Personal Project

Golf Game Requirements Specification

Team Members: Gurdeep Singh, Jagjit Singh, Maruf Tonmoy

This template is based on the Volere Requirements Specification Template v.20

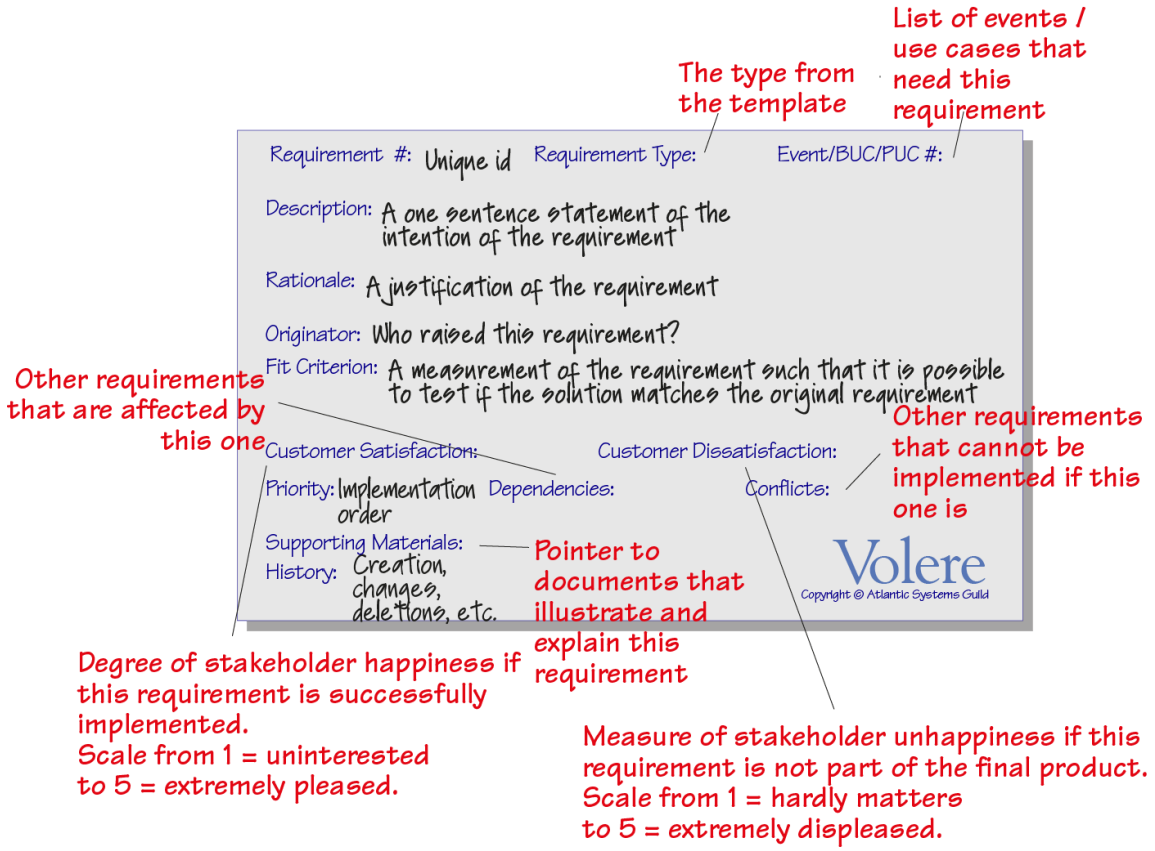
Atomic Requirements Shell 3
1. The Purpose of the Project 4
Goals of the Project 4
2. The Stakeholders 4
The Player The player has the final say on acceptance of the product, and thus must be satisfied with the product as delivered. 4
User Priority 5
Development Team (or Other Stakeholders) 6
3. Constraints 7
Solution Constraints 7
3g. Budget Constraints 8
4. Naming Conventions and Terminology 8
4a. Glossary of All Terms, Including Acronyms, Used by Stakeholders Involved in the Project 8
5. Assumptions 9
8. The Scope of the Product 10
8a. Use Case Diagram 10
8c. Use Cases 11
9. Functional Requirements 12
9a. Functional Requirements 12
Non-functional Requirements 13
10. Look and Feel Requirements 13
11. Risks 13

Atomic Requirements Shell

The requirements shell is a guide to writing each atomic requirement. The components of the shell (also called a "snow card") are identified below. An atomic requirement is made up of this collection of attributes.

An example snow card is shown below, however for the project you need only focus on the following:

1. *Requirement Id.*
2. *Requirement Type. (this will be expressed by the section that the requirement appears in)*
3. *Use Case Id*
4. *Description.*
5. *Rationale*
6. *Fit Criterion*
7. *Priority*


List of events / use cases that need this requirement
The type from the template
Requirement #: Unique id
Requirement Type:
Event/BUC/PUC #:
Description: A one sentence statement of the intention of the requirement
Rationale: A justification of the requirement
Originator: Who raised this requirement?
Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement
Other requirements that are affected by this one
Other requirements that cannot be implemented if this one is
Customer Satisfaction:
Customer Dissatisfaction:
Priority: Implementation order
Dependencies:
Conflicts:
Supporting Materials:
History: Creation, changes, deletions, etc.
Pointer to documents that illustrate and explain this requirement
Volere
Copyright © Atlantic Systems Guild
Degree of stakeholder happiness if this requirement is successfully implemented.
Scale from 1 = uninterested to 5 = extremely pleased.
Measure of stakeholder unhappiness if this requirement is not part of the final product.
Scale from 1 = hardly matters to 5 = extremely displeased.

1. The Purpose of the Project

Goals of the Project

Rules of Golf Card Game-

- When a player replaces a card from the deck, the replaced card has to be discarded into a discard pile either face up or down.
- When the draw deck runs out, shuffle the discard pile to make a new deck.

Objective of the Game-

- It is to score the lowest points. Each card has a certain point value if it is unmatched. Matched cards in the same column result in no points.

Game play-

- On a first player's first turn, the player flips over two cards from their hand.
- Two cards are only flipped in the first round.
- After the first round, the player then chooses to draw the top card from the discard pile.
- Either player can choose to replace the drawn card with one of the cards in their hand and discard the replaced card or discard the drawn card.
- Play continues this way until a player has all their cards facing up.
- Then the game pauses, everyone shows their card and count points.

Describe what you want the game to do. Do not be too wordy in this section—a brief explanation of the project's goals is usually more valuable than a long, rambling treatise. A short, sharp goal will be clearer and improves the chances of reaching a consensus for the goal.

A sentence or a graph or diagram that quantifies how you will measure whether or not the goals have been achieved. Any reasonable goal must be measurable. This is necessary if you are ever to test whether you have succeeded with the project. For example, if the goal of the project is

> **We want to give immediate and complete response to customers who order our goods online.**

you have to ask what advantage meeting that goal brings to the organization. If immediate response will result in more satisfied customers, then the measurement must quantify that satisfaction. For example, you could measure the increase in repeat business (on the basis that a happy customer comes back for more), the increase in customer approval ratings from surveys, the increase in revenue from returning customers, and so on.

2. The Stakeholders

This section describes the stakeholders—the people who have an interest in the product. It is worth your while to spend enough time to accurately determine and describe these people. This will help when you create the user stories for the project.

The Player

The player has the final say on acceptance of the product, and thus must be satisfied with the product as delivered.

Give your player(s) a name. Also, you may want to create a persona for the player(s) such as name, age, job, family, hobbies, where they live, favourite food, favourite music, likes, dislikes, where they go on holiday, attitude to technology, attitude to money, or any other characteristic that could influence the way that the persona thinks of the game. It can help if you include a photograph or drawing of the imagined person. You may want to consider the follow to guide your discussion:

1. *Subject matter experience: Summarizes the users' knowledge of the subject matter/business. Rate as novice, journeyman, or master.*
2. *Technological experience: Describes the users' experience with relevant technology. Rate as novice, journeyman, or master.*
3. *Other user characteristics: Describe any characteristics of the users that have an effect on the requirements and eventual design of the product. For example:*
4. *Physical abilities/disabilities*

5. *Intellectual abilities/disabilities*
6. *Attitude toward job*
7. *Attitude toward technology*
8. *Physical location*
9. *Education*
10. *Linguistic skills*
11. *Age group*
12. *Gender*
13. *Ethnic group/s*

Consider having more than one (but stop at three so as not get too complicated) to account for players with different backgrounds or technical levels. This can also help if you are planning to have any AIs that the player is going to compete against, as you can think of the different strategies they may use.

User Priority

Attach a priority to each category of user. This identifies the importance and precedence of the user. Prioritize the users as follows:

1. *Key users: They are critical to the continued success of the product. Give greater importance to requirements generated by this category of user.*
2. *Secondary users: They will use the product, but their opinion of it has no effect on its long-term success. Where there is a conflict between secondary users' requirements and those of key users, the key users take precedence.*
3. *Unimportant users: This category of user is given the lowest priority. It includes infrequent, unauthorized, and unskilled users, as well as people who misuse the product.*

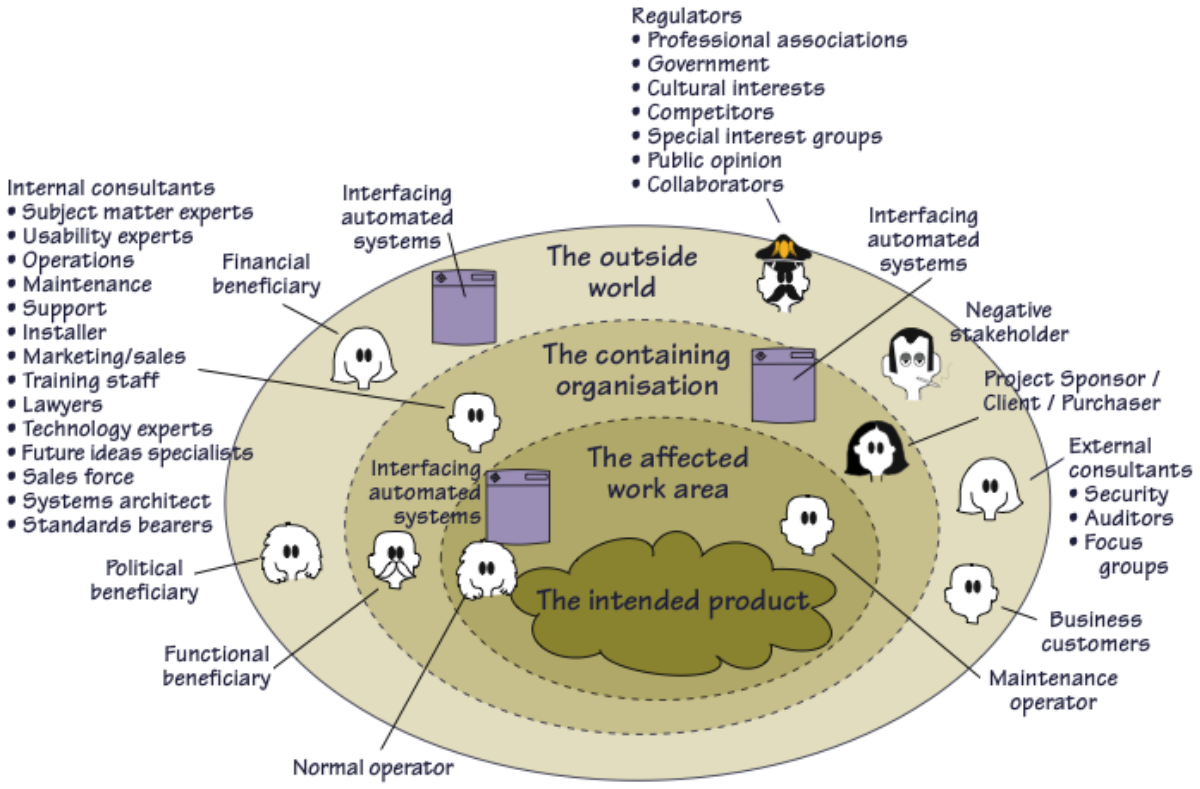
The percentage of the type of user is intended to assess the amount of consideration given to each category of user.

Development Team (or Other Stakeholders)

Consider the needs of the people who are going to develop the software. These can include (but are not necessarily limited to) the following development roles:

1. *Usability experts*
2. *Designers and developers*
3. *Testers*
4. *Software engineers*
5. *Technology experts*
6. *System Maintainers*

You may find the following diagram helpful in thinking about the other stakeholders.



3. Constraints

This section describes constraints on the eventual design of the product. Think of a constraint as: no matter how you solve the problem this must be satisfied.

Constraints are global—they are factors that apply to the entire product. The product must be built within the stated constraints. Often you know about the constraints, or they are mandated before the project gets under way. They are probably determined by management and are worth considering carefully—they restrict what you can do and so shape the product. Constraints, like other types of requirements have a description, rationale, and fit criterion, and generally are written in the same format as functional and non-functional requirements.

Solution Constraints

This specifies constraints on the way that the problem must be solved. Describe the mandated technology or solution. To make the constraint specific include any appropriate version numbers. You should also explain the reason for using the technology.

These constraints guide the development of the final product. Your client, customer, or user may have design preferences, or only certain solutions may be acceptable. If these constraints are not met, your solution is not acceptable.

We want to define the boundaries within which we can solve the problem. Be careful, because anyone who has experience with or exposure to a piece of technology tends to see requirements in terms of that technology. This tendency leads people to impose solution constraints for the wrong reason, making it very easy for false constraints to creep into a specification. The solution constraints should only be those that are absolutely non-negotiable. In other words, however you solve this problem, you must use this particular technology. Any other solution would be unacceptable.

Present the constraints as a subsection (or the like) with the following items:

7. *Description*
8. *Rationale*
9. *Fit Criterion (i.e. how will you know the constraint is met?)*

Examples

Constraints are written using the same form as other atomic requirements (refer to the atomic requirements snow card/shell for the attributes). It is important for each constraint to have a rationale and a fit criterion, as they help to expose false constraints (solutions masquerading as constraints). Also, you will usually find that a constraint affects the entire product rather than one or more product use cases.

The product shall use the current two-way radio system to communicate with the drivers in their trucks.

Rationale: The client will not pay for a new radio system, nor are any other means of communication available to the drivers.

Fit criterion: All signals generated by the product shall be audible and understandable by all drivers via their current two-way radio system.

The product shall operate using Windows XP.

Rationale: The client uses XP and does not wish to change to a later version.

Fit criterion: The product shall be approved as XP compliant by the MS testing group.

3g. Budget Constraints

This section shows the budget for the project, expressed in available resources (i.e. project personnel and time). The requirements must not exceed the budget. This limitation may constrain the number of requirements that can be included in the product.

The intention is to restrict the wildest ambitions and to prevent the team from gathering requirements for a AAA 3D first-person shooter when the budget is only for a text-based command-line game.

4. Naming Conventions and Terminology

All projects have their own unique vocabulary usually containing a variety of acronyms and abbreviations. Failure to understand this project-specific nomenclature correctly inevitably leads to misunderstandings, hours of lost time, miscommunication between team members, and ultimately poor-quality specifications.

4a. Glossary of All Terms, Including Acronyms, Used by Stakeholders Involved in the Project

Face value: A value which is printed on the card. It could range from the number 2 to 10 or it could be one of A, K, J and Q.

Points: These are number values which help in deciding who wins the game. Every card is assigned a value as follows:

Face value	Points
2-10	equal to face value
A	1
K	0
J, Q & 10	10

Shuffle: It is to mix up the playing cards in a draw pile so as to randomise the order in which they appear.

Deal: To distribute the cards to all the players before the beginning of a round.

Round: When one whole circle of each player turns ends.

Set: After revealing all the cards of each player ,the game decides a winner through summing all the points of each respective player.

This period and phase is called a Round.

Turn / move : When a player draws a card from a draw pile or shows one card.

Draw:

Deck (full deck):

Discard Pile: This is the deck where we discard cards after the play.
Draw Pile: We draw cards from this pile to play the game.
Total Score: Sum of all the points at the end of thegame to check who is the winner.

Players: The participants who are playing the game including the computer(AI).

Player hand: A set of cards in custody of the playerat the given moment.

Reveal card: It is a function to show one of the invisible cards from the player's hand.

Winner: the player who has a lower total score at the end of the game.

5. Assumptions

A list of the assumptions that the developers are making. These assumptions might be about the intended operational environment, but can be about anything that has an effect on the product. As part of managing expectations, assumptions also contain statements about what the product will ***not*** do.

We often make unconscious assumptions. It is necessary to talk to the members of the project team to discover any unconscious assumptions that they have made. It is important to state these assumptions up front. You might also consider the probability of whether the assumption is correct and, where relevant, a list of alternatives if something that is assumed does not happen.

Examples

1. *Assumptions about what your developers expect to be ready in time for them to use—for example, other parts of your products, the completion of other projects, software tools, or software components.*
2. *Assumptions about the technological environment in which the product will operate. These assumptions should highlight areas of expected compatibility.*
3. *The software components that will be available to the developers.*
4. *Dependencies on computer systems or people external to this project.*
5. *The requirements that will specifically not be carried out by the product.*

7. Functional Requirements

This is a specification for each atomic functional requirement. As for all types of atomic requirements (functional, non-functional, constraint), use the requirements shell as a guide for which attributes should be specified. A full explanation of the atomic requirement and its attributes is included in this template's introductory material.

The use cases will help you to determine the functional requirements.

Include the cost of requirements, which is the effort that you have to spend building them into a product. Once the requirements specification is complete, you can use an estimating method (e.g. Planning Poker) to assess the cost, expressing the result in story points.

8. *Requirement Id.*
9. *Requirement Type. (this will be expressed by the section that the requirement appears in)*
10. *Use Case Id*
11. *Description.*
12. *Rationale*
13. *Fit Criterion*

14. *Priority*

Example

75. The product shall record all the roads that have been treated

Priority or Implementation Order	5
Use Cases	7, 9
Rationale:	To be able to schedule untreated roads and highlight potential dangers
Fit Criterion	The recorded treated road shall agree with the driver's road treatment logbook and shall be up to date within 30 minutes of the completion of the road's treatment.
Story Points	3

Priority or Implementation Order	5
Use Cases	7, 9
Rationale:	To be able to schedule untreated roads and highlight potential dangers
Fit Criterion	The recorded treated road shall agree with the driver's road treatment logbook and shall be up

	to date within 30 minutes of the completion of the road's treatment.
Story Points	3

Non-functional Requirements

The following section(s) describe the non-functional requirements. The form of these requirements is the same as for the functional requirements as described in section 9 above. Your team can add additional non-functional sections as you see a need.

8. Look and Feel Requirements

The section contains requirements relating to the spirit of the product. This section captures the requirements for the appearance.

Include images of high-fidelity, horizontal prototypes of the expected interface, with accompanying descriptions.

9. Risks

All projects involve risk—namely, the risk that something will go wrong. Risk is not necessarily a bad thing, as no progress is made without taking some risk. Risk is only a bad thing if the risks are ignored and they become problems. Risk management entails assessing which risks are most likely to apply to the project, deciding a course of action if they become problems, and monitoring projects to give early warnings of risks becoming problems.

This section of the specification contains a list of the most likely and the most serious risks for your project. For each risk, include the probability of it becoming a problem and any contingency plans.

Risks will undoubtedly change during the lifetime of a project. The better you understand the requirements the better you can identify which risks are most serious for your project. The project manager determines how to manage the risks but the requirements specialists and developers provide input on new risks and which risks are turning into problems.

Use your knowledge of the requirements as input to discover which risks are most relevant to your project. Also, consider the following categories of project risks:

1. *Requirements/Design/Estimation*
    a. *The team planned a project that is too large (i.e. "eyes bigger than stomach").*
    b. *The team underestimated how long parts of the project would take.*
    c. *Major changes to design are needed during implementation.*
2. *People*
    a. *Addition or loss of team member (i.e. someone dropped the course, a new person joins the team)*
    b. *Unproductive team member(s)*
    c. *Team member(s) lacking expected technical background (e.g. don't know C++)*
    d. *Major life events (e.g. long-term illness, death of family member, birth of a child)*
3. *Learning & Tools*
    a. *Inexperience with new tools*
    b. *Learning curve for tools steeper than expected*
    c. *Tools don't work together in an integrated way*