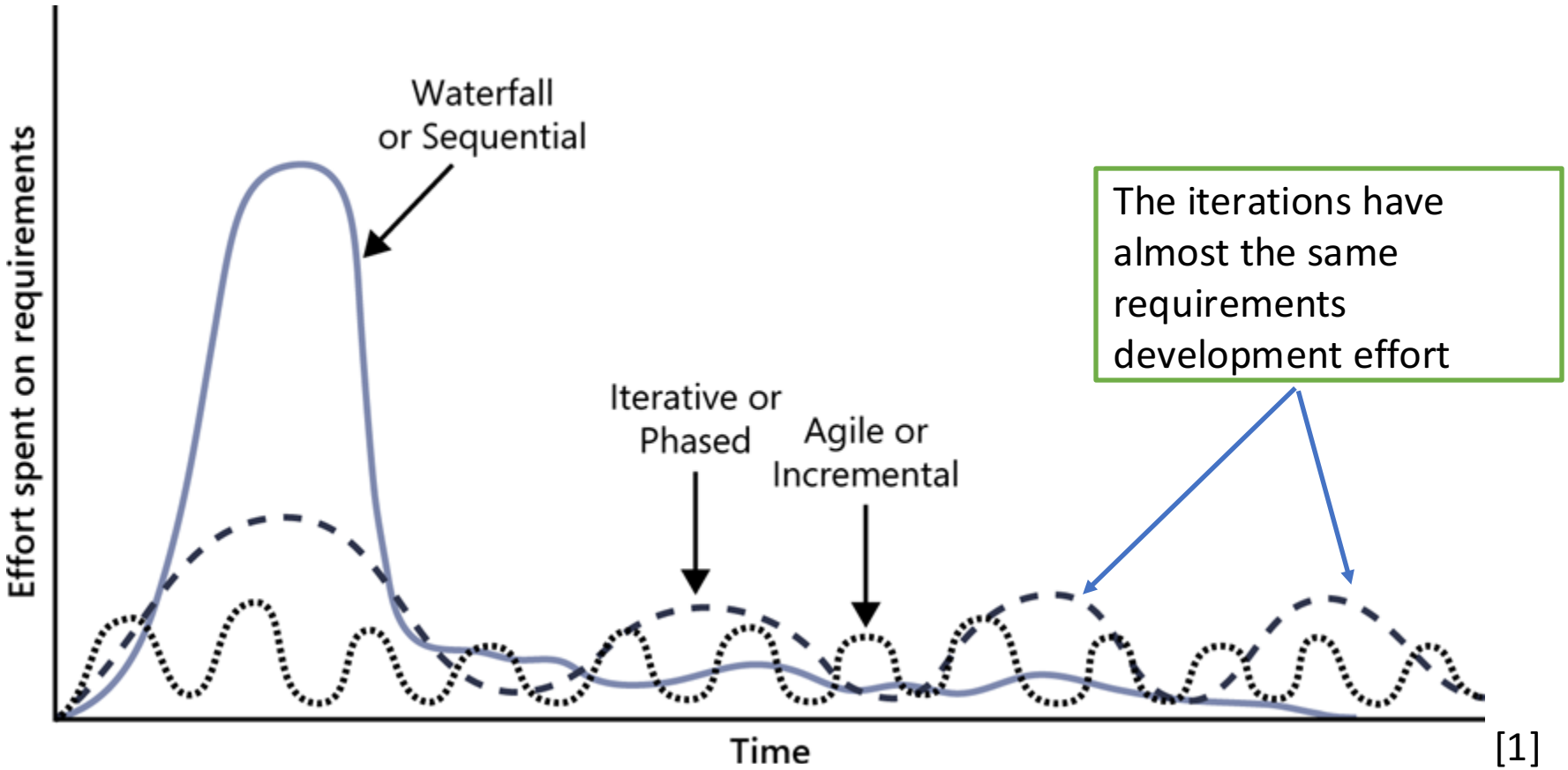
Repeat for iteration 2

Repeat for iteration 3

Repeat for iteration N

[1]

The big picture of employing the RDP



[1]

Which software development life cycle?

- Waterfall methodology can be used in:
  - Predictable, Structured Environment
  - Clear upfront requirements, minimal mid-process changes, extensive documentation, long development cycles, and strict regulatory compliance.
- Agile methodology is suitable when users do not know what they need. Thus, BA can explore and gather the full requirements through phases.
- Agile is also useful for the high-competitive market.

Objectives

- RE best-practices tool kit
- A requirements development process framework
- **Good practices**

Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

Requirement management

Knowledge

Project management

Requirements engineering good practices

Elicitation

Analysis

Specification

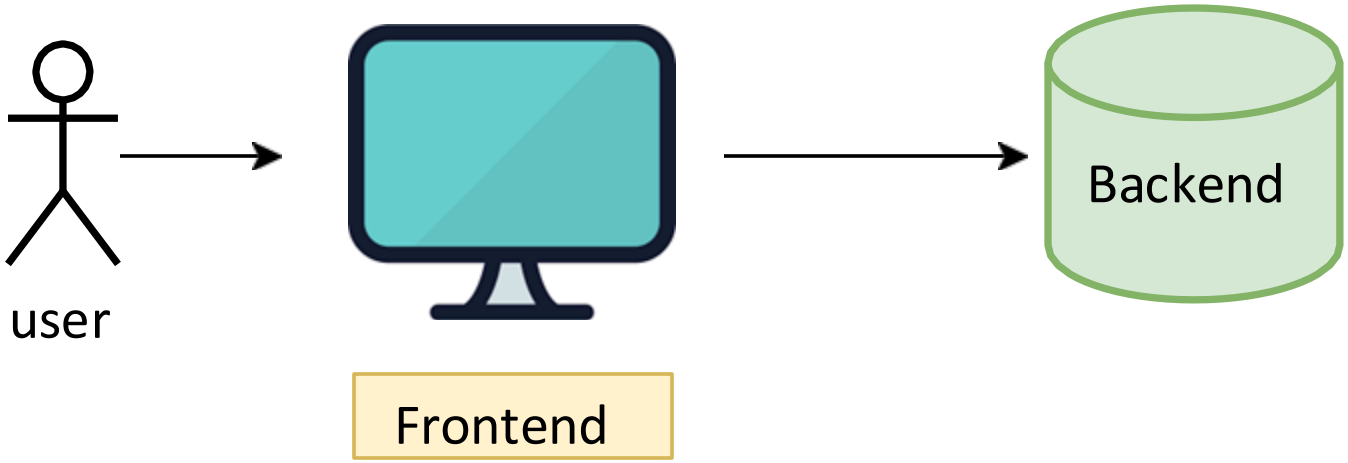
Validation

Requirement management

Knowledge

Project management

Online purchasing system

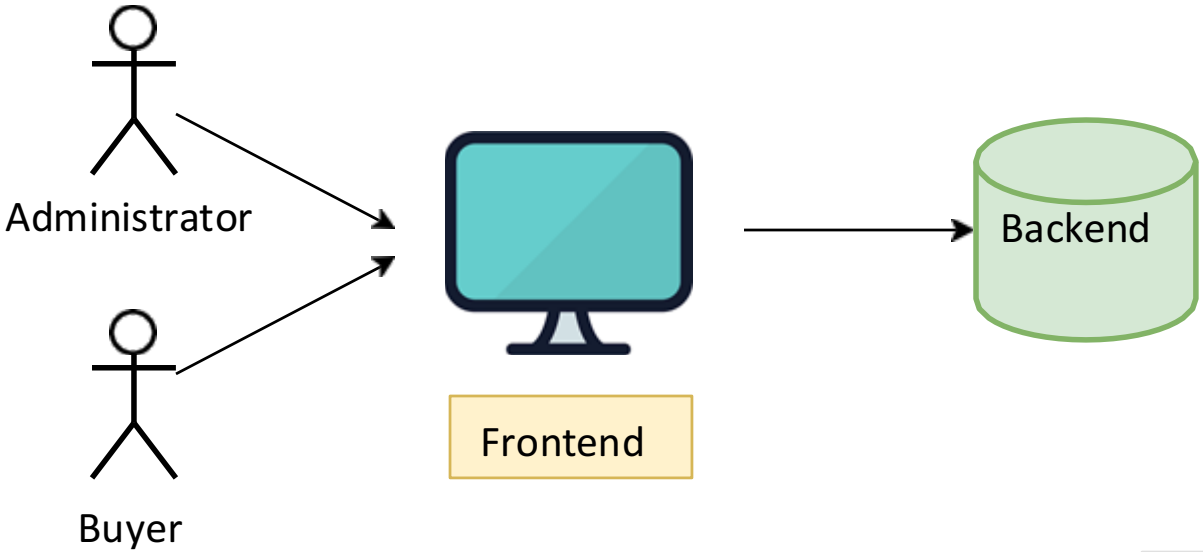


Good practices: Requirements elicitation

- Define product vision and project scope
  - *"The vision statement gives all stakeholders a common understanding of the product's outcome."* [1]
  - *"The scope defines the boundary between what's in and what's out for a specific release or iteration."* [1]
  - *"The vision should remain relatively stable throughout the project, but each planned release or iteration needs its own scope statement."* [1]

Good practices: Requirements elicitation

- Identify user classes and their characteristics
  - *"Identify the various groups of users for your product"* [1]
  - *"User classes might differ in frequency of use, features used, privilege levels, or experience (seniority level)."* [1]



Good practices: Requirements elicitation

- Select a **product champion** for each user class
  - *"Identify an individual who can accurately serve as the literal voice of the customer for each user class."* [1]
  - *"The product champion presents the needs of the user class and makes decisions on its behalf."* [1]
  - *"Product champion can be obtained from the current relationships with major customers or beta test sites"* [1]

Good practices: Requirements elicitation

- Conduct focus groups with typical users
  - *"Convene groups of representative users of your previous products or of similar products."* [1]
  - *Focus groups are helpful to identify the functional and non- functional requirements of the product.* [1]
  - *"Unlike product champions, focus groups generally do not have decision-making authority."* [1]

Good practices: Requirements elicitation

- Identify system events and responses
  - *"List the external events that the system can experience and the expected response to each event."* [1]
  - There are three classes of external events:
    - (1) signal events
    - (2) temporal or time-based events
    - (3) business events.

Good practices: Requirements elicitation

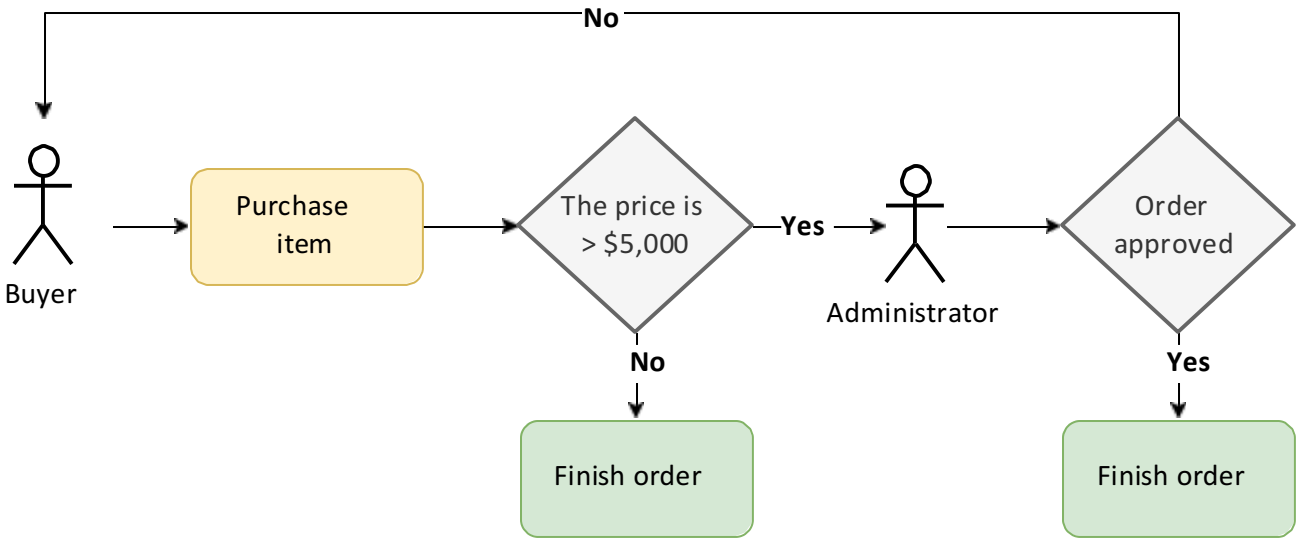
- Hold elicitation interviews
  - Interviews can be as a one-on-one meeting or with a small group of stakeholders.[1]
  - Interviews are useful to discuss the specific requirements that are important to the interviewed stakeholders.[1]
- Hold facilitated elicitation workshops
  - *Workshops are useful when you want to invite different stakeholders to resolve any conflict.* [1]
  - *Called Joint Application Design, or JAD, sessions*

Good practices: Requirements elicitation

- Observe users performing their jobs
  - Observing users, while using a prototype of the system, can quickly spot essential observations such as:
    - The most frequently used scenarios.
    - How easy our system is? For example, is the UI simple, or do users need to do many steps (e.g., clicks) to finish one scenario?

Good practices: Requirements elicitation

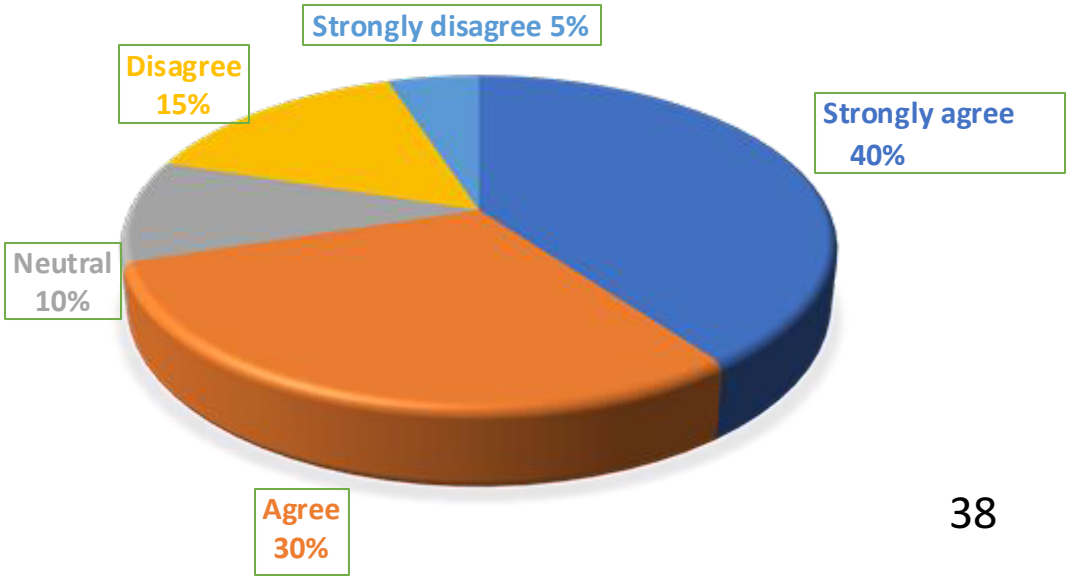
- Observe users performing their jobs
  - The process flow diagram can help BA identify the different user classes and how they interact.



Good practices: Requirements elicitation

- Distribute questionnaires
  - Surveys are a good option when you do not have champions.
  - Submitting surveys can help BA get feedback from many users and perform more in-depth analytics to the collected information.

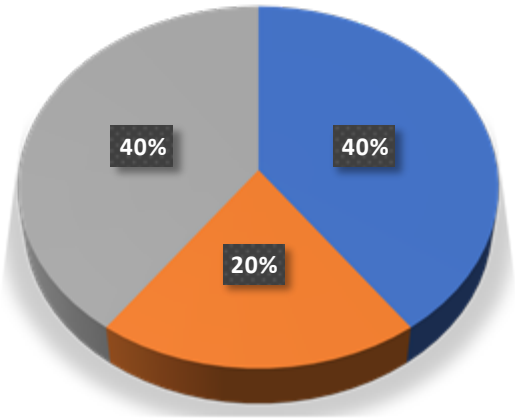
The system is useful and helps me purchase new items.



Good practices: Requirements elicitation

- Distribute questionnaires

Why do you see that our system is not useful?

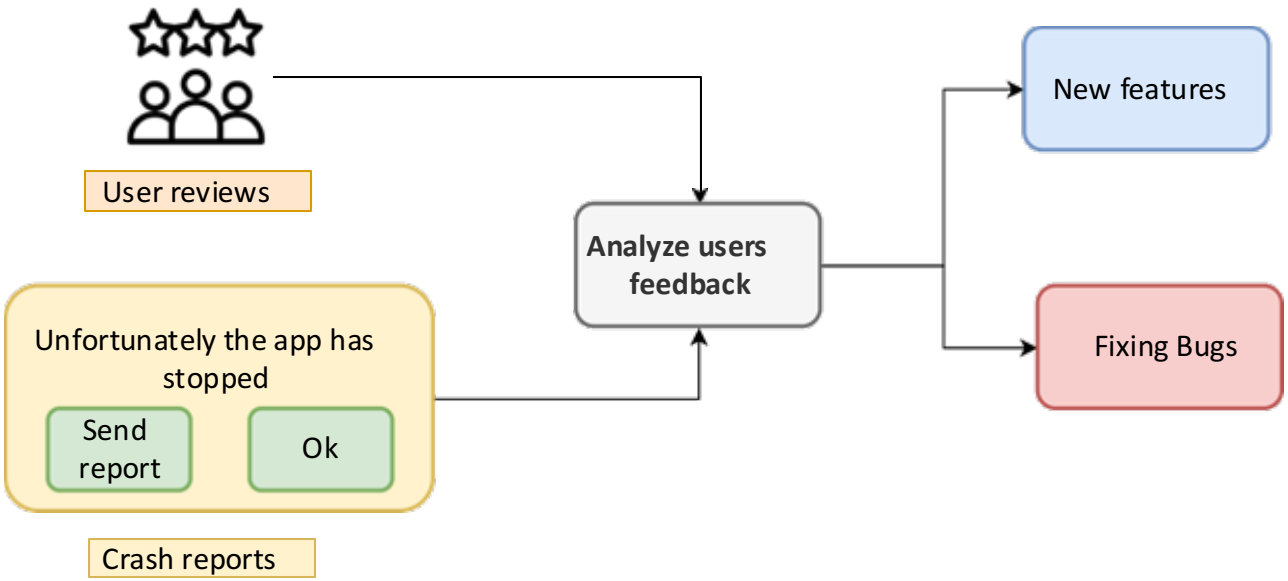


- I need many steps to purchase an item.
- The approval process takes a very long time.
- I cannot purchase many items in the same order.

Good practices: Requirements elicitation

- Examine problem reports of current systems for requirement ideas

  Bug reports and feature requests can help BA plan for the next releases



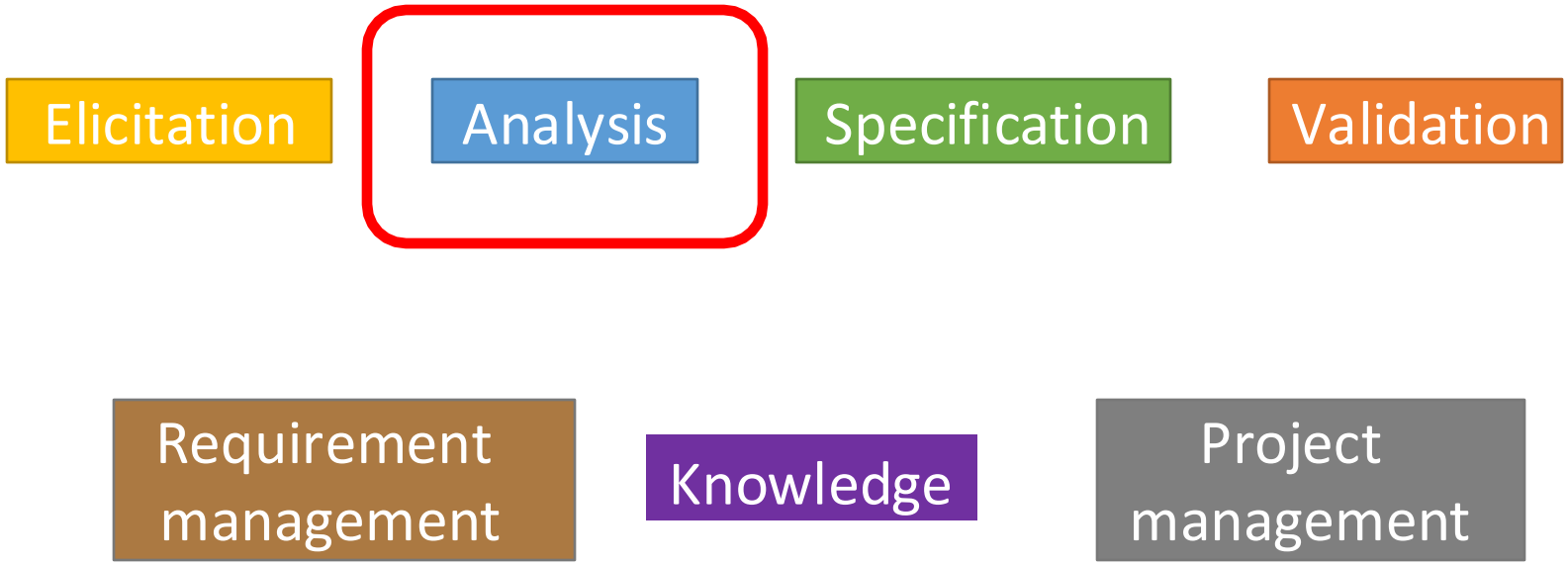
Good practices: Requirements elicitation

- Reuse existing requirements

  Reusing prior requirements can help BA save time in gathering requirements, as BA does not need to do things from scratch.

  - BA can reuse the commonly used requirements for authentication and security.
  - BA can also reuse the definition of the user classes from previous projects.

Requirements engineering good practices



Good practices: Requirements Analysis

- Requirements analysis includes different activities such as:
  - Identifying contradicting or missing requirements.
  - Assessing the feasibility of the requested features.
  - Prioritizing the requirements.
  - Representing the same requirement in multiple formats, each format is suitable for a specific group of stakeholders.

Good practices: Requirements Analysis

- Model the application environment
  - The context diagram represent the external entities that interact directly with our system.
  - An ecosystem map shows the systems that interacts directly/indirectly with our system.

Good practices: Requirements Analysis

- Create user interface and technical prototypes
  - Prototypes are useful when the customer is not sure about the final product characteristics.
  - Prototypes can be used as an initial (beta) release to get users' feedback.
  - Prototypes can help users better understand their needs.

Good practices: Requirements Analysis

- Analyze requirement feasibility
  - *"The BA should work with developers to evaluate the feasibility of implementing each requirement at acceptable cost(1) and performance(2) in the intended operating environment(3)."* [1]
  - Feasibility analysis includes identifying the impacted systems (e.g., the affected components when implementing a new change).

Good practices: Requirements Analysis

- Analyze requirement feasibility

The system should be able to finish processing each order within 2 seconds.

The new requirement adds extra dependencies to the backend systems, such as the billing system, the shipping system, and the suppliers.

The added dependencies make our online shopping system tightly coupled (depending on) the availability time and the integrated backend systems' performance.

Good practices: Requirements Analysis

- Prioritize the requirements
  - Do an initial prioritization of the identified requirements.
  - Analyze user's feedback after every release and identifies the new requirements.
  - Prioritize the requirements so that we may drop old features (or putting them at the end of the priorities list).

Customers may have initial expectations that certain features will be the most important ones. After trying the product's initial prototype, users discover new features with higher priority than the existing ones. Besides, users may also reduce the preference for current features.

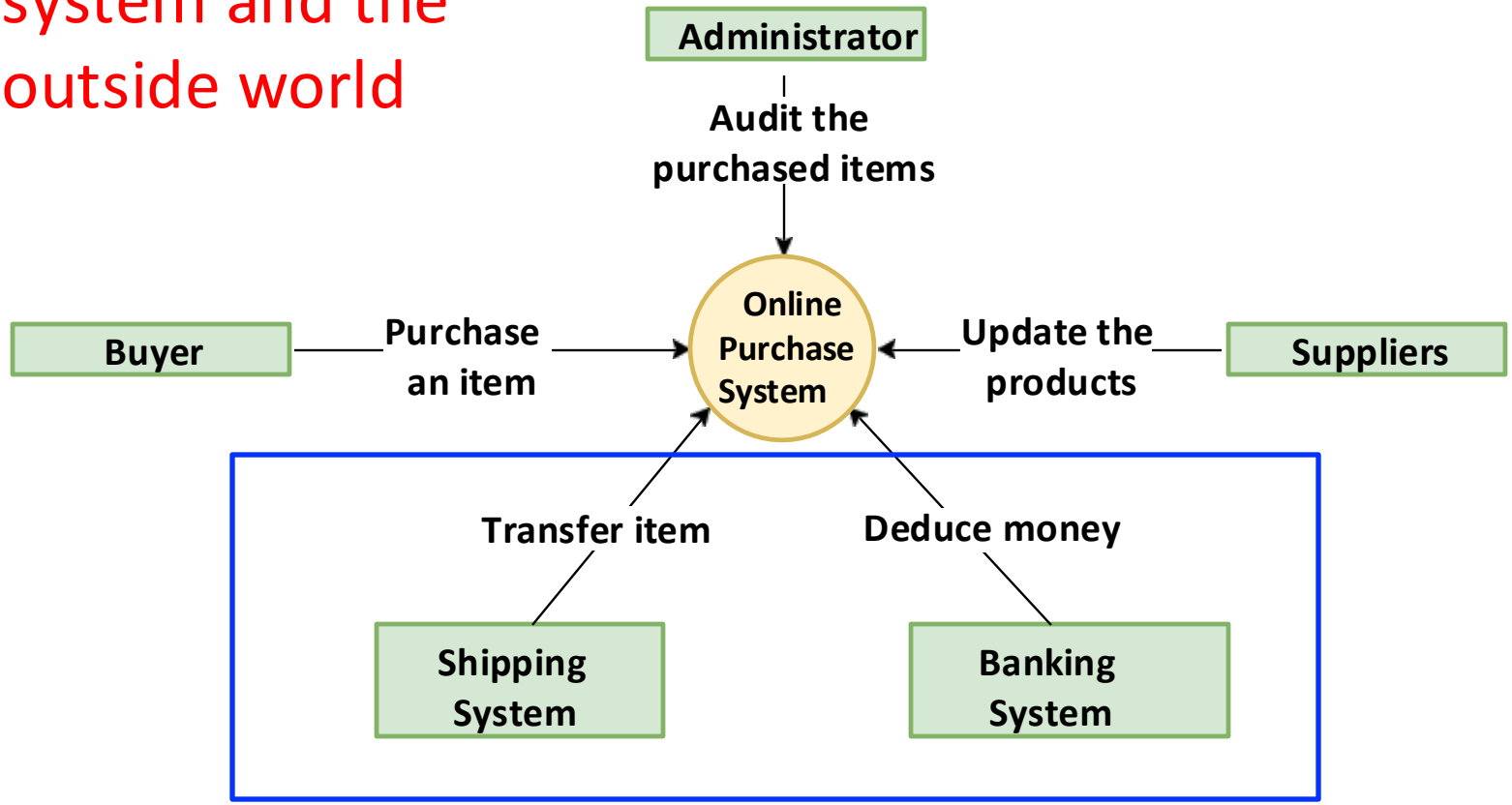
Good practices: Requirements Analysis

- Create a data dictionary
  - *"Definitions of the data items and structures associated with the system reside in the data dictionary.*
  - *This enables everyone working on the project to use consistent data definitions."* [1]

For example, if we say that our system receives a purchase order, processes and validates all the received orders. We need to define the purchase order in the data dictionary.

Good practices: Requirements Analysis

- Analyze interfaces between your system and the outside world



Good practices: Requirements Analysis

- Allocate requirements to subsystems
  - Represent every subsystem as a card and distribute the identified requirements across the identified subsystems.

Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

Requirement management

Knowledge

Project management

Good practices: Requirements Specification

- Adopt requirement document templates
  - Start with existing templates, and then tailor the template structure to be suitable for your project.

Good practices: Requirements Specification

- Identify requirement origins
  - Track every requirement to the original sources (e.g., meetings) when the requirement is introduced.

Version 1

Requirement	Origin	Impacted systems
Users shall be able to order an item by submitting a purchase order.	Meeting 1 on March 22nd 2023	Price (quota) validation system.

Version 2

Requirement	Origin	Impacted systems
Users shall be able to order multiple items by submitting a purchase order.	Meeting 4 on June 1st 2023	Price (quota) validation system.

Good practices: Requirements Specification

- Uniquely label each requirement
  - Define a convention that provides a unique identifier for each item.
  - *"The convention must be robust enough to withstand additions, deletions, and changes made in the requirements over time. Labeling the requirements permits requirements traceability and the recording of changes made."* [1]

Good practices: Requirements specification

- Uniquely label each requirement
  - To trace the origin of the requirement, we can follow this convention:
    - BR for Business requirements
    - UR for User requirements
    - FR for functional requirements
    - NR for non-functional requirements
  - To track the status of the requirement, we can follow this convention:
    - I means the initial stated requirement
    - N means newly added requirement
    - D means deleted requirement
    - U means updated requirement

ID	Requirement	Origin	Impacted systems
UR_U_13	Users shall be able to order multiple items by submitting a purchase order.	Meeting 4 on June 1st, 2023	Price (quota) validation system.

Good practices: Requirements Specification

- Record business rules
  - *"Business rules include corporate policies, government regulations, and standards."* [1]
  - Write the business rules in separate documents than the project requirements.
  - Treat business rules as an enterprise-level asset, not a project-level asset.
  - Track the impact of the mandated regulations reflected on the identified functional/non-functional requirements.

Good practices: Requirements Specification

- Specify non-functional requirements
  - *"It's possible to implement a solution that does exactly what it's supposed to do but does not satisfy the users' quality expectations."* [1]
  - BA needs to define all other factors that make the product successful.
  - Performance, Reliability, Usability, Modifiability

Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

Requirement management

Knowledge

Project management

Good practices: Requirements Validation

- Review the requirements
  - Construct a small group reviewers to make a peer review of requirements.

- Test the requirements and define acceptance criteria

"Writing tests requires you to think about how to tell if the expected functionality was correctly implemented." [1]

Good practices: Requirements Validation

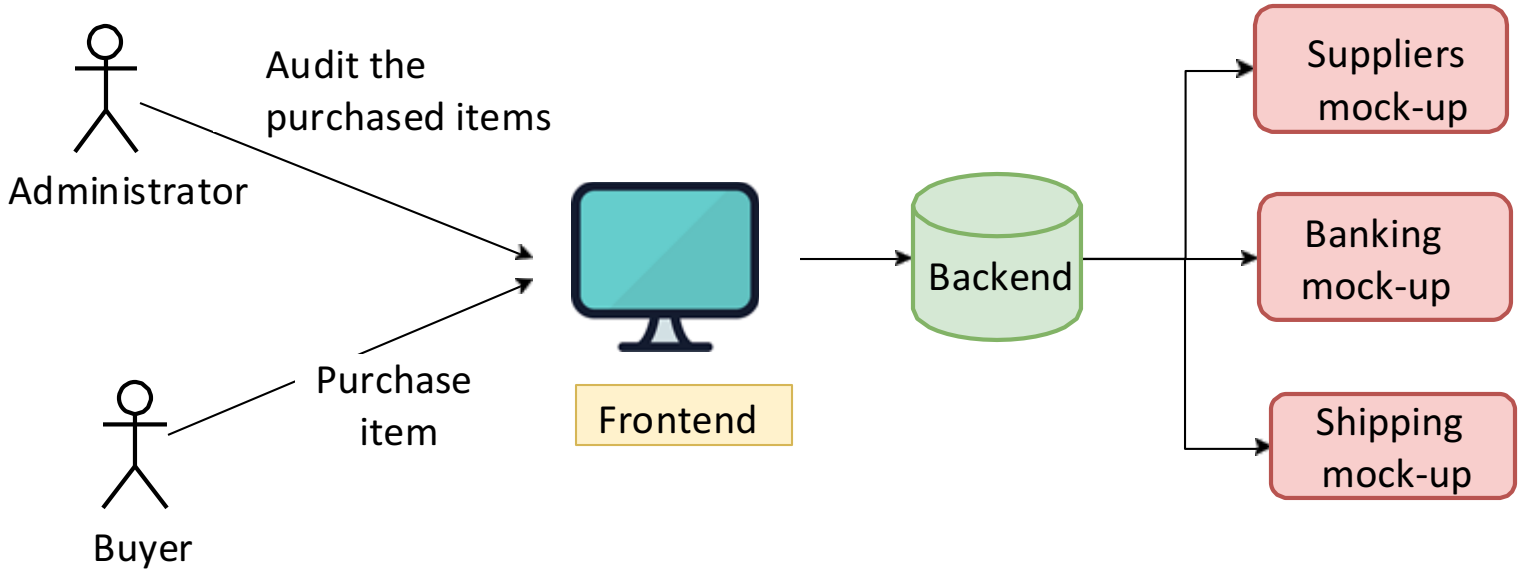
- Test the requirements and define acceptance criteria

ID	Requirement	Origin	Impacted systems	Test case ID	Test scenario
UR_U_13	Users shall be able to order multiple items by submitting a purchase order.	Meeting 4 on June 1st, 2023	Price (quota) validation system.	TC_01	A user (buyer) will log in to the system, select an item, and submit the purchase order. The system will fulfill the order and send a notification email to the buyer.

Define the acceptance criteria

Good practices: Requirements validation

- Simulate the requirements
  - Build mock-up (or simulated systems) so users can get a quick idea about the overall (end-to-end) scenario.



Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

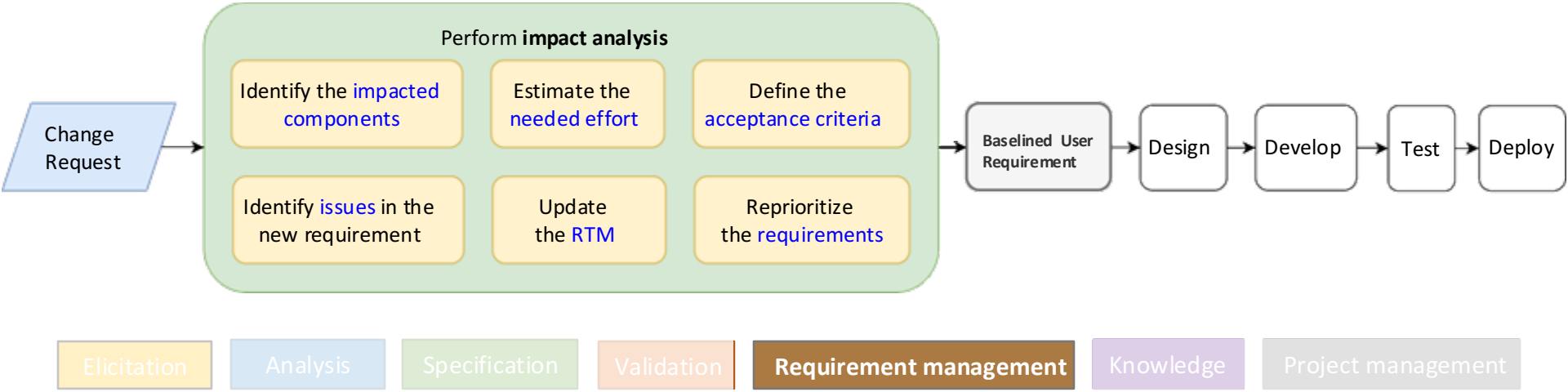
Requirement management

Knowledge

Project management

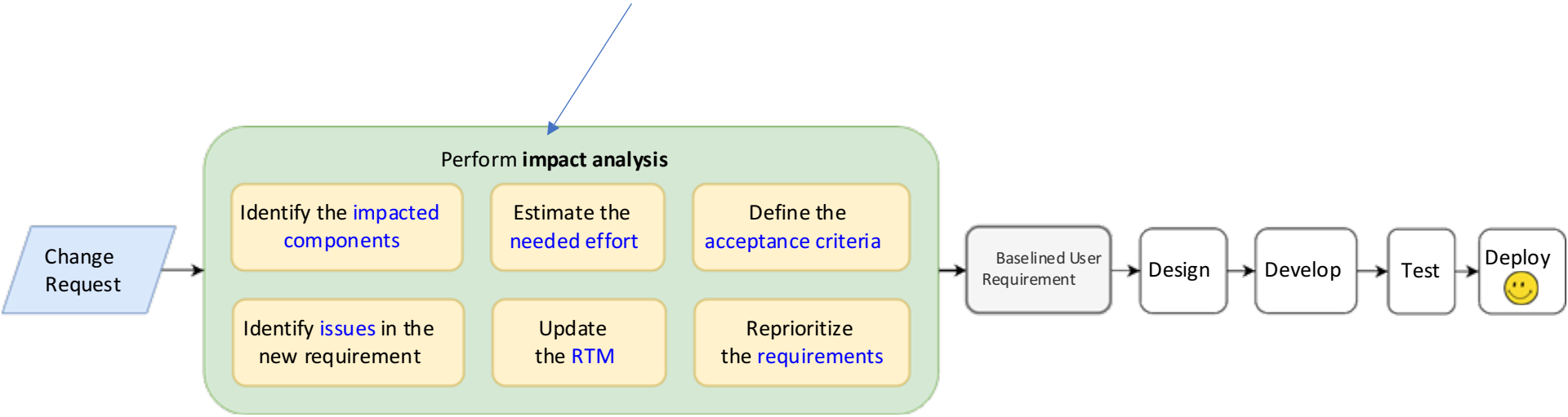
Good practices: Requirements Management

- Establish a requirements change control process
  - Rather than rejecting modifications, define a process that handles the changes.
  - *"The change process should define how requirements changes are proposed, analyzed, and resolved."* [1]
  - Charter a small group of *project stakeholders* as a *change control board (CCB)*



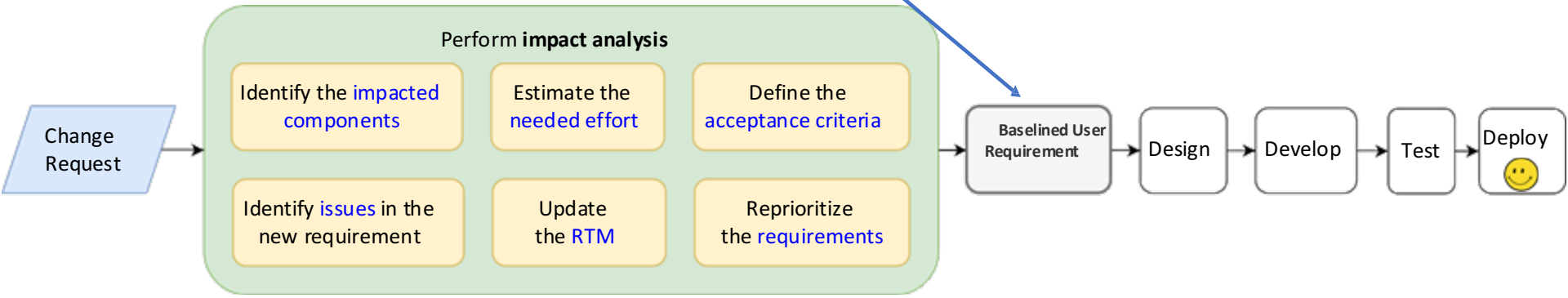
Good practices: Requirements Management

- Perform impact analysis on requirements changes



Good practices: Requirements Management

- Establish baselines and control versions of requirements sets



A baseline defines a set of agreed-upon requirements, typically for a specific release or iteration.

Good practices: Requirements Management

- Maintain a history of requirements changes

- The history can help us identify when a certain part of the requirement is changed.

- Maintaining a history of the requirements document enables us to roll back to an old version of the requirements.

Good practices: Requirements Management

- Maintain a requirements traceability matrix
  - The requirements traceability matrix (RTM) links the requirements with different levels of details (like business requirements and user requirements) to the development activities (e.g., system design, the implemented code, and the test cases). [1]

Benefits of using RTM:

- Ensures that all requirements are implemented
- Shows the source of each detailed requirement (e.g., the link between the SRS to the business requirements)
- Shows which module/test-case that handles each requirement

Good practices: Requirements Management

- Maintain a requirements traceability matrix

BR ID	Business need	Business requirement
BR_I_1	We need to develop a system that makes the order purchasing process **faster and more efficient.**	The system needs to fulfill the purchasing process of new items.

UR ID	BR ID	User requirement
UR_I_1	BR_I_1	The user (**buyer**) shall able to **buy** new items.
UR_I_2	BR_I_1	The user (buyer) shall able to **check the status** of the ordered items.
UR_I_3	BR_I_1	The user (admin) shall able to **approve** the new orders.

Good practices: Requirements Management

- Maintain a requirements traceability matrix

SR ID	UR ID	BR ID	Software requirement
FR_I_1	UR_I_1	BR_I_1	The buyer shall be able to see the list of available items.
FR_I_2	UR_I_1	BR_I_1	The buyer shall be able to select a suitable item and the delivery location.
FR_I_3	UR_I_1	BR_I_1	The buyer shall be able to review and confirm the order.
FR_I_4	UR_I_1	BR_I_1	The buyer shall be able to receive an email with the order status and details.
NR_I_5	UR_I_1	BR_I_1	The backend system shall be able to fulfill the order within 2 seconds.

Good practices: Requirements management

- Maintain a requirements traceability

SR ID	UR ID	BR ID	Software requirement	Design diagram	Component	Test case ID
FR_I_1	UR_I_1	BR_I_1	The buyer shall be able to see the list of available items.	Buy order flowchart	Item catalogue component	TC_01
FR_I_2	UR_I_1	BR_I_1	The buyer shall be able to select a suitable item and the delivery location.	Buy order flowchart	Order component	TC_02
FR_I_3	UR_I_1	BR_I_1	The buyer shall be able to review and confirm the order.	Buy order flowchart	Order component	TC_03
FR_I_4	UR_I_1	BR_I_1	The buyer shall be able to receive an email with the order status and details.	Buy order flowchart	Notification component	TC_04
NR_I_5	UR_I_1	BR_I_1	The backend system shall be able to fulfill the order within 2 seconds.	System interfaces diagram	Order system, backend integration component	TC_05

Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

Requirement management

Knowledge

Project management

Good practices: Knowledge

- Train BAs in RE
- Customers should educate BA about their business.
- Document the business terms into a glossary

Requirements engineering good practices

Elicitation

Analysis

Specification

Validation

Requirement management

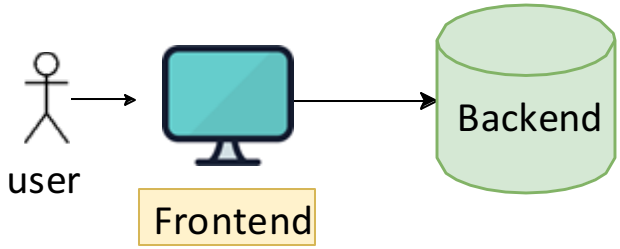
Knowledge

Project management

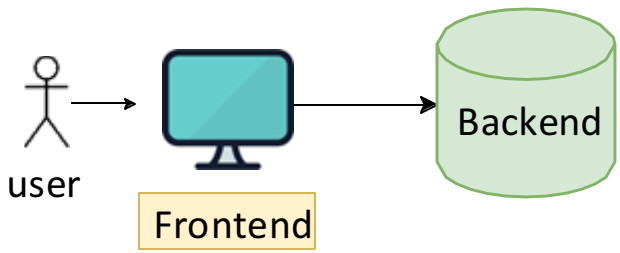
Good practices: Project Management

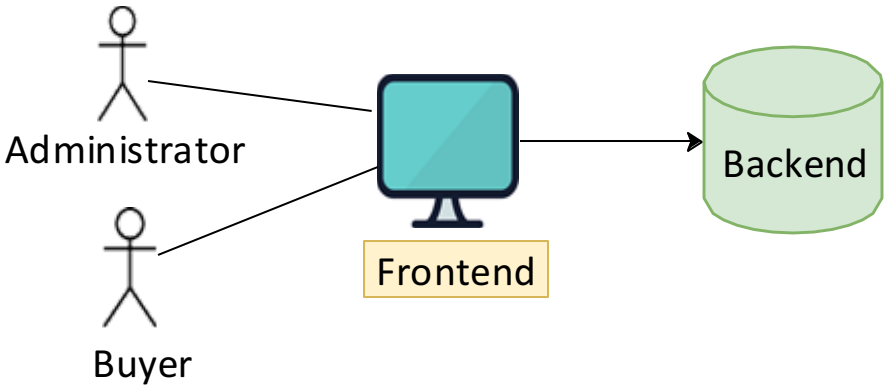
- Negotiate the project scope whenever a new requirement is introduced
- Select an appropriate software development life cycle
- Estimate requirements effort
- Identify requirements decision-makers
- Base project plans on requirements
- Document the lessons learned from the current project to use it in the next projects

Our understanding evolves over time

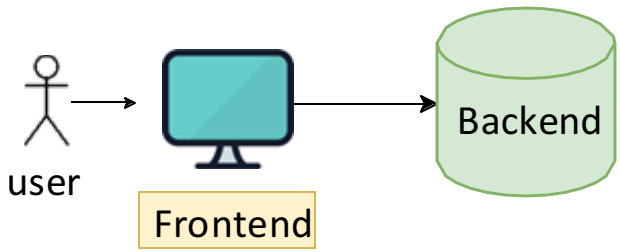


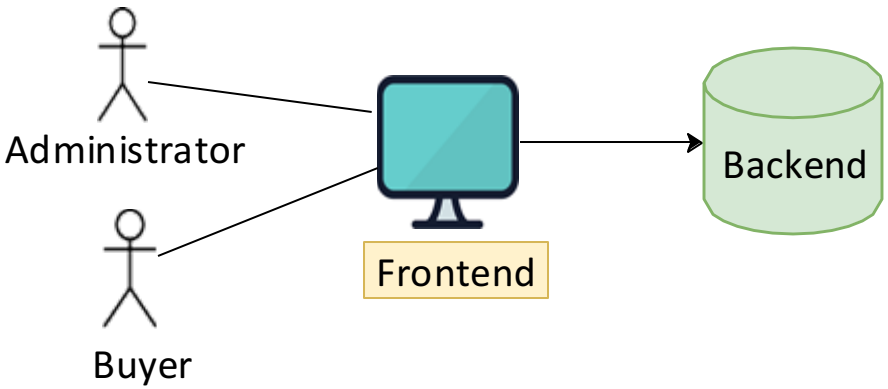
Our understanding evolves over time

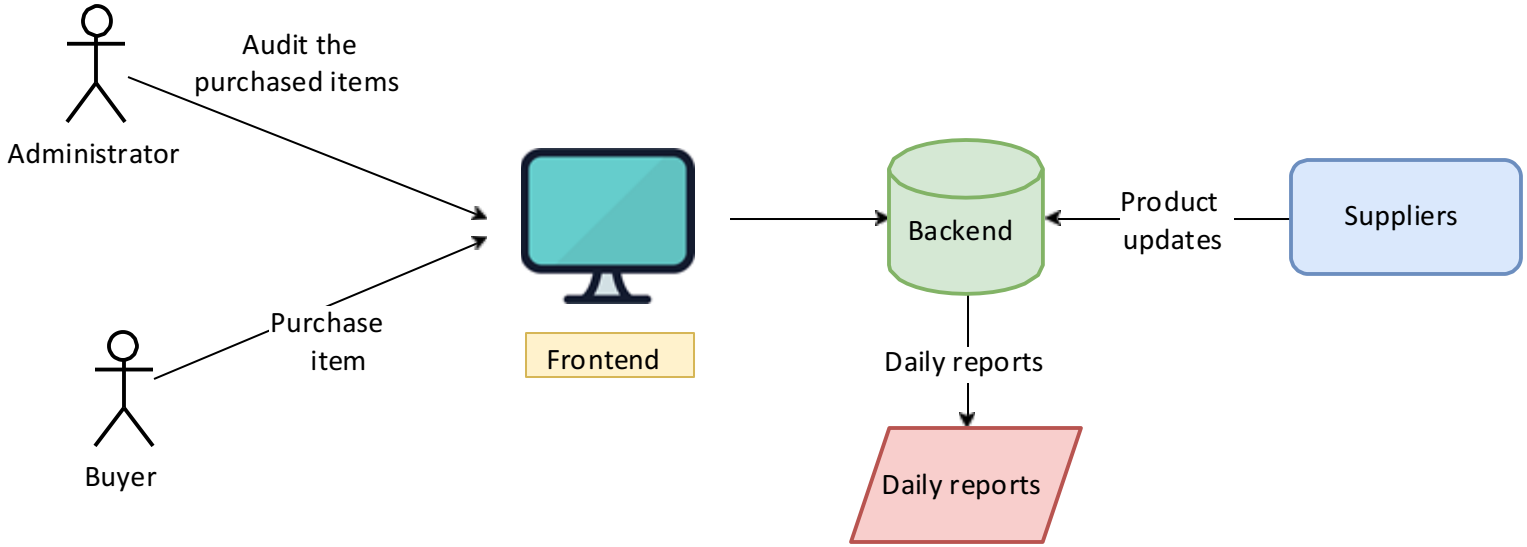




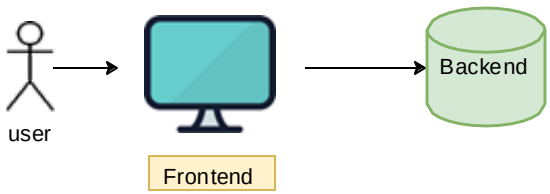
Our understanding evolves over time

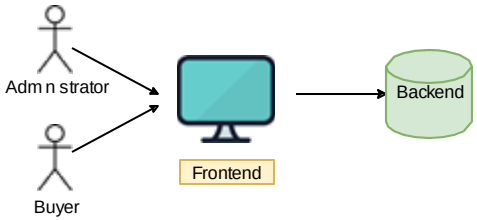


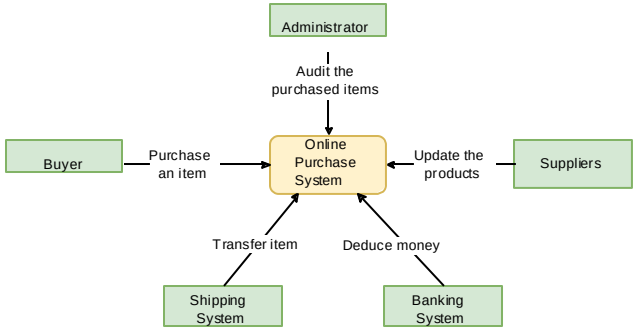




Our understanding evolves over time







SR ID	UR ID	BR ID	Software requirement	Design diagram	Component	Test case ID
FR_I_1	UR_I_1	BR_I_1	The buyer shall be able to see the list of available items.	Buy order flowchart	Item catalogue component	TC_01
FR_I_2	UR_I_1	BR_I_1	The buyer shall be able to select a suitable item and the delivery location.	Buy order flowchart	Order component	TC_02
FR_I_3	UR_I_1	BR_I_1	The buyer shall be able to review and confirm the order.	Buy order flowchart	Order component	TC_03
FR_I_4	UR_I_1	BR_I_1	The buyer shall be able to receive an email with the order status and details.	Buy order flowchart	Notification component	TC_04
NR_I_5	UR_I_1	BR_I_1	The backend system shall be able to fulfill the order within 2 seconds.	System interfaces diagram	Order system, backend integration component	TC_05

References

1. Software Requirements (Developer Best Practices) Karl Wiegers, Joy Beatty, 3rd Edition, Microsoft Press, 2013, ISBN-10: 0735679665