

# Requirements Engineering Report

## Elicitation Report

### **Describe the process of eliciting requirements for a software project in general including at least four sources and four common elicitation techniques.**

The process of eliciting requirements is one of the most fundamental for any given software project. It concerns how requirements are formed and how software engineers extract those formed requirements, setting the groundwork for the project team's understanding of the software project as a whole. Since the elicitation phase is primarily concerned with how software requirements are formed, it follows that it's heavily based on stakeholder interaction. This stage is where the Business Analyst and other project associates set up a foundation for communication between the project team and the project's stakeholders (who should, in the long run, be considered a part of this team). During this phase, the project's scope should be established and many requirements for the project should be appropriately elicited from stakeholders and well-formed.

During requirements elicitation, requirements are elicited from a wide variety of sources. These sources include goals, stakeholders, business rules, and the operational environment. Goals refer to the main objectives of the software. If these goals are well-defined, strong requirements will usually follow. Stakeholders are considered to be anyone who would have an interest in the success of the software project, such as a user or a manager of someone who uses the system. Stakeholders are a particularly effective source of requirements, as requirements elicited directly from those impacted by the system have a high chance of being relevant to the end product. Business rules refer to restrictions placed on an internal element of the business. As these serve as strict guidelines or restrictions, they can frame strong requirements easily. The operational environment can be considered to be the realm within which the system will operate. You can elicit requirements that frame interactions with other systems or requirements that restrict execution time from this context.

Requirements are also elicited using many different methods. These techniques include interviews, prototypes, observation, and facilitated meetings. Interviews are simple individual or group meetings with stakeholders. They're used to open a dialogue between the Business Analyst and clarify any vague requirements whilst eliciting new ones directly from valuable sources. Prototypes are interactable mockups of the system's concept. They're particularly valuable due to how they encourage iteration, with many requirements being formed out of stakeholder criticisms. Observation is a technique which involves the physical monitoring of employee workflows. This method is particularly useful for gleaning any requirements that stakeholders might take for granted or not understand about the system they currently operate. Lastly, facilitated meetings involve gathering a group of stakeholders in one room with the express purpose of discussing system requirements. The Business Analyst serves as a mediator, ensuring all viewpoints are heard and taking care to resolve any conflicts that arise between different stakeholders.

### **Describe how the requirements engineering process supports the elicitation of behavioral requirements.**

The main way that the elicitation of behavioral requirements is supported is through the utilization of user-oriented elicitation techniques such as user stories, use cases, use case diagrams, and prototypes. User stories are, in essence, bite-sized scenarios that describe a particular user, the desired function they wish to utilize, and why they wish to utilize said function. These stories go beyond the simple purpose of listing specific behaviors of the system, choosing additionally to contextualize their use and justify their existence within the system. By allowing stakeholders to write these user stories, they're given the opportunity to both properly formulate and justify a specific behavior of their desired system.

Rather than laying out stories, use cases choose to focus on specific actions that can be executed by specific users. To rephrase, a given use case outlines an action (use/function/behavior) of a system that can be performed by one or more users (actors). These use cases that describe individual behaviors are then often collected and organized into a use case diagram. This diagram is a visual aid for use cases, with lines connecting individual use cases with their respective actors to contextualize the behaviors of the system and show how they're organized for each user. Furthermore, relationships between use cases and actors are also expressed, showing how each behavior functions within the larger system. When it comes to behavioral requirements elicitation, use cases and use case diagrams are particularly useful for pinpointing specific details about a behavior, allowing for a constructive dialogue between the elicitor and their stakeholders that can lead to the formulation of new behavioral requirements or the clarification of existing ones.

Prototypes are another fantastic way of eliciting behavioral requirements. These are, in essence, interactable product interfaces that demonstrate some sort of system behavior when interacted with. Iterative prototypes that take feedback from the client and implement it in future prototypes are particularly useful for eliciting behavioral requirements. Let's say you present your client with a functional prototype based off their initial set of requirements. However, when you show it to them, they say that it's missing some features they'd like to see implemented in the future, such as specific actions that occur when you press a button or a specific interaction changing something in a different tab. By taking notes on these suggestions and implementing them in a future release, you've elicited and implemented behavioral requirements from your user.

### **Describe the fundamental challenges of requirements elicitation.**

The primary challenges of requirements elicitation often revolve around four factors: the unknown, vagueness, conflict, and change. Let's start by discussing the unknown. When a client doesn't have a clear vision of the product they desire, it can be difficult for them to properly formulate requirements for the system. As such, requirements elicitation techniques such as interviews and prototypes will need to be utilized in order to resolve this. By engaging the client in a dialogue about the system they desire, a clearer image of their product will begin to materialize.

Vagueness is both one of the most frustrating and most common parts of eliciting requirements. On one hand, when system boundaries are left to interpretation, for example, the door to feature creep and requirement bloat is flung right open. This can, in turn, lead directly to complications in staying within budget and remaining on time in the long run. On the other, when stakeholders are unclear with their requirements, a separate slew of issues arise. When it comes time to implement features and present prototypes, developers won't know what feature they're

implementing, often meaning their resulting prototype won't match the client's vision. It's important for the team to stay on top of these types of vague requirements to detect and resolve them before they cause trouble.

Conflict and change are two other challenges of requirements elicitation. When eliciting requirements from various stakeholders, it's not uncommon to run into requirements that are at odds with one another. It's up to the Business Analyst or a related requirements manager to engage these parties in direct dialogues when this occurs, such as through a workshop or focus group. This way, these conflicting requirements can get resolved in a fair, compromising manner. However, changes in requirements can be trickier to tackle. A change in an existing requirement, especially a fundamental requirement of the system, has the potential to fundamentally change the project. Documentation can need to be reworked in substantial ways depending on when this change occurs.

**Describe the specific process you took in eliciting requirements for your project in detail. Include your sources (your list of goals, your list of stakeholders, your business rules, etc.). Identify and describe the techniques you used including your interview questions and answers, observation notes, use cases, user stories, etc.**

The requirements elicitation process was a rather lengthy one. It began with a series of informal interviews with a representative stakeholder of the sponsoring company (Steven Paul, Assistant Manager). In these off-the-record interviews, we primarily discussed the framework of the project and its expectations. Information was given that pointed me towards the company's business rules, goals, and other associated policies. From there, I was able to draft a few potential requirements that I kept for myself while writing my BRS. After the completion of my BRS, I had the following information:

----- **Begin Collected Data** -----

**Goals:**

- The primary goal of the new system is to make inventory management a smoother, less costly process as a whole. Store spending on incoming shipments should go down as a result of ordering less items that are already in-stock. Store profits should go up as a result of more efficient stocking of high-demand items.
- By switching over the system to a technology-based approach rather than a manual, hand-counted approach, the inventory-keeping process will be largely streamlined. The two latter goals will be achieved by the automatic tracking portion of the system.
- Average store profits will increase by at least 5% within 6 months of system implementation.
- Weekly hours per Sales Assistant that are spent on inventory management will decrease by 1 hour within 1 month of system implementation.

**Stakeholders:**

- **Corporate (Upper Management)**
  - Will supervise development
  - Will use the reports generated to weed out unpopular items and evaluate successful items
- **Corporate (Representative)**
  - Will use the reports generated by the system to manage payment for orders
- **Store Managers/Assistant Managers**
  - Will use the system on a daily basis
  - Will use all functions of the system that sales assistants and assistant managers use.
  - Will also use the reports to make judgements on the minimums and maximums of store items and adjust them accordingly
- **Sales Assistants/Department Managers**
  - Will use the system on a daily basis
  - Will scan bought items and confirm their purchase, updating the system's stock count by doing so.
  - Will examine generated reports and bring up any important changes or trends in data to store managers or assistant managers.
- **Customers**
  - Customers will be the ones that directly affect the stock count.
  - When stock is bought, the system will have to reflect this.
  - Customer demand drives the company to improve their existing system.
- **Software Engineers**
  - Will build and develop the system
  - Will maintain and refine the system

**Business Rules:**

## 1. Confidentiality

During your employment with the company you will receive on-the-job training, experience and detailed information about the operations of the company. While performing your normal duties you may have access to confidential information concerning the Company, its customers, vendors and suppliers. Such confidential information includes, but is not limited to, information of the company's business plans and strategies, operations, assets, employees, products, pricing, costs, employee compensation, marketing plans, financial and sales information, policies, records, equipment, systems, methods, computer programs and software and trade secrets, etc.

This confidential information is a valuable asset of the Company and its unauthorized disclosure will cause irreparable harm to the Company. All employees must keep this information strictly confidential and use his or her best efforts to safeguard it from falling into the hands of any unauthorized persons.

### 1. OWNERSHIP OF ALL DATA

All data, whether voice, written, electronic, are owned by the Company.

### 1. MONITORING: Video, Phone, Voice Mail, E- Mail, US Mail, Internet, Intranet, etc.

You should expect to be monitored. All data, whether voice, written, fax, electronic, is subject to inspection and review at any time with or without notice of any kind.

Please be aware that all voice communications, all voice mail, inter-office E-mail, internet or intranet E-mail, US mail, Internet or intranet usage and other forms of communication received or initiated at our stores and corporate offices, are not private. This is true whether a particular form of communication is addressed to you or not. All e-mail, including e-mail to our remote stores, is routed through our server and is subject to periodic random review.

Violations uncovered as a result of monitoring may result in disciplinary action up to and including dismissal.

### 1. Inventory, Tasting Fairs, Special Events and Promotions

Inventories, Tasting Fairs, and other Special Events and Promotions occur throughout the year. Inventories are often done after store hours and on weekends. Tasting fairs are always on weekends. Many outside events are in the evenings or on weekends. All employees are expected to be willing to participate in inventories or any events if they are scheduled to do so, regardless of their regular job function.

**The following 2 sections are included because they tangentially relate to the Operational and Organizational Environments.**

#### Information Environment:

**Project Portfolio:** Evaluations of priority will be taken on a case-by-case basis. As the goals of the system are mostly straightforward, little complications should arise when working on the development of separate system entities. In general, any projects relating to the database's design or features should be tackled first.

**Long-Term System Plan:** The programming language chosen for the project will be either Java or Python. The database language is yet to be determined.

**Database Configuration:** Database information relating to inventory stock should be able to be viewed by any employee of the company. Internal restriction of this data is unnecessary, as stock information is not confidential. However, to avoid members of the public from viewing this data, employee ID should be used as a login to the system.

#### Business Environment:

The low-tech nature of the business' inventory system should be primarily what is taken into account. The system will be an entirely new concept to employees and management both, meaning that special care must be taken to externally document the system.

#### ----- End Collected Data -----

In the week following the submission of my BRS, I planned ahead for future elicitation by creating brief plans regarding future elicitation techniques, mockups and prototypes, and user stories. I settled on one interview with a stakeholder and a questionnaire sent out to two other stakeholders. After writing my first formal requirement for the system by referencing my mock requirements, I was able to shift my focus to eliciting user stories. I did this by requesting my stakeholder representative have himself and his coworkers write down a few user stories each. After he retrieved them, I combed over, rephrased them (retaining their meanings), and added them to my project board alongside some additional stories I created. The stories I wrote are as follows:

#### ----- Begin Collected Data -----

#### User Stories:

- As a Department Manager, I want to edit stock counts so that I can make manual corrections to any inaccurate stock counts.
- As a Sales Assistant, I want to view stock counts so that I can communicate them to inquiring customers.

- As a Store Manager, I want to edit product minimums and maximums so that I can maximize store profits.
- As an Assistant Manager, I want to view generated reports so that I can formulate plans to move struggling stock.
- As a Corporate Representative, I want to view generated reports so that I can manage orders.
- As a Corporate Executive, I want to view generated reports so that I can decide what items should be phased out of store circulation.
- As an Assistant Manager, I want to add new items to the inventory system so that I can track new products' sales performance.
- As a Store Manager, I want real-time changes to the inventory to occur so that my employees spend less time on inventory work.
- As a Corporate Executive, I want an employee login screen so that confidential inventory information is not leaked to unauthorized personnel.
- As an Assistant Manager, I want to edit product minimums and maximums so that shelves are not overstocked or under-stocked.

----- **End Collected Data** -----

The following week, I conducted my interview and sent out my questionnaires. The results are as follows:

----- **Begin Collected Data** -----

**The Interview - Steven Paul**

1. What functions are an absolute must for the system?
  - Min/max system that controls the minimum and maximum amount of product on the shelves at a given time.
  - Change min/max values (increase or decrease)
  - Login system for users – uses Employee ID and an assigned password.
  - Report generation that can be used to show best and worst sellers
  - Report archival on a store-by-store basis.
  - Real-time changes to inventory database (selling and buying)
  - Manual editing of stock (increase or decrease)
  - Automatic forwarding of reports to Corporate
  - Allow for the addition of new items to the system
1. What other pieces of your company's information network do you want the system to connect to?
  - Registers will connect to the system when items are sold (subtraction of stock) (one-way)
  - Hand scanners will connect to the system when stock is scanned by it (adding to stock) (one-way)
  - The system will connect to Corporate office to forward generated reports. Corporate will be able to view reports from stores at any time. (two-ways)
1. What security measures must be implemented?
  - Login system
    - Employees have their own login
    - Only those of the rank Assistant Manager and higher should be able to access generated reports
1. What should be automated and what should not?

**Should**

Automatically update the database when items are scanned in

Automatically update the database when items are sold

Generated reports (including any auto-filled orders) will be forwarded to Corporate.

**Should Not**

Ordering should NOT be automated.

1. What demographic do you think would have the most difficulty adjusting to the new system?

Older employees who are less electronically literate of rank Department Manager or higher. Changes for Sales Associates will be purely visual.

1. What possible feature of the inventory system would assist you most in your day-to-day work?

Report generation, as it can give Assistant Managers idea for moving inventory that's not selling via lowering their prices. It allows them to know what items to tactically phase out of the store's stock.

1. What is your biggest complaint regarding your current inventory system?

The current, manual inventory system does not reflect the needs of the store. Items that are abundantly ordered are not always accurate to what's actually selling well in the store, despite claims that orders are based on (manually documented) monthly sales.

1. Are there any accessibility features that you would want for the software?

Colorblind people should be taken into consideration when designing the interface.

1. What would you not want to change about your current inventory system?

Archival of inventory reports that can be referenced at a later date.

#### **Misc. Notes:**

Assistant Managers are the lowest rank to edit mins/maxes, view generated reports, and add new items.

Department Managers should be the lowest rank of employee that can edit stock counts.

Sales associates can simply view stock counts.

#### **The Questionnaires - Sandra Thompson and Christina Kelton**

**Note:** Small edits that retain the message's meaning but improve readability will be in brackets [ ].

##### **Sandra Thompson - Store Manager**

1. What possible feature of the inventory system would assist you most in your day-to-day work?

Theoretically, it would maintain a constant inventory level. Out of stocks would be minimal - only [occurring] in the case of warehouse/supplier outs. This would maximize sales.

1. How do you think your store would benefit from the automated inventory system?

It would free up the associates that are currently spending ten plus hours per week [on] ordering. This time could be spent on customer service, product education and store conditioning which would increase sales.

1. What is your biggest complaint regarding your current inventory system?

Human error. Currently associates have to physically scan each item that needs to be ordered. Items are missed, product tags fall off the shelf or are removed by an associate so the product does not get reordered. And, occasionally, the associate forgets to hit the send button so the order is not received for that cycle.

1. Are there any accessibility features that you would want for the software?

The ability for authorized staff to go in and adjust the min/max level of product inventory. It would also be nice if the system auto reordered the products that did not come in due to warehouse/supplier outs for an indefinite time period until removed by management. (I have worked with systems that only reordered twice and then dropped the product.) The ability to program in ongoing special orders for customers would be a desirable feature as well.

**(Note from Adam:** This question was misunderstood, but is still valuable feedback in regards to the login feature of the system.)

1. What would you not want to change about your current inventory system?

The ability to do special orders for customers. In some cases it is a one time only order and other times it is a consistently timed order.

**(Note from Adam:** This refers to a special order requested by a customer.)

##### **Christina Kelton - Department Manager**

**Note:** Since Christina responded to my questions in a free-form manner, I will be fitting information from her answers to each question. These will all line up exactly with the contents of her email, but will be rephrased to make them more concise and readable.

1. What possible feature of the inventory system would assist you most in your day-to-day work?

The ability to see how a particular S.K.U. (product) is leaving my inventory through the handheld scanner would be particularly helpful. This could be done through a setting on the scanner or by manually entering notes on particular shipments in the inventory system.

2. How do you think your store would benefit from the automated inventory system?

The store as a whole would benefit greatly from an auto-fill system, reducing the amount of hours spent on inventory management and ordering weekly. However, it would do little to affect my department in particular. As the manager of a supplements department, our market is rather unpredictable. Factors such as social media influencers and trends can wildly change public perception of our items at the drop of a hair, making for a volatile market.

3. What is your biggest complaint regarding your current inventory system?

In the current system, I'm unable to tell how the item entered inventory in the first place. Information such as whether the item was bought for us by the company or re-ordered by an employee is completely lost. This makes prioritization difficult upon re-ordering.

4. Are there any accessibility features that you would want for the software?

No, I'm unable to think of any possible accessibility features I'd request.

5. What would you not want to change about your current inventory system?

The current inventory system, while basic, provides a strong foundation for any improvements. I would rather the core functionalities of the system remain the same, with only the process being automated or improved.

----- End Collected Data -----

The following week, it was time to synthesize the information gained from my interview, the questionnaires, my BRS, and my Operational Concept by writing my first set of formal requirements. These requirements were, for the most part, based heavily on direct feedback from stakeholders. They were later verified to be accurate by the company representative. They are as follows:

----- Begin Collected Data -----

#### Requirements:

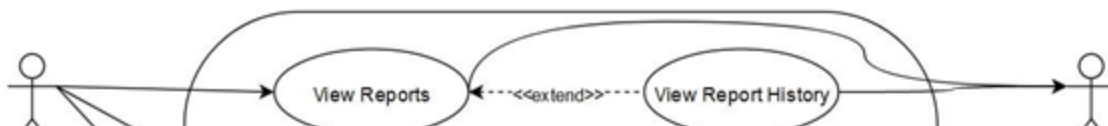
- When a cashier confirms a customer's purchase of an item, the system shall decrease the item's current stock count by an amount equal to the quantity purchased within three seconds.
- When an employee scans a non-new inventory item in, the system shall increase the item's current stock count by one within three seconds.
- The Report Archival system shall display generated reports in order from newest to oldest.
- The Report Generation system for a given store shall generate daily reports with auto-filled orders one hour after the system's respective store closes.
- The system's Employee Login feature shall require an employee ID and employee-chosen password to grant device permissions to employees.
- When an item's stock count reaches the item's minimum value, the system shall auto-fill an order form that will bring the stock count back up to the item's maximum.
- If the user is at minimum the rank of Assistant Manager, the system shall allow the user to change the values of product minimums and maximums while logged in.
- If the user is at minimum the rank of Assistant Manager, the system shall allow the user to add new items to the inventory system while logged in.
- The system's Add Items feature shall require a S.K.U., price, stock count, and in-store location when adding new items.
- System-generated reports shall detail daily profit, daily sales, items sold, and auto-filled orders.
- Sunday at 11 PM, the system shall generate a weekly report that utilizes the prior seven daily reports to detail the week's profit, the week's sales, the items sold that week, and the orders filled that week.

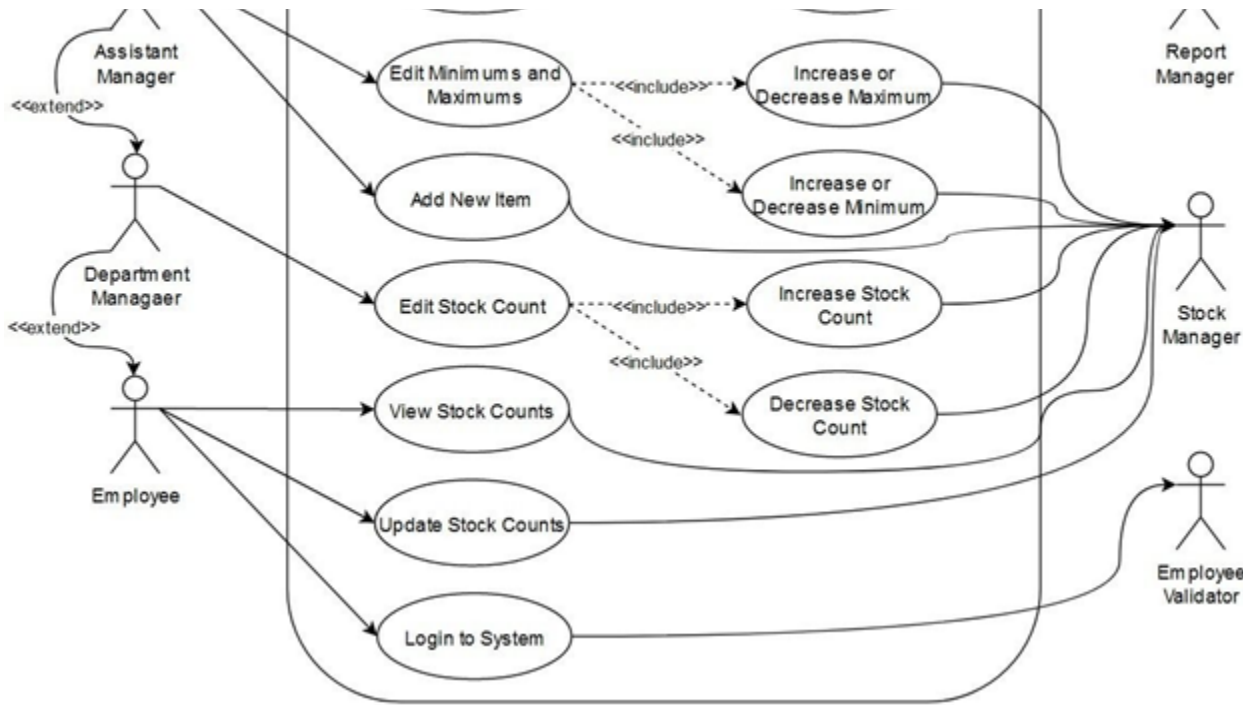
----- End Collected Data -----

The following week, it was time to synthesize user stories and requirements, turning them into use cases and a use case diagram. As I don't want to inflate the size of this report further, I'll simply include the use case diagram, which includes the use cases of the system within. The use case diagram created, including all the use cases of the system, is as follows:

----- Begin Collected Data -----

#### Use Case Diagram:





----- End Collected Data -----

The requirements elicitation process (up until now) concluded with the creation of my mockup. My mockup is available for download at [this link](#). The mockup was used as a means to verify my understanding of the requirements and elicit further ones. My stakeholder representatives' suggestions, which will be the conclusion of the report, were as follows:

----- Begin Collected Data -----

#### Mockup Suggestions:

One suggestion was that important heading text have borders to emphasize their importance. This was mainly a change for readability's sake. Additionally, he suggested that an undo button be implemented for the "Edit Stock" function. The main purpose of this button would be to reduce human error that could result from typos or misclicks. Last but not least, he suggested that I add a question mark icon button that would serve as a "help" button. This button would bring up more information on the current tab, meaning that new users would be able to either learn more about the system or refresh their memory on what they knew.

----- End Collected Data -----

----- End of Report -----

## Analysis Report

### In General (HOTS)

#### *Explain why requirements need to be analyzed after being elicited.*

Requirements elicitation, if you remember, primarily concerns the sources of requirements and the methods that can be used to elicit them. After user stories have been written and requirements have been elicited, the end result is often an unrefined, imprecise, and sometimes conflicting set of requirements. The requirements analysis phase is wherein these requirements are refined, categorized, and resolved.

The requirements analysis phase places a heavy emphasis on ensuring that your stakeholder, system, and software requirements are well-formed, follow appropriate guidelines (such as language criteria and the MITRE checklist), and do not conflict. This process is critically important, as it effectively ensures that the development team, the client, and any other stakeholders all have the same understanding of the system or solution at hand, leading to overall increased satisfaction. This phase is where items or functions are officially categorized as "out of scope" or "in-scope." In addition, conceptual modeling figures such as use case diagrams and context diagrams are created. These assist both the developers and the stakeholders by creating intuitive visual diagrams that display the system's full functionality and illustrate data flow respectively.

**Describe how requirements are analyzed for a software project in general. List and summarize in your own words the techniques in SWEBOK Ch. 1 section 4, especially requirements classification and conceptual modeling.**

The primary requirements tasks that occur during requirements analysis are requirement classification, conceptual modeling, architectural design and requirements allocation, requirement negotiation, and formal analysis.

Technique	Description
Requirement Classification	<p>Requirement classification is the stage wherein each requirement in the set of requirements is given a set of labels. This is primarily done to organize the project as a whole - both for future requirements phases and to avoid development conflicts due to requirements-based misunderstandings.</p> <p>Some examples of labels decided at this point are process vs product, nonfunctional vs functional, and priority. Product vs process defines whether the given requirement is a constraint that's placed on the product itself or the development process concerning it. Nonfunctional vs functional determines whether the given requirement describes an action the system actually performs or a quality which the system must possess. The priority is a simple concept to understand - it decides when a given requirement will be developed relative to others in the set.</p> <p>Requirements classification is a crucial step to take for any software project due to the implications it has on the project as a whole. A set of unclassified requirements can be effectively useless, especially as the set of requirements grows in size. For example, if you don't prioritize requirements, then your project is highly susceptible to any development issues that could occur. This could, in turn, lead to missed deadlines and incomplete products, as developers can end up spending all their time on low-importance features instead of fleshing out high-importance features that the system can't function without.</p>
Conceptual Modeling	<p>Developing conceptual models assists greatly in establishing the operational environment and organizational environment of the product. They excel at providing a visual aid when it comes to the environment the system operates in, the external entities the system interacts with, and the larger entity that the system belongs to.</p> <p>Some examples of conceptual models that are developed during this phase are the use case diagram and the level 0 context diagram. The use case diagram defines requirements in terms of user interactions with the system, meaning they're particularly valuable for conversations with stakeholders. The level 0 context diagram revolves around the concept of illustrating the data flow within the system. The system stays in the center, while external entities are placed around it. The diagram shows how data flows from one part of the system to another, with each external entity having the potential to both send and receive information to and from the system, respectively. As such, level 0 context diagrams are fantastic for gaining an understanding of the operational environment of the system, including its interfaces with external entities.</p>
Architectural Design and Requirements Allocation	<p>The architectural design activity primarily concerns analyzing the requirements to understand what they necessitate, both hardware-wise and software-wise. This is where many crucial elements such as database language, programming language, and frameworks for the project are decided. This activity works directly with conceptual modeling - particularly the level 0 context diagram that defines external entities and data flow.</p> <p>The requirements allocation activity ties in directly with the architectural design activity. At this point, requirements are allocated into different "components" of the system that interact with other components of the system. A new round of analysis usually occurs once these components are created, as it has to be determined whether a given requirement belongs to component A, component B, or sometimes both (though this usually indicates that a requirement isn't specific enough).</p>
Requirement Negotiation	<p>This stage of the process is often known as the conflict resolution stage. During analysis, it's somewhat common for conflicting stakeholder requirements to be found. It's during this activity when</p>



	stakeholders with conflicting requirements must make compromises for the sake of the system. The software engineer must serve as a mediator for these parties and avoid making rash decisions for either side that could breach contracts.
Formal Analysis	<p>Formal analysis involves the editing of existing requirements to use formally defined semantics. Formal analysis has two primary goals: to avoid misunderstood requirements by rewriting them to be precise and unambiguous, and to reason over requirements until their desired properties can be proven.</p> <p>However, formal analysis is difficult to complete until several items, such as the business goals and user requirements, have been fully shaped. This is due to how formal analysis is essentially “finalization” (used loosely to allow the addition of more requirements) work when it comes to a set of requirements. As such, it’s beneficial to complete this activity last in the sequence of potential analysis activities.</p>

## For Your Project (LOTS)

### Summarize how you *applied* the requirements analysis techniques for your project.

In my project, **requirement classification** was applied when I completed *Activity 6.3.3.6 Analyze Stakeholder Requirements* and *Activity 6.4.3.4 Analyze system/[software] requirements, make prototype*. In these assignments, I had to assign a series of priorities and labels to my requirements, as well as link them to user stories. I assigned priorities from lowest to highest and labels such as process/product, functional/nonfunctional, database, system, and more in order to classify my requirements!

**Conceptual modeling** was done when I created my use case diagrams and my level 0 context diagram. I used my use case diagram to illustrate user interactions with the system, categorizing each user by their respective permissions. In my context diagram, I was able to fully capture the user types and databases that would interact with my system, as well as the data that would be exchanged in their interactions with the system.

When it comes to **architectural design and requirements allocation**, this mostly took place when writing my system/software requirements.

In my case, I finished the level 0 context diagram before I made the system/software requirements for my project. As such, I was able to use the model as a base for my database, user interface, and system requirements. This, in addition to my constraints created at a prior date, were what primarily comprised the architectural design portion.

The requirements allocation portion took place during the same time. Many of my system/software requirements belonged to certain components of the system. As such, I was able to allocate them rather easily.

**Requirement negotiation** was not too present in my project. Aside from a minor issue wherein a mobile app requested by a stakeholder was declared out of the scope of the project, I was rather lucky when it came to stakeholder conflicts. The issue with the mobile app was quickly resolved, with the stakeholder involved agreeing with other stakeholders and myself when we mentioned that it was a feature that could be implemented at a later date.

**Formal analysis** took place during the *Activity 6.4.3.4 Analyze system/[software] requirements, make prototype* assignment. In this assignment, I first had to justify how my set of requirements fit the characteristics of a set of requirements specified in 29148. This fulfilled the second goal of formal analysis. After doing so, I had to then thoroughly review my requirements to ensure that they used formally defined semantics by using the MITRE checklist.

### Include evidence of having *analyzed* your requirements by classifying them as on the product or process, being functional or non-functional, and / or prioritizing them by including a Jira table.

Key	Summary	Labels	P
C3P-52	The migration process of the system shall occur on the night of October 9th, 2021, with employees of the minimum rank of Assistant Manager staying late to copy the current state of their store's inventory over.	nonfunctional	↑
C3P-51	If a logged-in Employee logs out, the system shall clear the fields in the Inventory, Add Items, and Reports tabs before returning the Employee to the login screen.	interface, nonfunctional	↓
C3P-50	If an Assistant Manager logs in, the system shall display the same UI elements outlined in C3P-49, the minimum and maximum editing function of the Inventory tab, the Add Items tab, and the Reports tab that allows for the viewing of archived reports.	interface, nonfunctional	↓
C3P-49	If a Department Manager logs in, the system shall display the same UI elements outlined in C3P-48 and a stock quantity	interface,	↓

	editing box in the same tab.	nonfunctional	
C3P-48	If a Sales Assistant logs in, the system shall only display the Inventory tab and the tab's searchable database.	interface, nonfunctional	↓
C3P-47	The system shall display an error screen and prevent further action until re-connection if the system drops a connection with any of the three databases specified in C3P-43, C3P-44, and C3P-45.	database, functional	↑
C3P-46	The system shall display an error screen and terminate the program if the system cannot establish a connection with the Employee Database, the Inventory Database, and the Reports Database on program startup.	database, functional	↑
C3P-45	On program startup, the system shall connect to a Reports Database for the purposes of sending new generated reports and loading archived reports.	database, functional	↑
C3P-44	On program startup, the system shall connect to an Inventory Database for the purposes of loading inventory items, increasing and decreasing inventory items' stock count, and adding new inventory items.	database, functional	↑
C3P-43	When an Employee logs in, the system shall connect to an Employee Database to verify that the user's employee ID and password exist in the database, then granting the permissions corresponding to the Employee's role.	database, functional	↑
C3P-34	Sunday at 11 PM, the system shall generate a weekly report that utilizes the prior seven daily reports to detail the week's profit, the week's sales, the items sold that week, and the orders filled that week.	functional	↑
C3P-33	System-generated reports shall detail daily profit, daily sales, items sold, and auto-filled orders.	nonfunctional	↑
C3P-32	The system's Add Items feature shall require a S.K.U., price, stock count, and in-store location when adding new items.	nonfunctional	↓
C3P-31	If the user is at minimum the rank of Assistant Manager, the system shall allow the user to add new items to the inventory system while logged in.	nonfunctional	↓
C3P-30	If the user is at minimum the rank of Assistant Manager, the system shall allow the user to change the values of product minimums and maximums while logged in.	nonfunctional	↑
C3P-29	When an item's stock count reaches the item's minimum value, the system shall auto-fill an order form that will bring the stock count back up to the item's maximum.	functional	↑
C3P-28	The system's Employee Login feature shall require an employee ID and employee-chosen password to grant device permissions to employees.	nonfunctional	↑
C3P-27	The Report Generation system for a given store shall generate daily reports with auto-filled orders one hour after the system's respective store closes.	functional	↑
C3P-26	The Report Archival system shall display generated reports in order from newest to oldest.	nonfunctional	↓
C3P-25	When an employee scans a non-new inventory item in, the system shall increase the item's current stock count by one within three seconds.	functional	↑
C3P-24	When a cashier confirms a customer's purchase of an item, the system shall decrease the item's current stock count by an amount equal to the quantity purchased within three seconds.	functional	↑

21 issues

Include evidence of having *evaluated* your requirements by comparing them against the characteristics of a set of requirements from 29148 and / or the Characteristics of Good Requirements from MITRE.

#### Characteristics and Justifications

Characteristic	Justification
Complete	The requirements statements in the set describe functions, quality aspects, or performance aspects for exactly one facet of the system per statement.
Consistent	The only 3 requirements in this set that could be argued to "overlap" all have features that are specific to themselves, they just borrow from lesser-permissioned statements. Terminology is consistent. All items can be measured either visually or via time (seconds, minutes, hours, etc.)

Feasible	All requirements are feasible to complete by October 1st, 2021.
Comprehensible	All requirements in the set are clear and specific, with care taken to remove ambiguous wording.
Able to be Validated	Validation conditions are built into each requirement.

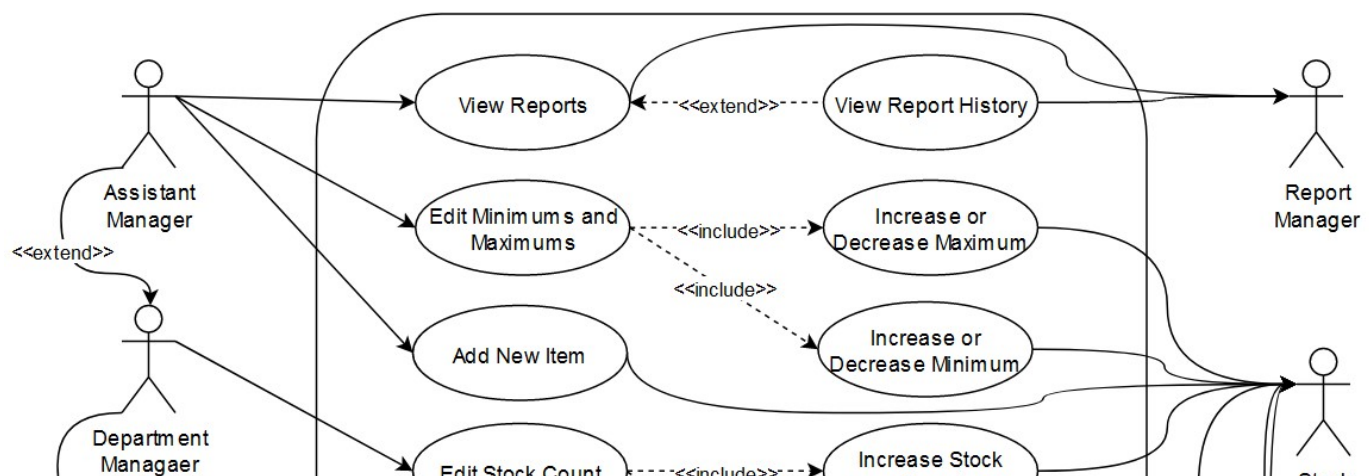
#### MITRE Checklist

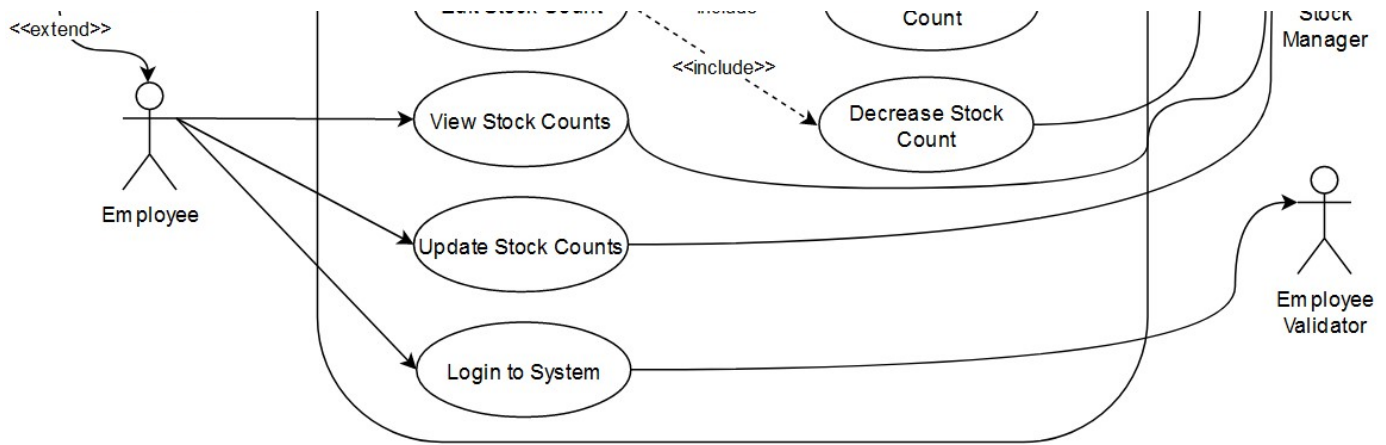
Checklist Item	Complete?
The system-level technical requirements are traceable to the user requirements.	<input checked="" type="checkbox"/>
Each system requirement describes something relevant: a function the system must perform, performance a function must provide, a constraint on the design, or a reference such as to an interface definition.	<input checked="" type="checkbox"/>
The level of detail that the requirements provide about system functionality is appropriate.  The requirements are sufficient to describe what the overall system must do, what its performance must be, and what constraints an engineer should consider. There are few requirements that specifically affect the design of only one component of the system. The major requirements drivers (e.g., those stressing the design) and associated risks should be identified	<input checked="" type="checkbox"/>
The requirements include any legal or regulatory constraints within which the system must perform.  <i>Example: There may be restrictions on the use or quantity of certain hazardous materials in a system.</i>	<input checked="" type="checkbox"/>
The requirements include enterprise architecture constraints within which the system must integrate (or toward which the system is desired to migrate). Requirements include appropriate open systems and modularity standards.  <i>Examples: DoD Net-Ready requirements, modular open system architecture concepts, Electronic Systems Center strategic technical plan goals.</i>	<input checked="" type="checkbox"/>
Environmental design requirements are specified.  <i>Example: A control unit may be in a controlled office environment and the other major components may be outdoors, thus two environments must be defined and associated with the functionality operating in each environment.</i>	<input checked="" type="checkbox"/>
All external interfaces for the system are included. Major internal interfaces may also be included if they are important to system modularity, or future growth in capability.  These may include physical (mechanical fastening, electrical wiring, connectors), functional (mechanical stress transfer points, cooling, power sources, antennas, wire message formats, data exchanges), and software (software interface specifications, library calls, data formats, etc.).  Remember that an internal interface between two subsystems that use a transport mechanism that is not part of the system is a hidden external interface. For example, two subsystems that communicate internally with each other over a sensitive but unclassified network as the internal interface (the data exchanged between them) and an external interface (the wiring and internet protocols to enable the data exchanges with the network).	<input checked="" type="checkbox"/>
Requirement statements use the word "shall" or "should."  The word "shall" has meaning in contractual language and is enforceable legally. Other words like "will," "may," "should," and "must" may show intent but are not legally binding in contracts. In some situations, it may be desirable to use "should" to show the government's intent and preference while at the same time allowing flexibility and latitude.  <i>Example: "The system shall have a mean time between failures of greater than 500 hours."</i>	<input checked="" type="checkbox"/>
Requirements statements are unambiguous.  Terminology is clear without the use of informal jargon. Statements are short and concise.	<input checked="" type="checkbox"/>  (If a requirement is on the longer side, I deemed it necessary for the purpose of retaining clarity.)
Performance requirements statements (including logistics/sustainment/support) are quantifiable, testable, and/or verifiable.  Avoid the phrase "shall not." It is very difficult to prove a negative.	<input checked="" type="checkbox"/>

<p>Avoid qualitative words like “maximize” or “minimize.” They force an engineer to judge when the design is good enough. The user may think that the engineer did not “minimize enough” and get into a legal argument with the contractor.</p> <p>Note: Every user requirements document includes: “the system shall be easy to use” requirement. Talk to other MITRE staff for examples from other projects and seek out a human factors specialist for requirements wording that is suitable both for specifying these requirements and methodologies for verifying them.</p> <p>Avoid specific, one-point values when defining requirements. Use ranges (minimum of, more than, less than, maximum of, within, etc.) to accommodate appropriate interpretation. Using a single point value may cause arguments if the system is tested at that exact value only, or if a test appears to be successful from an intent perspective, but does not meet the exact value stated in the system requirement.</p> <p><i>Example: The system shall process a minimum of 100 transactions/sec.</i></p> <p><i>Example: The system shall be operable up to and including 30,000 ft.</i></p> <p><i>Example: The system shall operate in temperatures between 5 and 35 degrees Celsius.</i></p>	
<p>If objective performance values are included as goals, ensure they are clearly identified and distinguished from firm requirements.</p> <p>User requirement documents refer to threshold requirements (those that must be provided), and objective requirements (better performance has value to the user, but not above the objective requirement).</p> <p><i>Example: The system shall detect and display up to 100 targets within the surveil-lance volume with a goal of detecting and displaying up to 125 targets</i></p>	✓
<p>The operational and support environment is described and defined.</p> <p><i>Example: The system shall be maintainable by an Air Force level 5 technician.</i></p> <p><i>Example: The system shall be reparable while in flight.</i></p>	✓
<p>The requirements include appropriate use of Government and industry specifications, standards, and guides.</p> <p>Only include them if they are relevant and ensure that the correct version is listed in a list of reference documents.</p>	✓
<p>Verification approaches for all system performance and sustainability requirements are complete and appropriate.</p> <p>Every requirement must have a verification method identified.</p> <p>If a requirement cannot easily be verified by a direct inspection, measurement, or one-time demonstration of the requirement, the verification requirement should include an expanded test criteria description to ensure that there is no disagreement later in the program. This can include describing the number of trials, statistical criteria to be used, conditions of the test such as simulated inputs, etc.</p>	✓

Include evidence of having *created* a use case diagram and context diagram.

## Use Case Diagram





**Note:** The actors on the right are components of the system and not human entities.

**Note:** Use of the <<extends>> tag for actors was taken from [this provided resource](#).

**Note:** Left actors are labelled by minimum rank for the permissions they have

## Context Diagram Background

A Context Diagram or, more descriptively, a Data Flow Diagram, is one of the many visual aids used in software development. Their primary purpose is to serve as a visual representation of how data flows to and from the system via external entities. They should be simple enough for any stakeholder of the project to understand, yet detailed enough for it to be clear how data flows to, from, and within the system. A given context diagram is made up of three key components: a black circle signifying the system itself, black boxes signifying external entities (human or non-human) that interact with the software, and directional lines with accompanying text that illustrate data flow and describe the data that is flowing.

To create this context diagram, I pulled up Notepad, my project notes, and my stakeholder feedback and asked myself, "In what ways does data flow to and from the system?" Immediately, I thought of the several databases that would accompany the project - I had one that would be used for the login tab, one that would be used for the stock-related tabs, and one that would be used for the report tab. I jotted these down and thought further before the more intuitive answer hit me - the database will need to display data to and receive data from the three primary user types associated with it. I jotted all 6 of these cases down, then roughly outlined their interactions (input/output) with the system. From there, I simply opened up <http://draw.io>, my diagramming site of choice, sketched out the diagram using the entities I'd thought of, and fleshed out the individual interactions that occur in a clear and concise manner.

## Level 0 Context Diagram

**Note:** Unfortunately, draw.io has a tendency to absolutely destroy kerning. My apologies on that.

**Note:** The user connections within the context diagram are defined by the role's permissions. For example, Assistant Manager's permissions would be the exact same as someone higher, such as a Corporate Representative.

**Describe how you have applied critical thinking to your requirements, such as by using the Elder Paul Critical Thinking Model. A good strategy would be to identify how each Element of Thought helped you to think about your requirements, ideally along with a specific example or anecdote from your project experience.**

Elder Paul Element of Thought	How This Element Was Applied
Point of View	While constructing my requirements, I had to constantly consider the points of view of the system's separate users and their own sets of wants and needs. Though many of these perspectives were illustrated through user stories, some had to be formulated on my own.
Purpose	When analyzing my requirements, I always had a list of my business rules, goals, and objectives up. Every time I checked a requirement, I would always bring this list up and ensure that it fulfilled at least one item in one of the three lists. This process, while simple, streamlined my analysis greatly and increased its effectiveness.

Question at Issue	When writing my requirements, I would often consider first the problems that pertain to Earth Origins' current inventory system. For a given problem, I would often write a requirement down, then immediately stop and analyze it. I would ask myself questions such as, "is this relevant to the problem at hand," and, "is there any other problem that this requirement addresses?" If the answers were yes to the former and no to the latter, then I knew I likely had a strong base form of a requirement!
Information	<p>When thinking of a more abstract requirement, I would immediately question whether it were feasible. By cross-referencing the company's resource allocations, I was able to make decisions as to whether these requirements were in scope or not.</p> <p>A perfect example was a beta requirement I had constructed for a mobile application. This was, in fact, a feature that was requested by one of the stakeholders. However, upon consulting the concrete data I had - the budget and time constraints - I knew this was unrealistic and decided to label it as out of scope, but potentially able to be developed down the line.</p>
Interpretation and Inference	<p>In most cases, the results of the inferences made were reflected the requirements themselves. My set of requirements is, at its core, built to solve a series of problems. As such, any inferences I had lead to the development of solutions.</p> <p>For example, the lack of automation for item scanning was a problem. However, the employees already had access to wireless tools that could help them automate the process - hand scanners. By connecting these two pieces of information, I was able to formulate a solution - C3P-25.</p>
Concepts	<p>The most obvious concepts and rules that I applied during the writing of my requirements were the SMART acronym and the requirements language criteria outlined in 29148. This meant the rewriting of several of my requirements after my initial draft to narrow down their scope and utilize proper, precise language.</p> <p>C3P-48 through C3P-50 are perfect examples of this in action. The use of the words "every" and "or" are particularly easy traps to fall into when describing roles that inherit the permissions of lower ranking roles.</p>
Assumptions	<p>To handle any assumptions I made, I simply asked myself whether a given requirement had some sort of precondition. If so, I ensured that said precondition was stated at either the start of the requirement or the end of the requirement.</p> <p>Many requirements in my set, such as C3P-48 through C3P-51, are prime examples of this practice in action.</p>
Implications and Consequences	<p>When writing requirements, I made sure to ask myself the question of what the outcome of the requirement would be. Would the state of the system change? Would information be transferred somewhere else? Could an error occur during a process that would result in a system malfunction? These questions were kept in mind and considered extensively during the process of writing my requirements.</p> <p>In some cases, such as during the writing of C3P-43 through C3P-45, these questions would lead to extensions of old requirements that lead directly to new ones. In the case of C3P-43 through C3P-45, C3P-46 and C3P-47 were prime examples of these extensions.</p>

**Describe how you recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts.**

In both regular and high-stakes engineering situations, I think it's extremely important to utilize the critical thinking skills that we, as engineers, develop on a daily basis. Far too often in the real world, engineers opt to use the, "quick" solution that solves a problem in a way that's convenient to them without considering the long-lasting consequences of their decision. More often than not, this has led to disastrous consequences for all parties involved - the engineers, their company, and the people whom their mistake impacted. When put in a particularly challenging situation with a given project, I think it's crucial to carefully consider the factors that put you in the position you're in and carefully the methods at your disposal to handle it.

For example, say you're a software engineer in charge of a project to build a brand new piece of security software. You're less than a week from launch, but one of your final test cases just unveiled a major security threat that, if exploited, has the potential to leak the personal information of all your customers. Though they think it's unlikely it'll be found soon after launch, the potential for it to happen still exists. However, one of your developers says they have a solution that could lessen the blow overall, but still leaks multiple parties' information. You're presented with two options: accept this solution or delay the project.

In this sort of situation, it's critical to not panic. Thinking critically, you should begin to ask for more information on the issue - where did it originate, what systems does it affect, what circumstances led to it being discovered. Depending on the answers you receive, you could call an emergency meeting with your team and any available stakeholders to discuss the matter with them. In addition to considering their input, you should also consider how a resulting scandal could affect the reputation of your company, your team, and the users of the system who had their



data stolen. If no 3rd solution is found that creates a win-win scenario, then the engineer will have to think critically as to what the best course of action is. However, at the end of the day, it is my opinion that it's an engineer's responsibility to not willfully endanger the people they're developing these projects for.

When it comes down to it, engineering is a profession that, at its core, is intended to revolve around the concept of problem solving. Big or small, these "problems" are going to be what the average engineer spends the majority of their career dealing with. However, if your solution to a problem ends up creating larger problems than the one you started with, it cannot truly be called a solution to a problem. Despite this, I believe that critical thinking is a key component to recognizing these difficult ethical situations and paving the road to effective solutions to these problems.

## Specification Report

### Describe the content, audience, and purpose of the BRS, StRS, SyRS, and SRS in general.

The **Business Requirements Specification (BRS)** is a document primarily concerned with outlining the high level business goals of the project at hand. It's developed in the pre-planning phase, typically by both a Business Analyst and the business itself. Within the BRS, the initial concept for the proposed system is suggested alongside the sponsoring business' purpose of development, their business goals and objectives, the business operation's modes and models, and more. Stakeholders are also considered, with their perspectives and needs being reflected in certain sections of the document. In addition, the first groundwork for proper software requirements is laid via avenues such as stakeholder needs, business requirements, organizational requirements, and business rules. All in all, the intent of the BRS is to outline what the business plans to change about its current model, and how the proposed system will benefit the business itself.

The true importance of the BRS becomes more evident when you consider its ramifications on the rest of the software project. In essence, the BRS serves as a contextualizing agent for the problem at hand, clearly outlining the purpose of, organizational model of, and environment of the proposed system. It's one of the most foundational documents in the requirements engineering process, making it invaluable as a reference for its target audience as the project continues. The Business Analyst, who often authors it, can utilize it to engage in conversations with the sponsor business, who can verify and edit the document. System analysts can analyze the business model and the proposed system to identify technical solutions to the problems faced. Lastly, system or software engineers developing the SyRS or SRS can derive limitations, constraints, and certain requirements from the details provided by the sponsor company.

The **Stakeholder Requirements Specification (StRS)** is a stakeholder-oriented document with the primary purpose of describing how the organization will utilize the system to contribute to the business. Additional purposes include descriptions of the business' motivation for the project, the expression of stakeholder needs in unambiguous fashions, and the definition of processes and policies concerning the project. Due to their overlap in purpose and key content, this document can sometimes be combined with the BRS based on the environment of the user. In contrast to the BRS' heavy focus on the business and Business Analyst, the StRS' elements are specified by the stakeholders of the software project. These elements include organizational requirements, business requirements, and user requirements. The StRS sets the basis for stakeholders' involvement by mediating between them in order to achieve a consensus. It also allows for system or software engineers to extrapolate stakeholder requirements for the SyRS or SRS.

The **System Requirements Specification (SyRS)** is, effectively, a document made with the intention of communicating the business' requirements to the individual or team of technical experts that will then specify and build the system. This, naturally, means that the set of requirements linking the two entities together must be well-formed enough that they can be understood by both parties. Additionally, the SyRS seeks to describe the domain's perspective of the high-level system requirements, as well as contextualizing information regarding the system's objectives and its target environment. This includes a statement of the system's constraints, assumptions, and non-functional requirements.

The content of the SyRS can be quite variable, only requiring a from appropriate for the system's intended rules. As such, conceptual modeling a popular method through which SyRS creators construct their documents. Such of these models include but are not limited to paper documents, models, prototypes, and non-paper documents. So long as these forms can be traced to a common source of systems requirements information, the choice is left to the systems analyst or software engineer creating the document. Upon its completion, the SyRS should define the intended function of the system, its environment, its usage profile, its performance parameters, and corresponding effectiveness and verification activities.

Of the documents we've discussed so far, the **Software Requirements Specification (SRS)** is the most all-encompassing of the bunch. Its purpose is to serve as a specification for a software product, program, or a set of programs that performs specific functions in a certain environment. Being written by a representative of the acquirer, a representative of the supplier, or both, the SRS takes on the mountainous task of defining all the required capabilities of its specified software product. This includes everything from the product's name and overview, to its use cases, and even its (usually lengthy) list of requirements. If the BRS is the document that answers the, "why," of the system, the SRS is the document that answers the, "what."

Due to its all-encompassing nature, the SRS also has the largest target audience of any of the documents discussed so far. Stakeholders can review it for the purpose of validating the document's accuracy and other specification elements. Developers and team managers can utilize it as a basis for coding the software product it's describing. QA testers can design tests around the specifications of the system outline. Additionally, operations, maintenance, and project consultants must reference this document in order to ensure that the performance of their job tasks does not conflict with the specifications laid out within.

### Summarize your experience of creating your BRS and SRS.

#### Creating the BRS

The creation of my BRS, despite being so early on in the project, was one of the most formative undertakings of the semester. After obtaining a formal sponsor through my Dad (Steven Paul, Assistant Manager of a Earth Origins location), I had to request he obtain information regarding the system's concept from his company. This involved discussing the topic with his co-workers to obtain their needs. It took a few days before he felt he was fully prepared to answer any questions I had. Once he was ready, we blocked out a night to sit down together and go through the whole BRS together.

Before our informal interview, he provided me with information on company policies, procedures, and the like through a digital copy of the company manual. Prior to our meeting, I made sure to comb through this manual thoroughly, taking note of any apparent business rules that I thought might be applicable to the vague concept of the project that had been described to me until that point. Additionally, I did some detailed research on the company itself, attempting to dig up as much information as I could to be maximally prepared for my interview.

On the night we sat down together, we formulated a process as to how we would handle the questions posed. First, I would read through the informational item, process it, translate it to layman's terms, and then recite it to him. From there, we would have a discussion regarding the question, creating a back and forth dialogue with no time limit. Once we had reached a conclusion, I would then write the information I had obtained down. This process worked wonderfully, allowing us to fill out most of the BRS without issue.

However, some parts of the BRS were easier than others. After the aforementioned process was complete, we only had the diagramming and the business goals and policies sections to fill out. Picking out relevant information from the company manual had proven to be difficult, so I had decided to tackle items relating to it after we had handled the rest. Once we had reached this phase, I decided to show my Dad my list of items I'd plucked from the company manual. After discussing each one with him, we trimmed down my own list slightly (mainly due to my own misunderstandings of the rules or policies). After this, we finished up by covering the diagramming portions. This was a fairly straightforward process - nearly identical to the back-and-forth dialogue format I mentioned earlier - that was only left for last due to the time it took to conceptualize and create the diagrams. After this, I simply had to format and submit my document!

## Creating the SRS

As this was a much more prolonged and less clear process than creating my BRS, I'll simply choose to cover the process of going through the 5 activity assignments labelled as SRS activities.

The first assignment was Activity 6.3.3.2, an assignment in which I was tasked with defining the user characteristics of my project. The first of said tasks involved defining the roles that each stakeholder of the project would play within the project itself. This was a fairly simply task, as I was able to reference my BRS' notes on my stakeholders as well as my Dad. This conceptualization of my stakeholders would later impact how I defined the individual user roles of the system. After this, I briefly defined my user characteristics, stating that not too many people, aside from the blind, would have issues with the system, as it required no new technology or external interfaces to operate. I concluded the assignment by outlining my elicitation plan by listing the questions I would use to elicit requirements, stating the medium through which I would develop the prototype of my system, and creating a method to develop user stories.

The second assignment for my SRS was the ever-crucial Activity 6.3.3.4, wherein I developed the Operational Concept. This assignment began with the creation of my purpose statement, a paragraph-long description of why the software solution I was developing was needed. Directly after came the scope section, where I finally declared the name of my product to be the Inventory Control Ordering System (ICOS), summarized its role and goals, and set the limits of the system by declaring what was out of scope. This assignment then concluded with a listing of both my sponsor's and my own assumptions, descriptions of the major functions of my inventory system, and several limitations that I had found through my own research and analysis of the project.

The third assignment, Activity 6.3.3.5, was the first assignment wherein I had to express the requirements of my system. For these 10 requirements in particular, I had to adapt the stakeholder requirements that I had elicited via interview, questionnaire, and user stories into full-blown, formal requirements. This was a long and arduous process, as I had to adhere to the strict language guidelines and characteristics of good requirements set forth by 29148. After this was done, I concluded the assignment by spending some time formulating stakeholder constraints for my system.

The fourth assignment, Activity 6.3.3.7, had me create a mockup for the system that I had developed. This was a static, navigable user interface that demonstrated visually the look and feel of the program. This assignment was what allowed me to learn Adobe XD, a fantastic mockup and prototyping software with a boatload of useful plugins. The assignment had me consider plenty of nuanced concepts such as designing UI around accessibility, designing for user retention, and designing around stakeholder-placed constraints on the UI. After this assignment was completed, I was finally able to demonstrate a good-looking proof of concept to my sponsor, who had patiently supervised my project.

The last assignment, Activity 6.4.3.2, had me define the functional boundaries of my system. Though this wasn't required at the time, this was also the point at which I decided to create a Level 0 Data Flow Diagram, also known as a Context Diagram. This diagram helped me immensely with understanding the ways in which data could flow from component to component within my system. For every piece of data that would flow from my system, I had to ensure that there was some way in which the system would initially receive said data. This involved plenty of consideration of what the external entities that interacted with my system were, as well as what pieces of data they stored and sent. I felt that this assignment was particularly useful in helping me formally recognize the databases I needed for my project. It felt as though the last piece of the puzzle I'd needed to complete my sponsor's vision of the project had been found.

Some additional activities that played major roles in my SRS' completion were the creation of my use cases, use case diagram, and requirements validation techniques. Creating the use cases of my system helped me fully conceptualize the functions of my system. This was made even more effective by forcing me to illustrate the possible flows of each use case. The use case diagram brought these use cases to life, illustrating the permissions, flows, and users involved in each action in an easily-digestible format. Lastly, the requirements validation technique assignment forced me to carefully evaluate my project's full set of requirements to understand how each one could be properly illustrated.



After these 5 assignments concluded, alongside the completion of the other SRS activities not labelled as belonging to the document, I was finally able to construct my SRS. I believe that this document demonstrates a true synthesis of all the information I've collected over the past several months.

## Validation Report

### Explain in general, how, why, and when you validate requirements in general.

Formally, requirements validation is a set of actions performed to confirm that a given requirement's purpose and functions comply with those desired by stakeholders. In other words, requirements validation is the act of checking that developers are working on the right problem, rather than checking if the problem was solved correctly. Although this step of the requirements engineering process technically comes last, successful software engineering projects begin validation checks much earlier on in the process. Furthermore, these validation checks should oftentimes be extended past the requirements engineering process, culminating in a final validation after system integration.

The truth is that requirements misunderstandings or errors can come at any stage in the requirements engineering process. In the elicitation phase, the BA or similar software representatives could ask the wrong question, misunderstand unintentionally-vague stakeholder wording, or even forget to clear up their misunderstandings of the stakeholders' requirements. In the analysis phase, mediators could do an ineffective job at resolving requirements conflicts, resulting in a situation where neither stakeholders' wants are fulfilled. In the specification phase, engineers giving the final pass-through of requirements before the SRS is created could miss an unaddressed requirement that was the result of a misunderstanding. These simple, easy mistakes can lead to disastrous consequences for the final product if left unaddressed, resulting in unfulfilled engineers and unhappy clients.

Some common methods of requirements validation are stakeholder reviews, prototyping, modelling and simulation, conceptual modelling, and formal modelling. However, it's crucial to consider that not just any requirements validation technique can be used on a given project. Each stakeholder has different interest, values, and intentions for the system. A security company, for example, might be interested in the under-the-hood data security techniques utilized, meaning that a prototype that focuses on user interface interactions may not be fitting. Likewise, it's unlikely that a low-ranking employee for the store you're developing the product for will want to see how the databases in the project interact through the exchange of data with the system.

### From the SEEK *Requirements validation* topics, describe: reviews and inspections, prototyping to validate requirements, and acceptance test design.

**Reviews and inspections** are a more classical method of requirements validation. Rather than making some validation vehicle such as a prototype or acceptance test, this activity simply involves getting a requirements review team, with at least one customer representative in the midst, to carefully review each requirement. The composition of this group is to be of special consideration, as the perspectives of the reviewers could affect how they interpret the requirements. During these reviews, it's also important to ensure that stakeholder requirements can be understood by those they were elicited from, preferably proven through sign-off.

**Prototyping** is a method of requirements validation wherein an audience is presented with an interactive user interface that loosely demonstrates the system's intended function. This method is particularly effective when unclear requirements are given, as the team can repeatedly iterate on this prototype if it doesn't fit the needs of the stakeholders and users. This method is low-cost and less time-consuming than going forth with a misunderstood premise. The end result is that the aforementioned imprecise and vague requirements gain a more concrete form, with a client's vision of the system coming to life.

**Acceptance test design** refers to the act of assigning each requirement in a set its own acceptance test. These acceptance tests are categorized as inspections, analyses, demonstrations, or tests depending on the unique circumstances surrounding the given requirement. However, the actual acceptance tests are **not** to be performed. The idea behind doing this is to confirm the testability of each requirement. If a requirement cannot be validated through one of the aforementioned methods, then it's less of a requirement and more of a wish.

### Describe how the requirements engineering process supports the validation of behavioral requirements.

The two primary methods for validating behavioral requirements lie in two of the three validation topics discussed in the last question: prototyping and acceptance test design.

Prototyping is the more obvious fit for behavior requirement validation. A visual demonstration of the team's understanding of the requirements is an efficient way to confirm their understanding of desired behavioral requirements. The client can interact with the prototype themselves, observing the result of their actions in real time as they attempt to test the behavioral requirements that they wanted. If the prototype lacks said functionality, the client can inform the team and the prototype can be revised until the behavioral requirement is well-formed and validated.

Though prototyping is the obvious built-in support, acceptance test design is also another fantastic way to validate behavioral requirements. In particular, tests and demonstrations are the most in line with this particular type of requirement. Tests are performed on specific elements of the system using special equipment, meaning that an ideal execution of the function is planned out. In other words, the requirement's behavior is being tested. Demonstrations test observable characteristics of the system, which can often map directly to behavioral requirements that describe an action that leaves a visual impact.

From your project, convert your notes from your requirements review into a detailed summary of the experience in the form of a memo that you could send to stakeholders and add to your project documentation. The structure of this document does not need to be formal, it is just something to help you remember what happened and cover yourself in the future. Identify and justify the composition of the group that conducted the review. Include information about any other validation you did, such as sharing a prototype.











Despite being in frequent contact with Steven Paul, the individual who the company had assigned to work with me for the project, I decided to go about the requirements review in as standard of a manner as possible. About a week ago, he and I had set up a time of 8:00 AM on April 16th, 2021 to conduct our formal requirements review via phone call. However, due to his busy schedule, he would not have time to look over the requirements set until after he got up at around 6:00 AM that day.

As I had been in frequent contact with my sponsor's representative, often consulting him to verify requirements align with the business' goals, the requirements review call was fairly straightforward. I had little explaining about individual requirements to do on my end. However, we still decided to conduct a full review. Starting from the top of the list, the process went as such:

1. I would read a requirement.
2. I would ask my Steven to explain how he interpreted it.
3. I would then explain how I meant for it to be interpreted.
4. We would then discuss whether the way the requirement was worded accurately conveyed its goal.
5. Then, any other questions or concerns regarding the requirement would be discussed.
6. The requirement would then be checked off, and we would continue to the next requirement in the list.

This process took about an hour long total. As I had already rewritten my requirements for clarity several times prior to our meeting, this process was conducted in an exceedingly smooth manner. After we were done with the requirements review, we discussed the prototype demo that I sent him. As it closely adhered to the feedback given to my mockup, I was told that it was effectively exactly what they had wanted. The call then concluded with him telling me that he would send an email response verifying his sign off on the requirements.

The set of requirements signed off on was as follows:

Key	Summary	T	Linked Issues	Labels
C3P-52	The migration process of the system shall occur on the night of October 9th, 2021, with employees of the minimum rank of Assistant Manager staying late to copy the current state of their store's inventory over.			nonfunctional
C3P-51	If a logged-in Employee logs out, the system shall clear the fields in the Inventory, Add Items, and Reports tabs before returning the Employee to the login screen.		C3P-17	interface, nonfunctional
C3P-50	If an Assistant Manager logs in, the system shall display the same UI elements outlined in C3P-49, the minimum and maximum editing function of the Inventory tab, the Add Items tab, and the Reports tab that allows for the viewing of archived reports.		C3P-12 , C3P-15 , C3P-18	interface, nonfunctional
C3P-49	If a Department Manager logs in, the system shall display the same UI elements outlined in C3P-48 and a stock quantity editing box in the same tab.		C3P-9	interface, nonfunctional
C3P-48	If a Sales Assistant logs in, the system shall only display the Inventory tab and the tab's searchable database.		C3P-10	interface, nonfunctional
C3P-47	The system shall display an error screen and prevent further action until re-connection if the system drops a connection with any of the three databases specified in C3P-43, C3P-44, and C3P-45.		C3P-16	database , functional
C3P-46	The system shall display an error screen and terminate the program if the system cannot establish a connection with the Employee Database, the Inventory Database, and the Reports Database on program startup.		C3P-16	database , functional
C3P-45	On program startup, the system shall connect to a Reports Database for the purposes of sending new generated reports and loading archived reports.		C3P-12 , C3P-14	database , functional
C3P-44	On program startup, the system shall connect to an Inventory Database for the purposes of loading inventory items, increasing and decreasing inventory items' stock count, and adding new inventory items.		C3P-9 , C3P-10 , C3P-11	database , functional
C3P-43	When an Employee logs in, the system shall connect to an Employee Database to verify that the user's employee ID and password exist in the database, then granting the permissions corresponding to the Employee's role.		C3P-17	database , functional

					I
C3P-34	Sunday at 11 PM, the system shall generate a weekly report that utilizes the prior seven daily reports to detail the week's profit, the week's sales, the items sold that week, and the orders filled that week.	☰	C3P-12 , C3P-14	functiona	I
C3P-33	System-generated reports shall detail daily profit, daily sales, items sold, and auto-filled orders.	☰	C3P-12 , C3P-13	nonfuncti	onal
C3P-32	The system's Add Items feature shall require a S.K.U., price, stock count, and in-store location when adding new items.	☰	C3P-15	nonfuncti	onal
C3P-31	If the user is at minimum the rank of Assistant Manager, the system shall allow the user to add new items to the inventory system while logged in.	☰	C3P-15	nonfuncti	onal
C3P-30	If the user is at minimum the rank of Assistant Manager, the system shall allow the user to change the values of product minimums and maximums while logged in.	☰	C3P-18	nonfuncti	onal
C3P-29	When an item's stock count reaches the item's minimum value, the system shall auto-fill an order form that will bring the stock count back up to the item's maximum.	☰		functiona	I
C3P-28	The system's Employee Login feature shall require an employee ID and employee-chosen password to grant device permissions to employees.	☰	C3P-36 , C3P-17	nonfuncti	onal
C3P-27	The Report Generation system for a given store shall generate daily reports with auto-filled orders one hour after the system's respective store closes.	☰	C3P-13	functiona	I
C3P-26	The Report Archival system shall display generated reports in order from newest to oldest.	☰		nonfuncti	onal
C3P-25	When an employee scans a non-new inventory item in, the system shall increase the item's current stock count by one within three seconds.	☰	C3P-16	functiona	I
C3P-24	When a cashier confirms a customer's purchase of an item, the system shall decrease the item's current stock count by an amount equal to the quantity purchased within three seconds.	☰	C3P-16	functiona	I

21 issues

## Clearly identify how you met specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors. (ABET)

The primary needs I met were in regards to safety and environmental factors.

In the case of safety, I'm actually referring to data safety! One of the initial requests by my stakeholders were that the system be secure. This is a very vague definition, especially from a computing standpoint. However, when you consider the business rules from my sponsor regarding the confidentiality of the company's sales, store, and employee data, it doesn't take much consideration to understand the gist of what they mean. Regardless, I took special care to inquire about these matters. This gave me confirmation on what I had already suspected - my sponsor wanted a login page that kept unauthorized personnel out, a database that would store the inventory information, and a database that would store the store's reports.

With these factors in mind, I had to carefully consider what kind of features the system would need in order to grant these requests. Though the databases, including a 3rd database for holding employee login information, were easy enough to consider, the first request was a slight roadblock. However, after some further consultation with my sponsor, I eventually decided that a dynamic permissions system would be the way to go. When an employee logged in, their login information would be sent to the Employee database to verify their credentials. Upon confirming the credentials of the user, the database would return the permission level of the user. The user would then be granted the permissions that corresponded to their role within the store. This minimized chances for data interception, as only 1 query that includes login information is made to the Employee database. Although it wasn't a perfect solution, it does have a large impact on the data safety of the system compared to an unprotected login. The measures I suggested decrease the probability of someone hacking into the Employee database and prevent unauthorized personnel from accessing highly-confidential store data that is not safe in their hands.

Economic and environmental factors were considered when it came to the report generation function of the system. As of now, Earth Origins still uses a pen and paper report generation methodology. Originally, the only reports the system was to generate were auto-filled orders for low-stock items. However, after hearing that the company used pen and paper still, my mind immediately thought about how much paper and ink they were wasting per year, harming the environment in the process. As such, I decided to suggest integrating the functionality of their report system into the project as well. My sponsor was extremely receptive to this idea, going as far as to add it as a main requirement for the project.