SOFTWARE REQUIREMENTS PART I

EGCO343 SOFTWARE DESIGN

KANAT POOLSAWASD
DEPARTMENT OF COMPUTER ENGINEERING
MAHIDOL UNIVERSITY

WHAT IS A REQUIREMENT?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
- May be the basis for a bid for a contract - therefore must be open to interpretation.
- May be the basis for the contract itself - therefore must be defined in detail.
- Both these statements may be called requirements.

TYPE OF REQUIREMENT

- **User requirements**
  Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- **System requirements**
  A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

- **Functional requirements**
  Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- **Non-functional requirements**
  Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- **Domain requirements**
  Requirements that come from the application domain of the system and that reflect characteristics of that domain.

FUNCTIONAL REQUIREMENTS

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

REQUIREMENTS COMPLETENESS AND CONSISTENCY

- In principle, requirements should be both complete and consistent.
- **Complete**
  - They should include descriptions of all facilities required.
- **Consistent**
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, it is impossible to produce a complete and consistent requirements document.

NON-FUNCTIONAL REQUIREMENTS

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

NON-FUNCTIONAL CLASSIFICATIONS

- **Product requirements**
  Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- **Organizational requirements**
  Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.
- **External requirements**
  Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

GOALS AND REQUIREMENTS

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- **Goal**
  A general intention of the user such as ease of use.
- **Verifiable non-functional requirement**
  A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

REQUIREMENTS MEASURES

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target-dependent statements Number of target systems

REQUIREMENTS INTERACTION

- Conflicts between different non-functional requirements are common in complex systems.
- Spacecraft system
  - To minimize weight, the number of separate chips in the system should be minimized.
  - To minimize power consumption, lower power chips should be used.
  - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

DOMAIN REQUIREMENTS

- Derived from the application domain and describe system characteristics and features that reflect the domain.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

SYSTEM REQUIREMENTS

- More detailed specifications of system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system.
- They may be incorporated into the system contract.
- System requirements may be defined or illustrated using system models.

USER REQUIREMENTS

- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don’t have detailed technical knowledge.
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

PROBLEMS WITH NATURAL LANGUAGE

- **Lack of clarity**
  Precision is difficult without making the document difficult to read.
- **Requirements confusion**
  Functional and non-functional requirements tend to be mixed-up.
- **Requirements amalgamation**
  Several different requirements may be expressed together.

PROBLEMS WITH NL SPECIFICATION

- **Ambiguity**
  The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.
- **Over-Flexibility**
  The same thing may be said in a number of different ways in the specification.
- **Lack of Modularization**
  NL structures are inadequate to structure system requirements.

GUIDELINES FOR WRITING REQUIREMENTS

- Invent a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.

ALTERNATIVES TO NL SPECIFICATION

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT (Ross, 1977) (Schoman and Ross, 1977). Now, use-case descriptions (Jacobsen, et al., 1993) and sequence diagrams are commonly used (Stevens and Pooley, 1999).
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

STRUCTURED LANGUAGE SPECIFICATIONS

- The freedom of the requirements writer is limited by a predefined template for requirements.
- All requirements are written in a standard way.
- The terminology used in the description may be limited.
- The advantage is that the most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification.

FORM-BASED SPECIFICATIONS (1)

- Definition of the function or entity.
- Description of inputs and where they come from.
- Description of outputs and where they go to.
- Indication of other entities required.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

FORM-BASED SPECIFICATIONS (2)

Insulin Pump/Control Software/SRS/3.3.2

Function	Compute insulin dose: Safe sugar level
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1)
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered
Destination	Main control loop

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requires	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2
Side effects	None

TABULAR SPECIFICATION

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ((r2 – r1) > (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

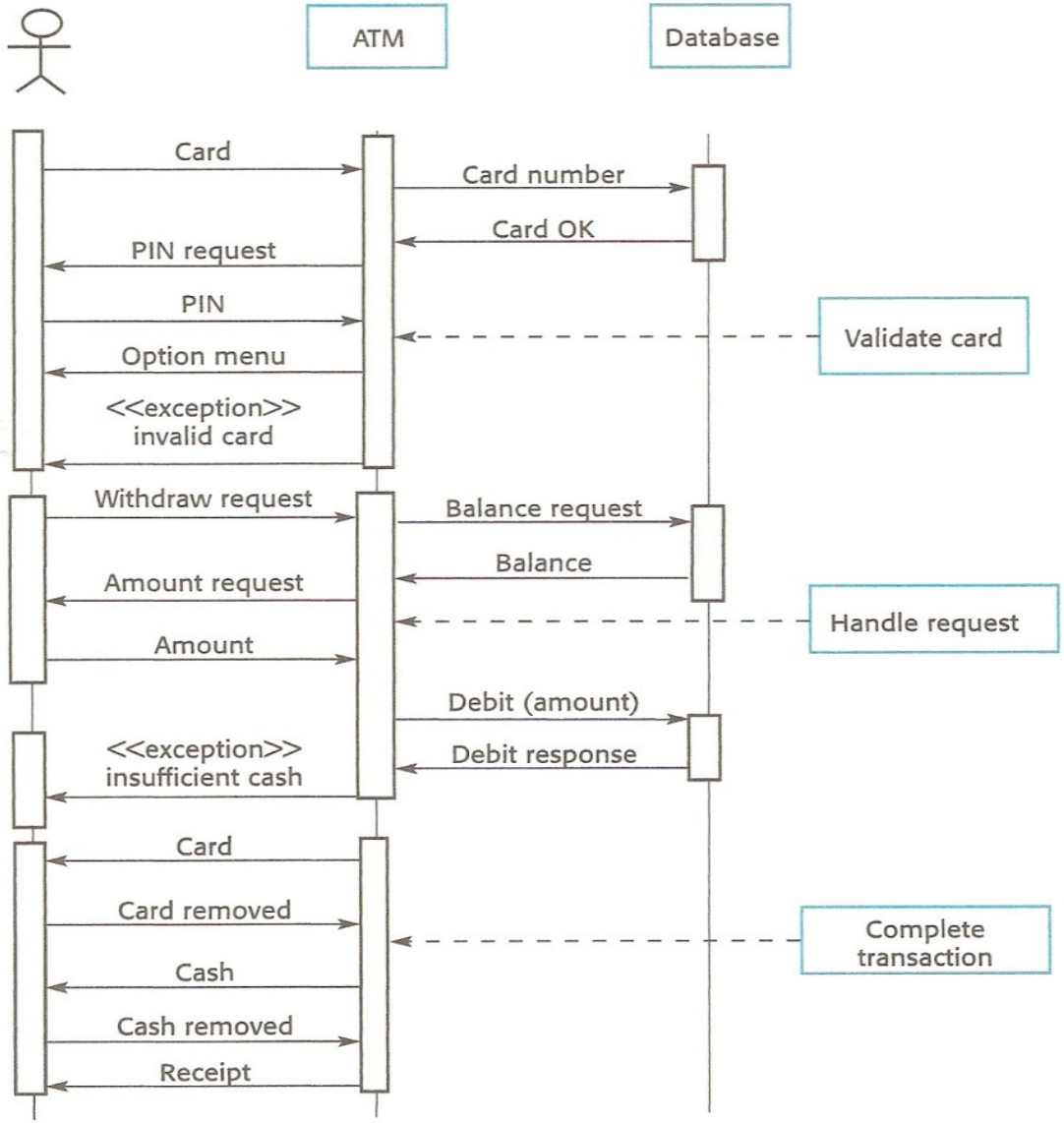
GRAPHICAL MODELS

- Graphical models are most useful when you need to show how state changes or where you need to describe a sequence of actions.
- Different graphical models are explained in next week.

SEQUENCE DIAGRAMS

- These show the sequence of events that take place during some user interaction with a system.
- You read them from top to bottom to see the order of the actions that take place.
- Cash withdrawal from an ATM
  - Validate card.
  - Handle request.
  - Complete transaction.

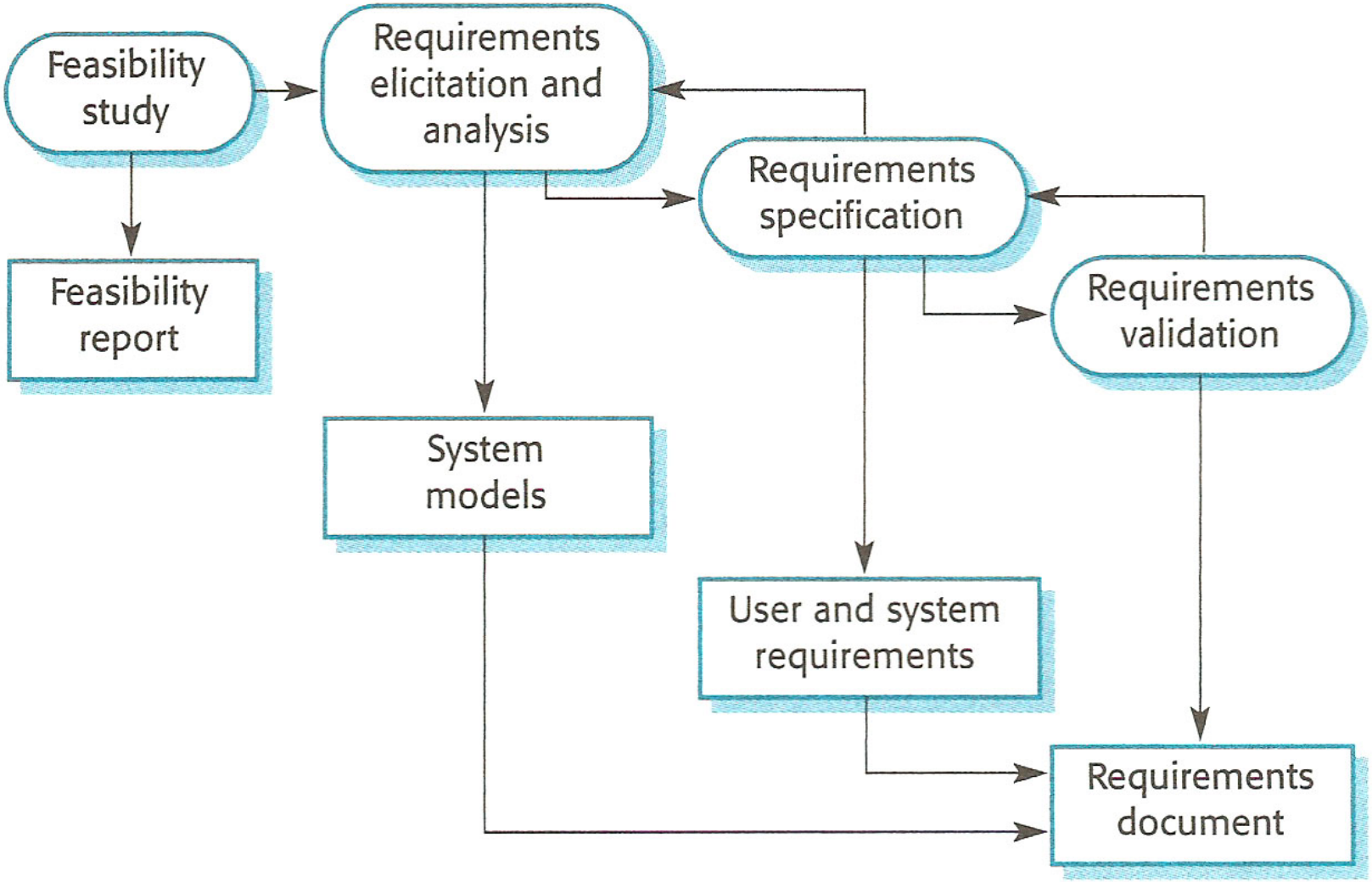
SEQUENCE DIAGRAM OF ATM WITHDRAWAL



INTERFACE SPECIFICATION

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
- Three types of interface may have to be defined
  - Procedural interfaces.
  - Data structures that are exchanged.
  - Data representations.
- Formal notations are an effective technique for interface specification.

REQUIREMENTS ENGINEERING PROCESSES



FEASIBILITY STUDIES

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organizational objectives.
  - If the system can be engineered using current technology and within budget.
  - If the system can be integrated with other systems that are used.

ELICITATION AND ANALYSIS

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called stakeholders.

PROBLEMS OF REQUIREMENTS ANALYSIS

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organizational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

VIEWPOINTS

- Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders.
- Stakeholders may be classified under different viewpoints.
- This multi-perspective analysis is important as there is no single correct way to analyze system requirements.

TYPE OF VIEWPOINTS

- **Inter-actor viewpoints**
  People or other systems that interact directly with the system. In an ATM, the customer's and the account database are inter-actor VPs.
- **Indirect viewpoints**
  Stakeholders who do not use the system themselves but who influence the requirements. In an ATM, management and security staff are indirect viewpoints.
- **Domain viewpoints**
  Domain characteristics and constraints that influence the requirements. In an ATM, an example would be standards for inter-bank communications.

INTERVIEWING

- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
  - Closed interviews where a pre-defined set of questions are answered.
  - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

INTERVIEWS IN PRACTICE

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviews are not good for understanding domain requirements

EFFECTIVE INTERVIEWERS

- Interviewers should be open-minded, willing to listen to stakeholders and should not have pre-conceived ideas about the requirements.
- They should prompt the interviewee with a question or a proposal and should not simply expect them to respond to a question such as ‘what do you want’.

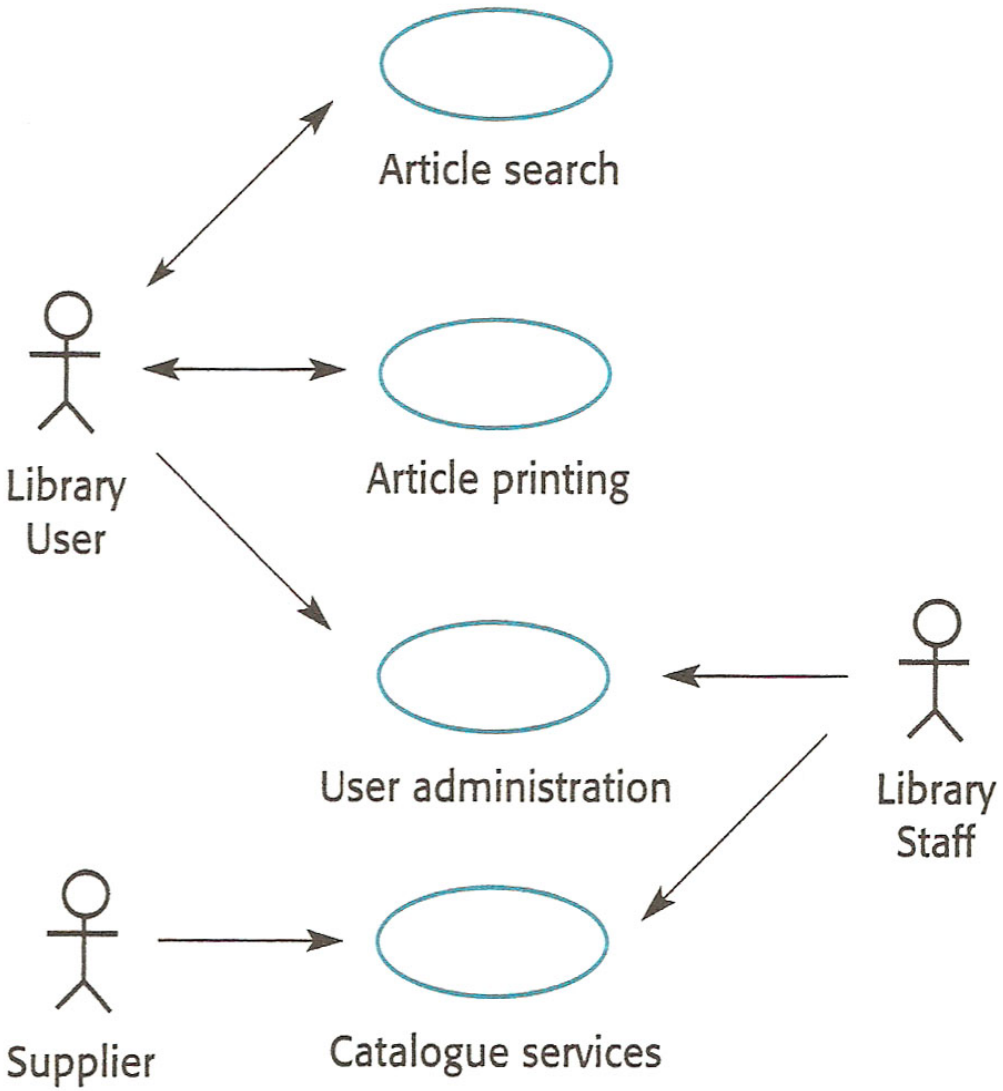
SCENARIOS

- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation.
  - A description of the normal flow of events.
  - A description of what can go wrong.
  - Information about other concurrent activities.
  - A description of the state when the scenario finishes.

USE CASES (1)

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

USE CASES (2)


Article search
Library User
Article printing
User administration
Library Staff
Supplier
Catalogue services

SOCIAL AND ORGANIZATIONAL FACTORS

- Software systems are used in a social and organizational context. This can influence or even dominate the system requirements.
- Social and organizational factors are not a single viewpoint but are influences on all viewpoints.
- Good analysts must be sensitive to these factors but currently no systematic way to tackle their analysis.

REQUIREMENTS VALIDATION

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

REQUIREMENTS CHECKING

- **Validity**. Does the system provide the functions which best support the customer's needs?
- **Consistency**. Are there any requirements conflicts?
- **Completeness**. Are all functions required by the customer included?
- **Realism**. Can the requirements be implemented given available budget and technology
- **Verifiability**. Can the requirements be checked?

REQUIREMENTS VALIDATION TECHNIQUES

- **Requirements reviews**
  Systematic manual analysis of the requirements.
- **Prototyping**
  Using an executable model of the system to check requirements.
- **Test-case generation**
  Developing tests for requirements to check testability.

REQUIREMENTS REVIEWS

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

REVIEW CHECKS

- **Verifiability**. Is the requirement realistically testable?
- **Comprehensibility**. Is the requirement properly understood?
- **Traceability**. Is the origin of the requirement clearly stated?
- **Adaptability**. Can the requirement be changed without a large impact on other requirements?

REQUIREMENTS CHANGE

- The priority of requirements from different viewpoints changes during the development process.
- System customers may specify requirements from a business perspective that conflict with end-user requirements.
- The business and technical environment of the system changes during its development.

ENDURING AND VOLATILE REQUIREMENTS

- **Enduring Requirements**. Stable requirements derived from the core activity of the customer organization. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- **Volatile Requirements**. Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

REQUIREMENTS CLASSIFICATION

Requirement Type	Description
Mutable requirements	Requirements which change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.
Emergent requirements	Requirements which emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements which result from the introduction of the computer system. Introducing the computer system may change the organisation's processes and open up new ways of working which generate new system requirements.
Compatibility requirements	Requirements which depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

TRACEABILITY

- Traceability is concerned with the relationships between requirements, their sources and the system design
- **Source Traceability**
  Links from requirements to stakeholders who proposed these requirements.
- **Requirements Traceability**
  Links between dependent requirements.
- **Design Traceability**
  Links from the requirements to the design.

CASE TOOL SUPPORT

- **Requirements Storage**
  Requirements should be managed in a secure, managed data store.
- **Change Management**
  The process of change management is a workflow process whose stages can be defined and information flow between these stages partially automated.
- **Traceability Management**
  Automated retrieval of the links between requirements.

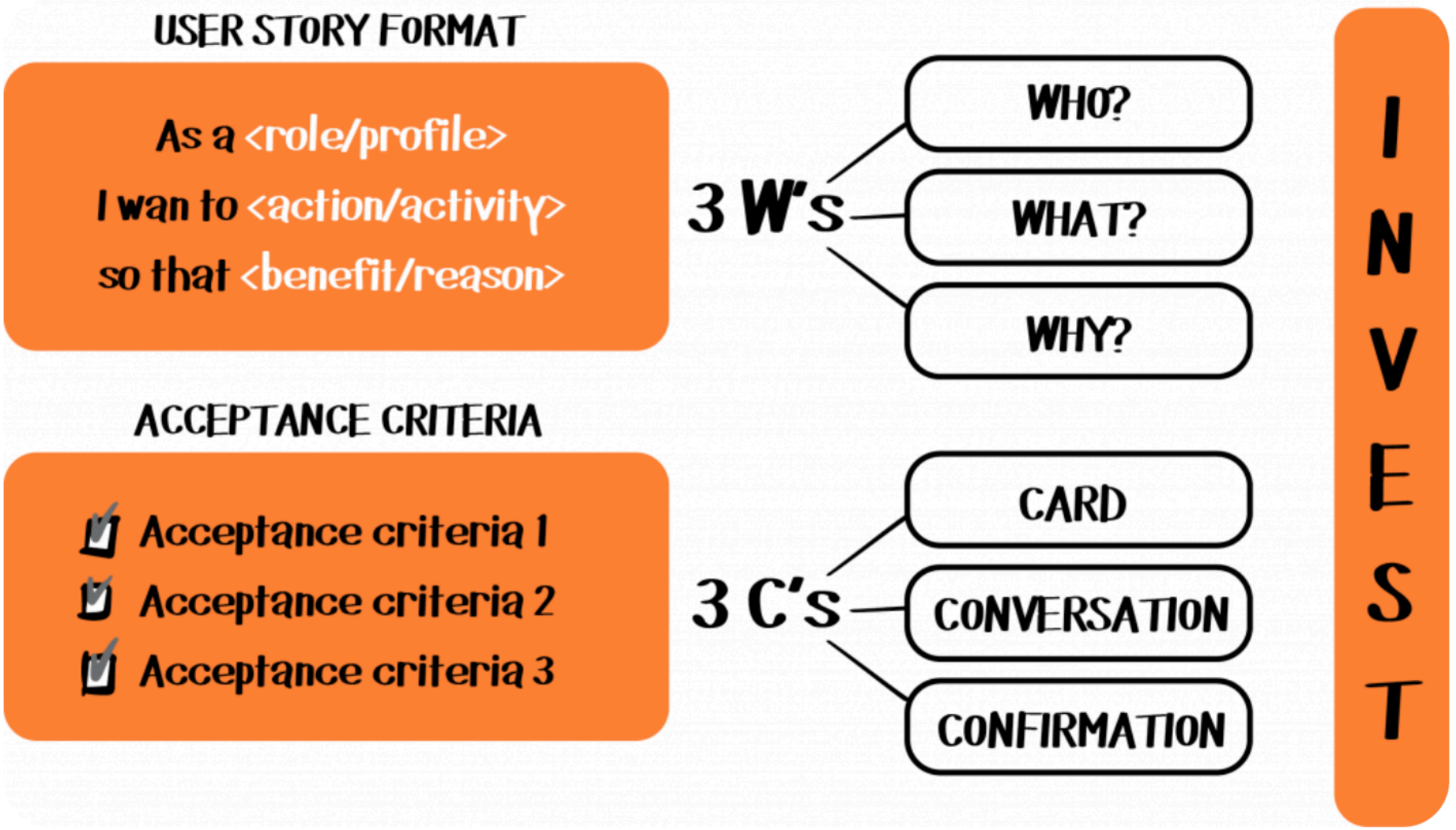
REQUIREMENTS CHANGE MANAGEMENT

- Should apply to all proposed changes to the requirements.
- Principal stages
  - **Problem analysis:** Discuss requirements problem and propose change.
  - **Change analysis and costing:** Assess effects of change on other requirements.
  - **Change implementation:** Modify requirements document and other documents to reflect change.

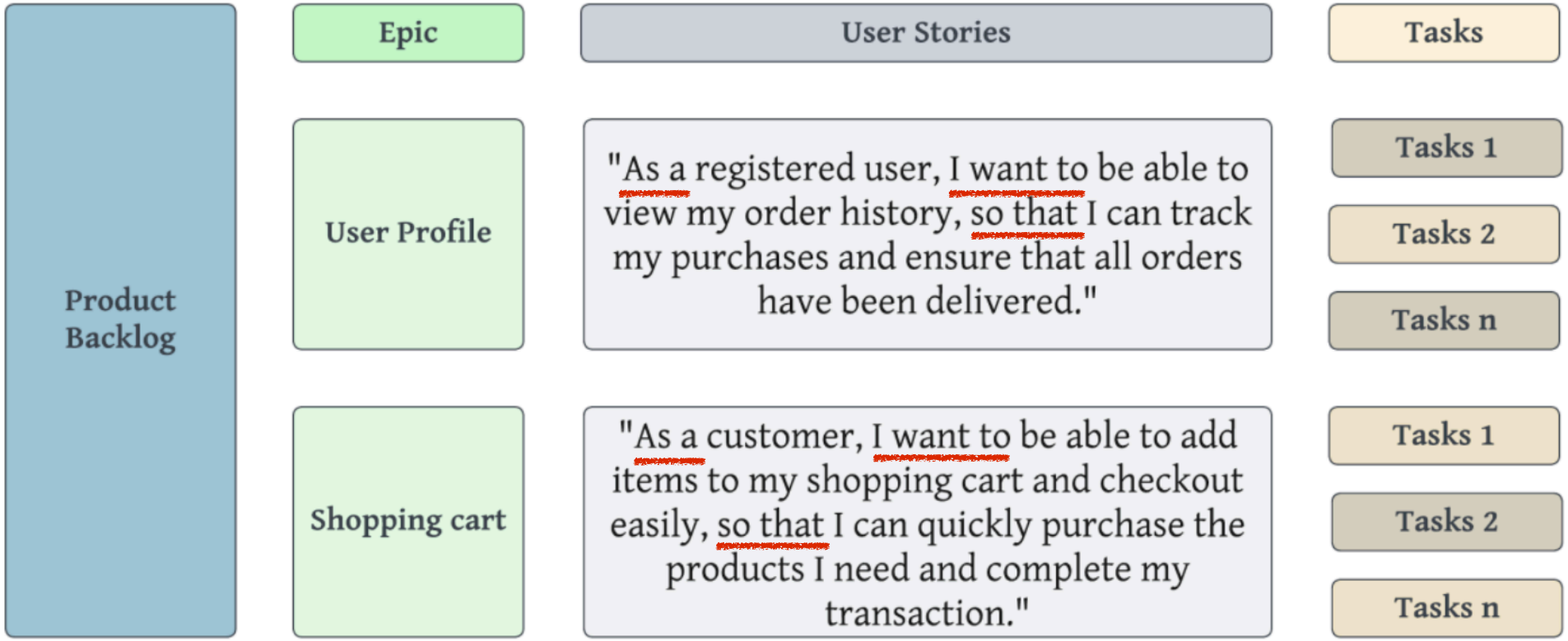
USER STORY

- A user story is a short, simple description of a feature or functionality told from the perspective of the end user.
- It’s commonly used in Agile and Scrum development to capture what the user needs and why, without going into technical details.

USER STORIES FORMAT (1)



USER STORIES FORMAT (2)



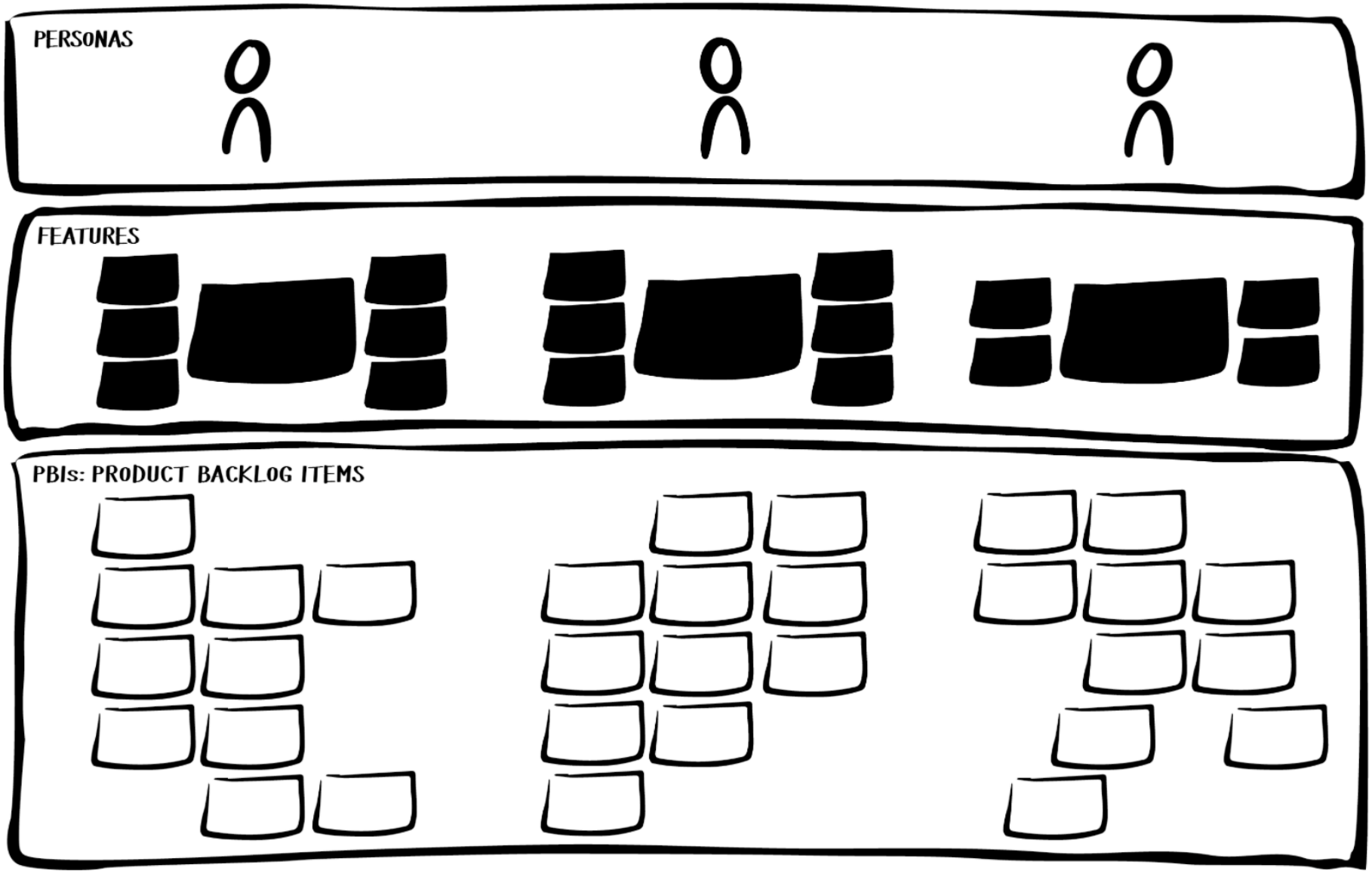
EFFECTIVE USER STORIES (1)

- To be effective, user stories should be written using the **INVEST** criteria, which stands for:
  - **Independent:** Each user story should be independent of others, meaning that it can be developed and delivered without relying on other stories.
  - **Negotiable:** User stories should be open to negotiation and discussion, and allow the team to make changes as needed based on feedback and new information.

EFFECTIVE USER STORIES (2)

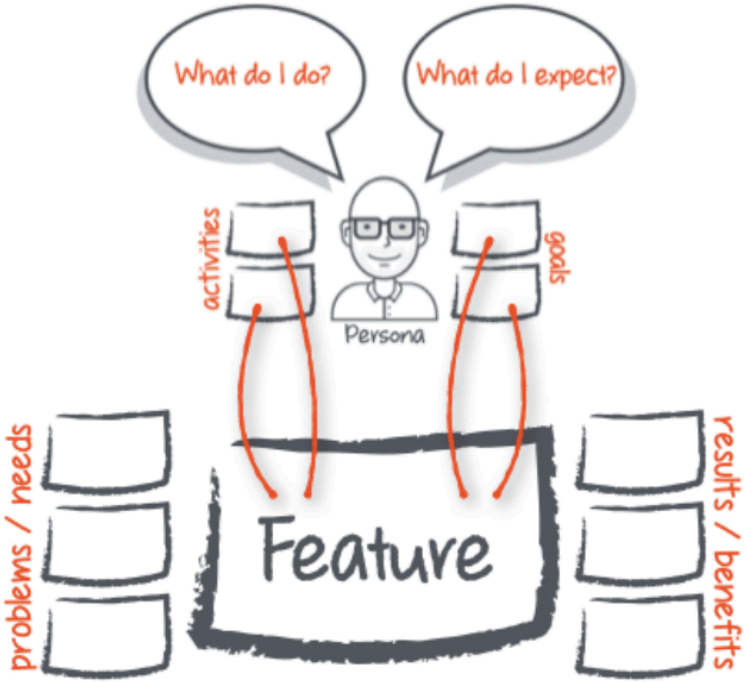
- **Valuable:** User stories should be valuable to the end user or customer, and contribute to the overall goals of the project.
- **Estimable:** User stories should be estimable, meaning that the team can provide a reasonable estimate of the effort needed to complete the story.
- **Small:** User stories should be small and manageable, and able to be completed within a single sprint.
- **Testable:** User stories should be testable, meaning that the team can create acceptance criteria that define what needs to be done for the story to be considered complete.

PRODUCT BACKLOG BUILDING (PBB)



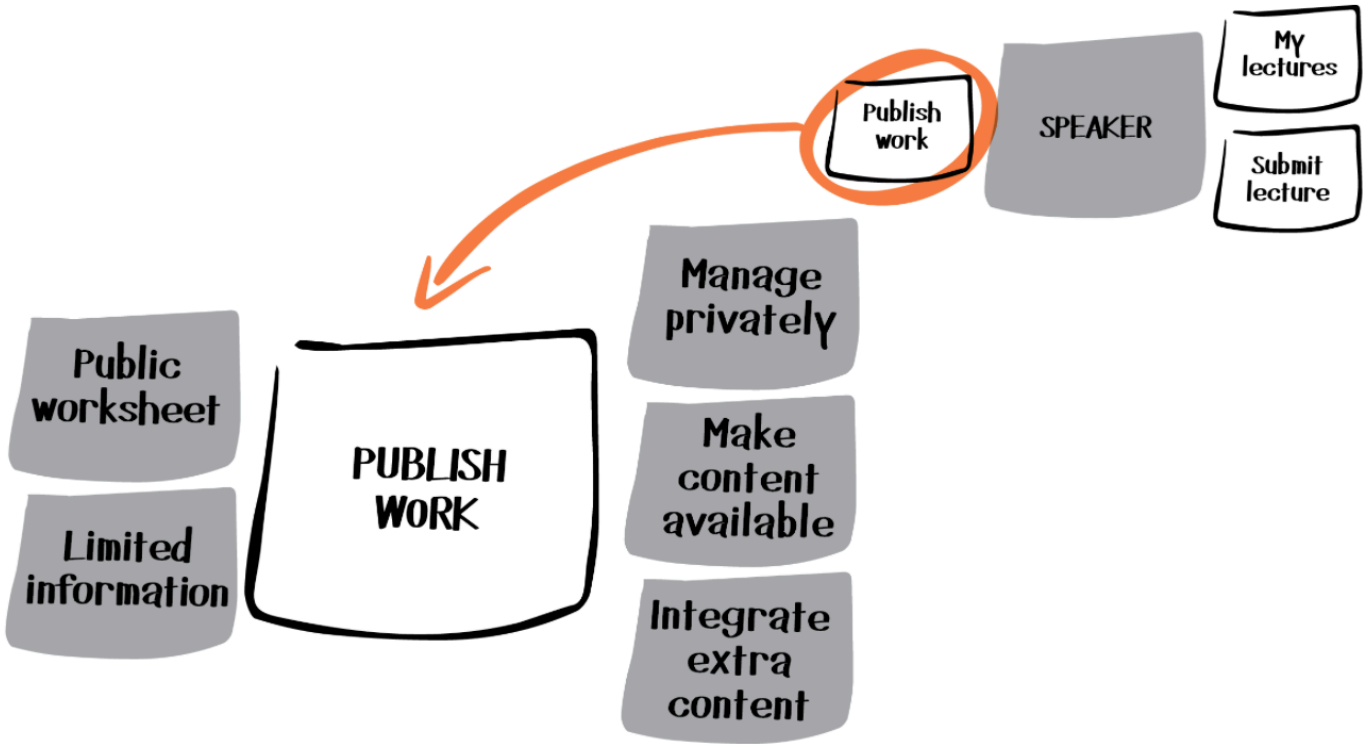
FROM PERSONA TO FEATURE

- Describe the features from a answering the following questions:
  - The user is trying to do something, so the product must have a feature for that. What is it?
  - What persona issues does this feature solve?
  - What benefits does it bring to the persona?



FEATURE BLOCK EXAMPLE

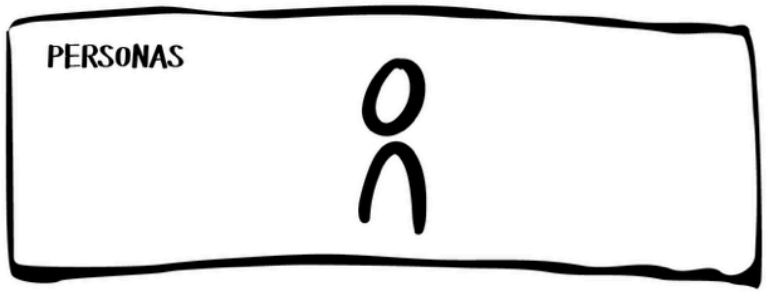
- Write the feature description on one big post-it, then write down the challenges and benefits on smaller post-Its and place them next to the post-It, describing the feature.

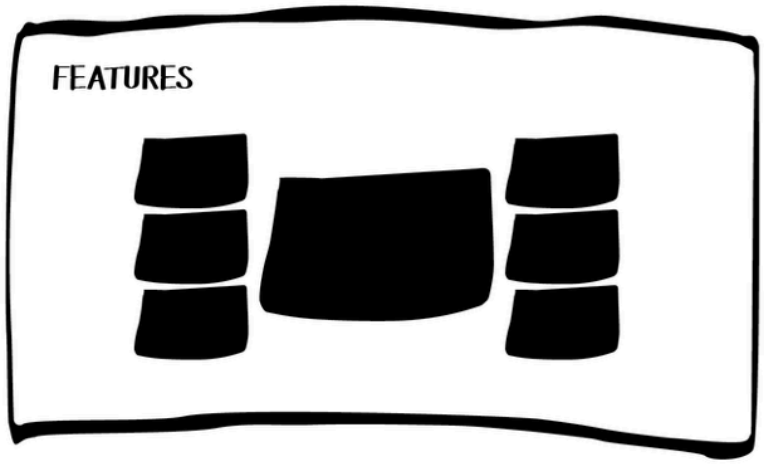


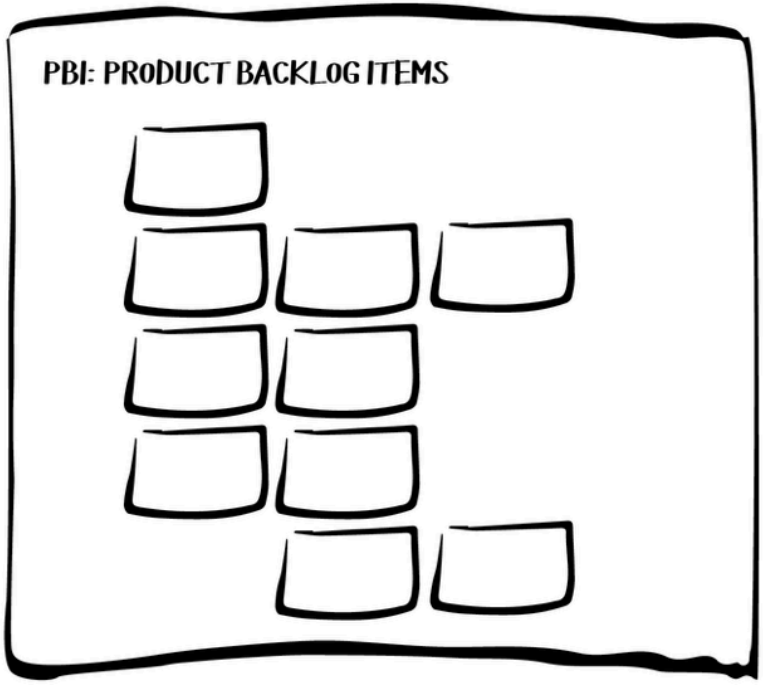
IDENTIFY THE PBI(S)

- Product Backlog Items (PBIs) are elements that make up the product backlog. They reflect the development work needed to improve the product and to meet customer or stakeholder needs.
- Now it becomes necessary to break these down further to allow for smaller, more accurate items. These are called PBIs.
- To identify the respective PBIs in the product backlog, ask participants to answer the following questions: *"What is the first work item (or step) for this feature? And the second? And the next ones?"*

PRODUCT BACKLOG BUILDING







Publish work

SPEAKER

My talks

Submit talk

Public worksheet

Limited information

PUBLISH WORK

Manage privately

Make content available

Integrate extra content

Perform the speaker registration

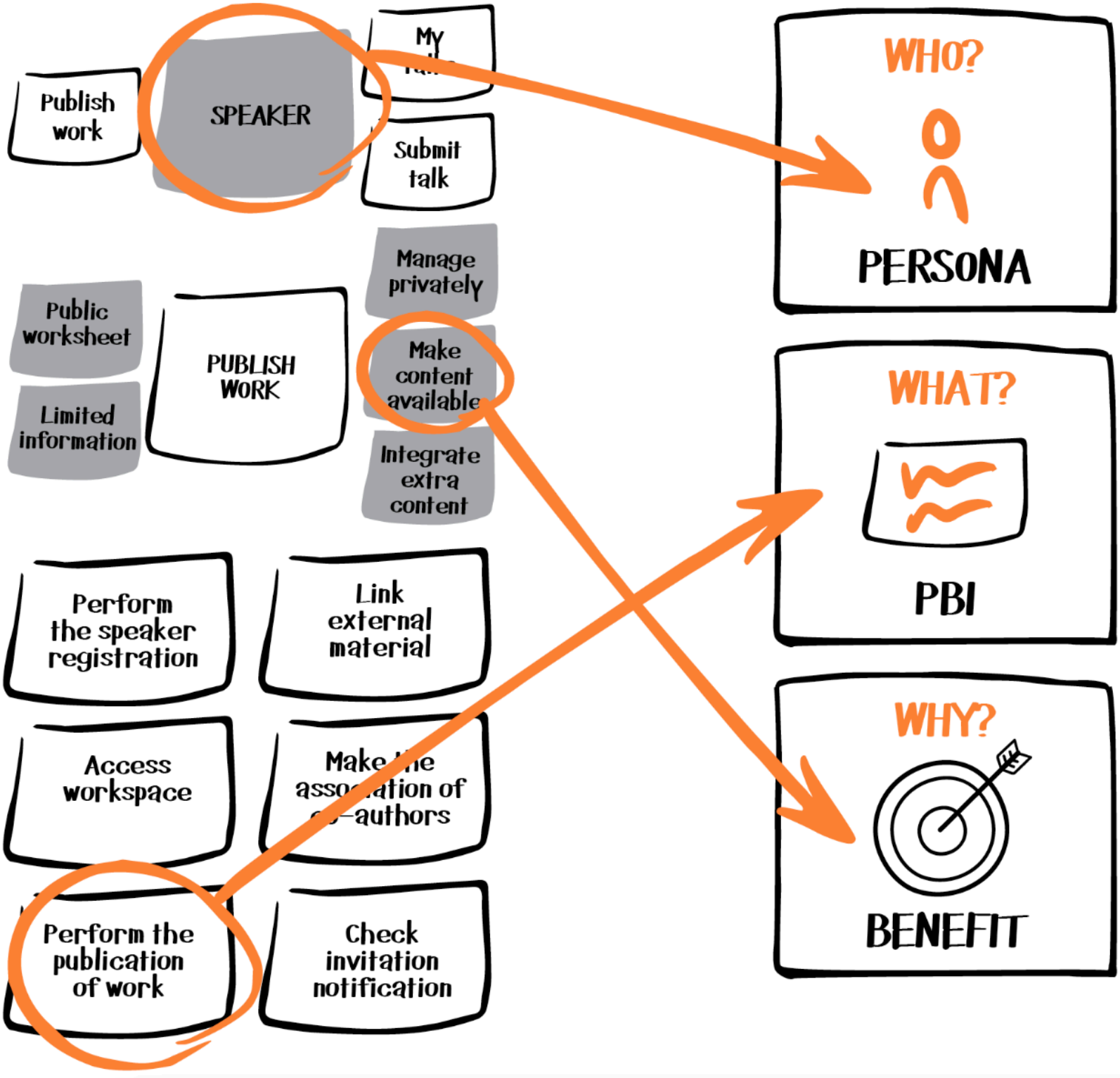
Link external material

Access workspace

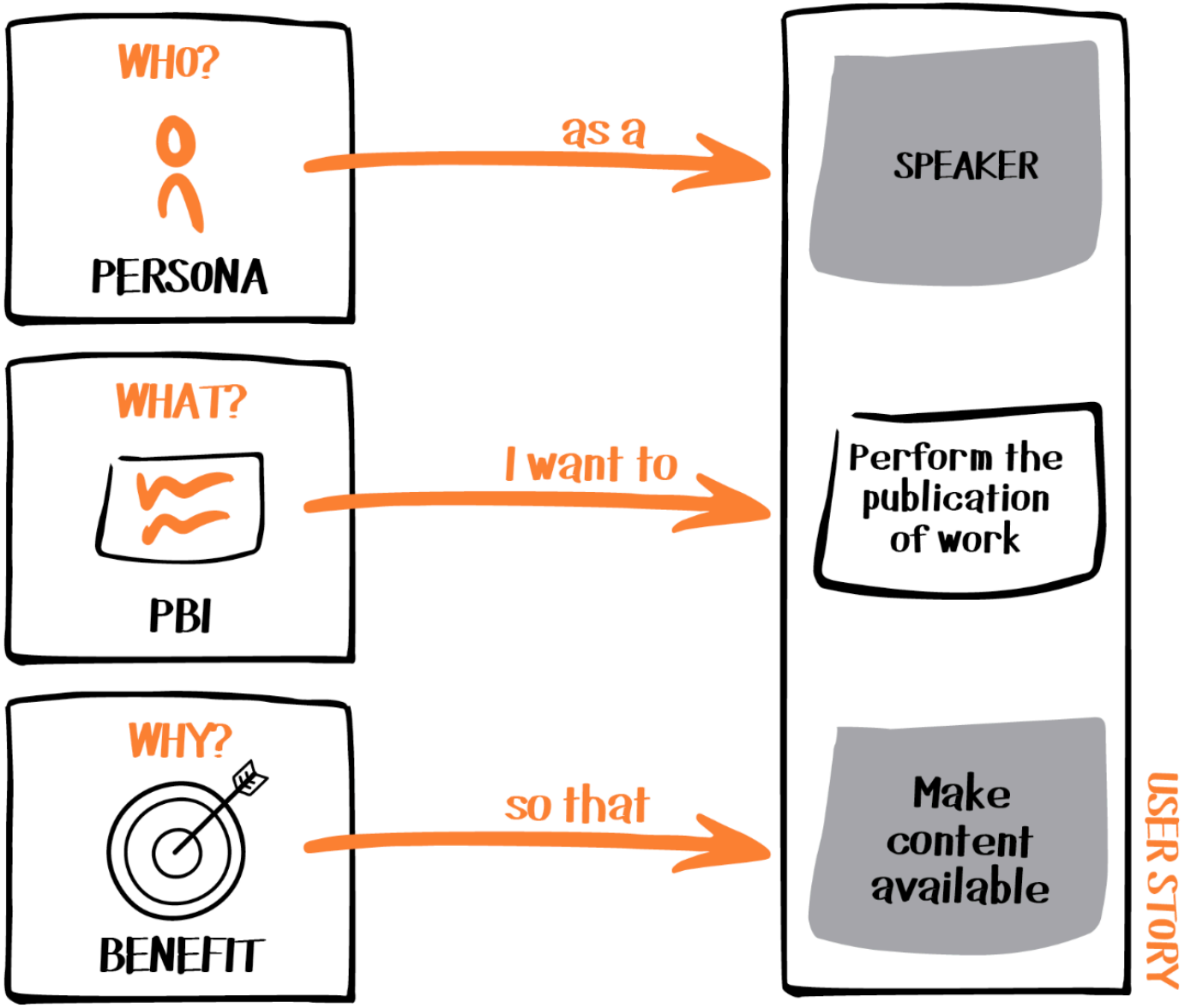
Make the association of co-authors

Perform the publication of work

Check invitation notification


Publish work
SPEAKER
My
Submit talk
Public worksheet
Limited information
PUBLISH WORK
Manage privately
Make content available
Integrate extra content
Perform the speaker registration
Link external material
Access workspace
Perform the publication of work
Check invitation notification
WHO?
PERSONA
WHAT?
PBI
WHY?
BENEFIT

PRODUCT BACKLOG BUILDING (PBB)



EXAMPLES

Persona: Speaker

Feature 1: Publish talk

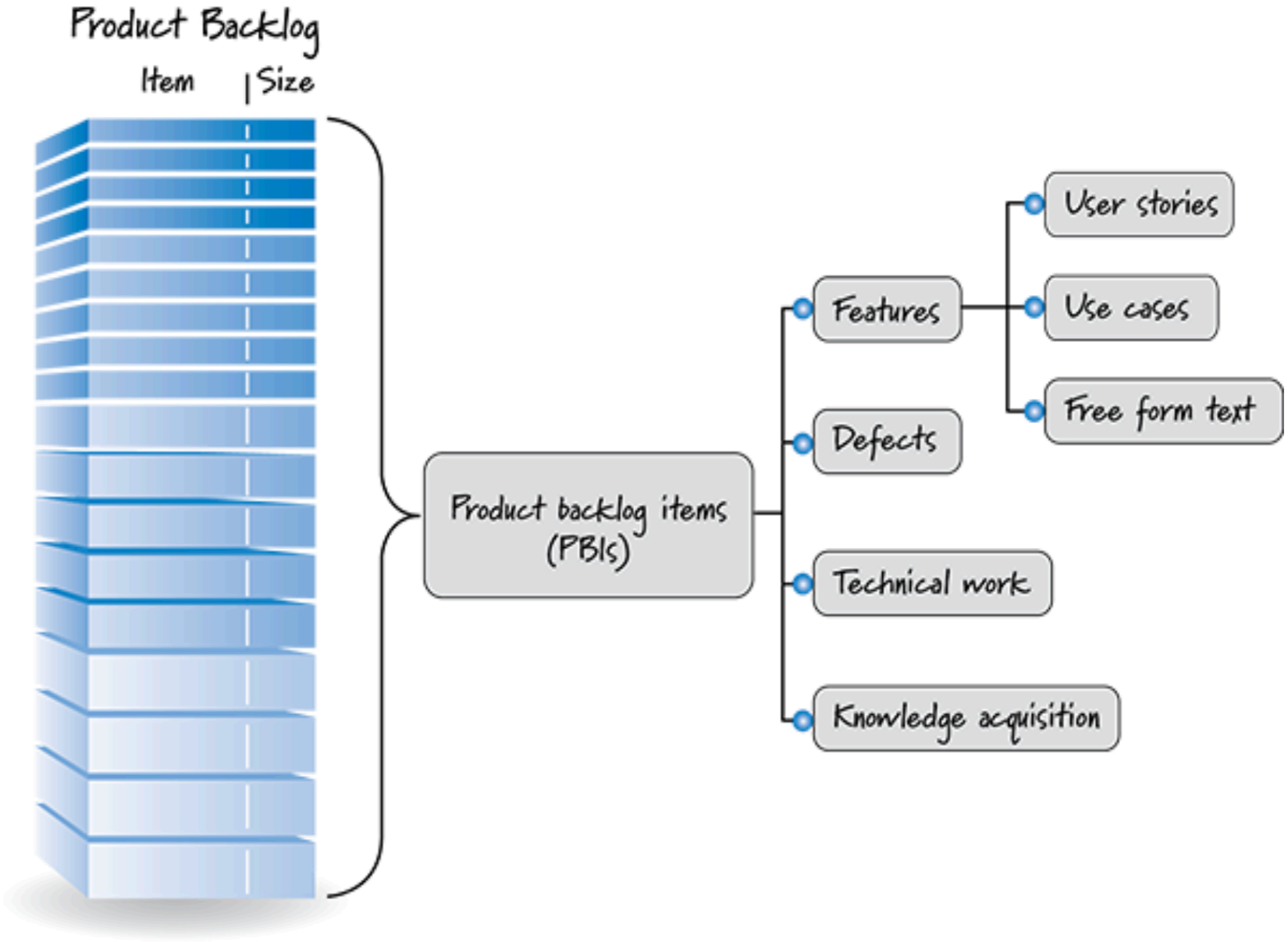
- Story 1.1: As a speaker, I want to access a workspace in order to manage talks privately
- Story 1.2: As a speaker, I want to publish talks in order to make content available.
- Story 1.3: As a speaker, I want to link the external presentation in order to integrate talks

Persona: Participant

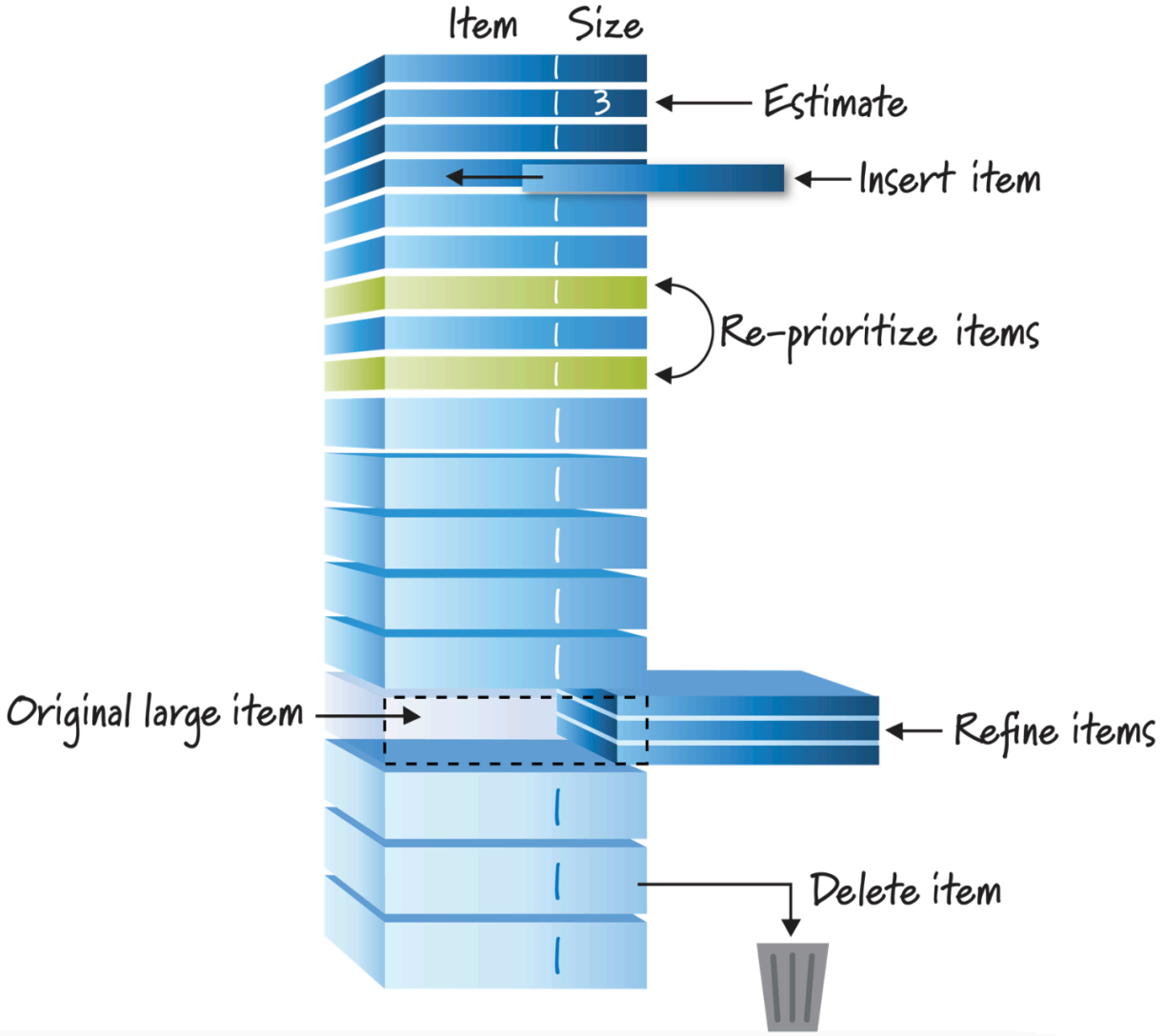
Feature 2: Participate in an event

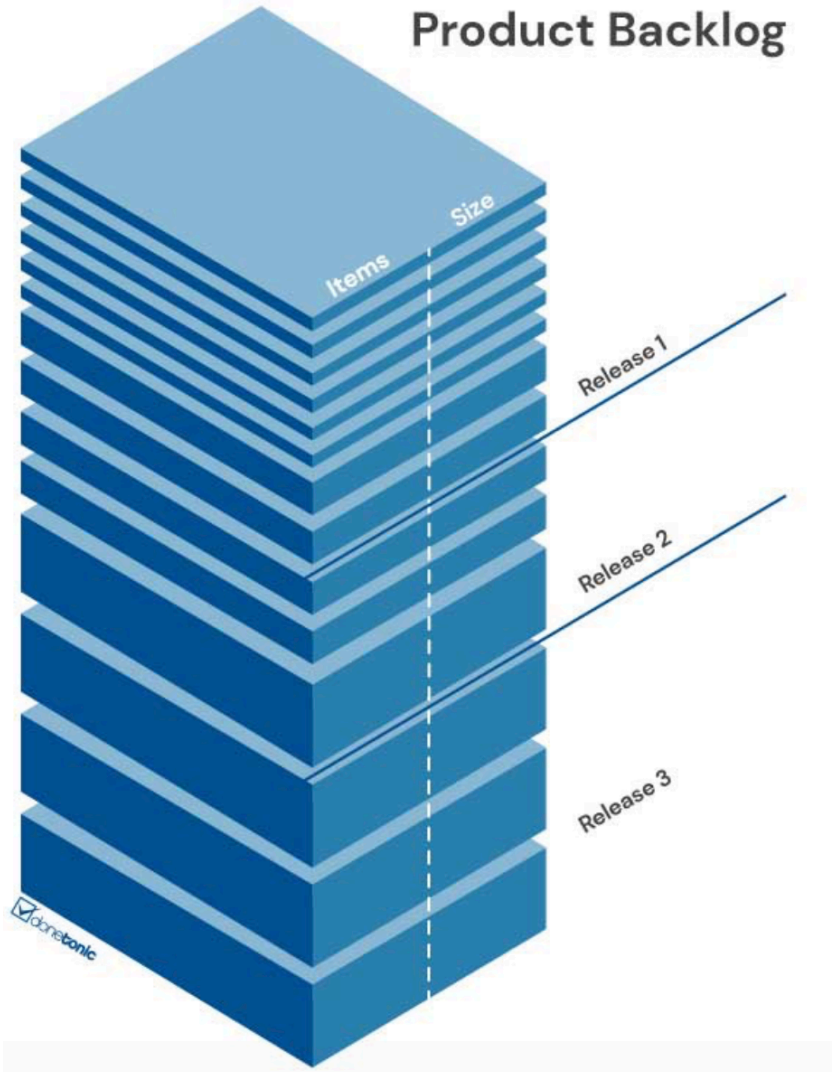
- Story 2.1: As a participant, I want to find the available event in order to view the schedule.
- Story 2.2: As a participant, I want to register for an event in order to attend it.
- Story 2.3: As a participant, I want to check in at the event in order to confirm attendance.

PRODUCT BACKLOG CHARACTERISTICS (1)



PRODUCT BACKLOG CHARACTERISTICS (2)





EXAMPLE OF PBB (1)

- **Case Study:** An online coffee ordering system for coffee shops. (Coffee Order App)
- **Vision:** An application that allows customers to order coffee and snacks via mobile phone, pay online, and choose either to pick up their order at the store or have it delivered to their home.
- **Features:**
  - Registration/Login
  - Menu and Order
  - Payment
  - Tracking
  - Rate and Review

EXAMPLE OF PBB (2)

- **Registration/Login**
  "As a customer, I want to sign up with my email or social media so that I can place orders easily."
- **Menu and Order**
  "As a customer, I want to browse coffee and bakery menus so that I can choose what I want."
- **Payment**
  "As a customer, I want to pay via credit card or e-wallet so that my payment is quick and secure."
- **Tracking**
  "As a customer, I want to track my order status so that I know when it will be ready."
- **Rate and Review**
  "As a customer, I want to rate and review my order so that I can share my feedback."

EXAMPLE OF PBB (3)

ID	Feature	User Story	Priority	Estimation
PBB-001	Registration/Login	As a customer, I want to sign up with email or social media so that I can place orders easily.	High	5
PBB-002	Menu & Ordering	As a customer, I want to browse coffee menus so that I can choose my drink.	High	8
PBB-003	Online Payment	As a customer, I want to pay via credit card or e-wallet so that my payment is quick and secure.	High	8
PBB-004	Order Tracking	As a customer, I want to track my order so that I know when it will be ready.	Medium	5
PBB-005	Review & Rating	As a customer, I want to rate and review my order so that I can share feedback.	Low	3

SOFTWARE REQUIREMENTS SPECIFICATION (SRS) [1]

- A Software Requirements Specification (SRS) is a document that describes the nature of a project, software or application.
- In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application.
- This document is also known by the names SRS report, software document. A software document is primarily prepared for a project, software or any kind of application.

SOFTWARE REQUIREMENTS SPECIFICATION (SRS) [2]

- There are a set of guidelines to be followed while preparing the software requirement specification document.
- This includes the purpose, scope, functional and nonfunctional requirements, software and hardware requirements of the project.
- In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

WHAT IS A SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT?

- A Software requirements specification document describes the intended purpose, requirements and nature of a software to be developed. It also includes the yield and cost of the software.

TABLE OF CONTENTS FOR SRS DOCUMENT

- Introduction
- Overall Description
- System Features (Functional Requirements)
- Non-Functional Requirements
- Appendix

INTRODUCTION

- Purpose
- Scope
- Definition, Acronyms, and Abbreviations
- References

OVERALL DESCRIPTION

- Product Perspective
- Product Functions
- User Characteristics
- Constraints
- Assumption and Dependencies

SYSTEM FEATURES

- External Interface Requirements
- Functional Requirements
- Design Constraints
- Software System Attributes

NON-FUNCTIONAL REQUIREMENTS

- Assumptions and Dependencies
- Security
- Scalability
- Reliability
- Robustness
- Interoperability
- Operating Environment
- Supportability
- User Documentation

EXAMPLES OF SRS DOCUMENT

- The example of an SRS can be downloaded from the course materials on the course website.

ASSIGNMENT 2 (1)

- **กรณีศึกษา:** ให้แต่ละกลุ่มเก็บข้อมูลของ “ระบบจองห้อง” จากเจ้าหน้าที่ของภาควิชาวิศวกรรมคอมพิวเตอร์ (2 ท่าน)
- **เป้าหมาย:** นักศึกษาและบุคลากร สามารถจองห้องของภาควิชาฯ
  - อาคารวิศวกรามศาสตร์ 2: CO-309/1, CO-309/2, CO-310
  - อาคารวิศวกรามศาสตร์ 3: 6272, 6273, 6274, 6275, 6276
- **ผู้มีส่วนได้ส่วนเสีย (Stakeholders)** มี 3 กลุ่ม ได้แก่ “ผู้จอง” “ผู้รับจอง” และ “ผู้ดูแลห้อง”
  - ผู้รับจอง (พี่พลอย) ทำหน้าที่ตรวจสอบเวลาและรับจองห้อง
  - ผู้ดูแลห้อง (พี่เบิร์ท) ทำหน้าที่เตรียมห้องและเปิดห้องตามเวลาที่จอง
- ให้แต่ละกลุ่มเลือกตัวแทนมากลุ่มละ 2 คน เพื่อเก็บข้อมูล (สัมภาษณ์) จากเจ้าหน้าที่ของภาควิชาฯ ทั้ง 2 ท่าน
- สมาชิกที่เหลือให้เลือกสมาชิกมา 1 คน เพื่อแสดงบทบาทเป็น “ผู้จอง”

ASSIGNMENT 2 (2)

- แต่ละกลุ่มมีเวลา 30 นาทีสำหรับการเก็บข้อมูลจากเจ้าหน้าที่แต่ละท่าน รวมเวลาเก็บข้อมูลทั้งสิ้น 1 ชั่วโมง
  - เขียน User Story ของทุกกลุ่มผู้มีส่วยได้ส่วนเสีย ลงใน Post-It
- วิเคราะห์ความต้องการของระบบลงในไฟล์ Excel (ให้เวลา 1 ชั่วโมง)
  - กำหนดรหัส จัดลำดับความสำคัญ (High/Medium/Low) และเรียงลำดับของ User Story ในไฟล์ Excel (ชีท User Story)
  - รายละเอียดของ Requirements (ชีท Requirements) โดยมีรายละเอียด
    - Requirement Origin ให้อ้างอิงจาก User Story ID
    - Functional, Non-Functional, Domain Requirements
    - Enduring or Volatile Requirements
    - Requirement Dependencies