Developing a Platform for Strategic Cybersecurity Awareness Games

# Motivation

The purpose of this research project is to develop a software application, with which strategic cybersecurity awareness games (SCAG) can be created and conducted. SCAGs are targeted towards the management and senior management of an organization. Typical participants in SCAGs will have a high-impact role in their organization. It is therefore crucial that they are aware of cybersecurity risks and familiar with relevant mitigation measures.

The SCAG itself will follow a collaborative, scenario-based approach, wherein a group of participants are confronted with a description of realistic situations and must discuss the best course of action in such situations. Aside from gaining a better understanding of cybersecurity risks, the communication among the participants will also foster better understanding of different perspectives on cybersecurity. This will potentially improve cooperation with other cybersecurity stakeholders in their respective organizations.

The SCAP will be facilitated by a trainer, who provides guidance when group discussions become stuck or stray off track and who can make ad hoc changes to the game setting to adapt to the needs of the participant group.

# Objectives

## Project-Level Objectives

The overarching purpose of SCAGs is to foster a culture of cybersecurity awareness within organizations, because only this will truly harden the “weakest link” of cybersecurity - the human factor - in the long term. Such a culture is best developed, if managers and senior managers lead by example and exhibit cybersecurity awareness themselves. By exposing these managers and senior managers to a realistic cybersecurity scenario, the SCAG seeks to equip them with the necessary skills and attitude to adequately react to such situations, thereby setting an example for their reports (see Figure 1).

By discussing the these risks and potential mitigation strategies as a group, the participants will establish an informal standard for communicating cyber-related topics, which will further contribute to the development of a culture of cybersecurity awareness. This exchange of perspectives will also allow participants to better communicate with cybersecurity stakeholders, thereby allowing them to serve as “multipliers” of cybersecurity awareness within their organization.

This research project aims to develop a software to support the conception and execution of SCAGs. Because this software is still at the proof-of-concept stage, emphasis will be placed on developing a functional and usable prototype, while other aspects of software quality may be added later.

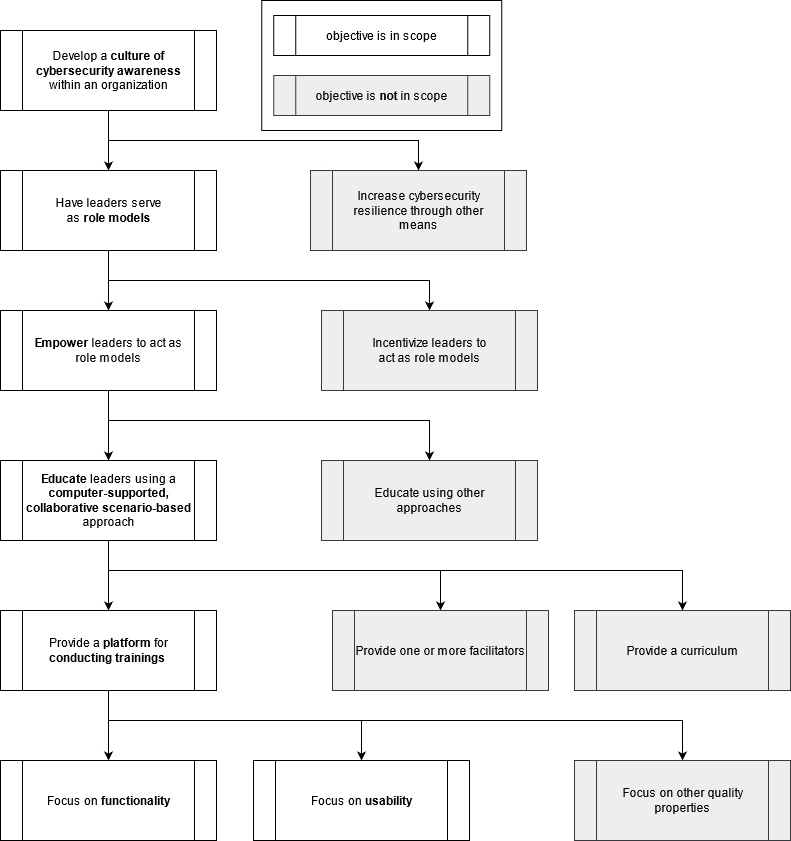


Figure 1: The overall goals of this project (source: own work).

## Functional Objectives

The functionality of the software must satisfy three primary objectives. These are, in order of priority:

1. **Conduct a game.** This is by far the most important use case of the software and implies that a trainer can facilitate a game which is being played by participants.
2. **Develop a scenario.** To be able to play through a game scenario, the scenario must first be designed. While this use case does not by itself satisfy the primary goal of the software, it is nonetheless essential for productive use.
3. **Receive feedback.** Learning happens through feedback. A platform dedicated to learning should provide feedback to participants and trainers alike. While this use case is not strictly essential to fulfilling the project goal, it will support the continuous improvement of the games, thus increasing the quality of the software in the long term.

Figure 2provides an overview of how the functional goals of the software can be decomposed.

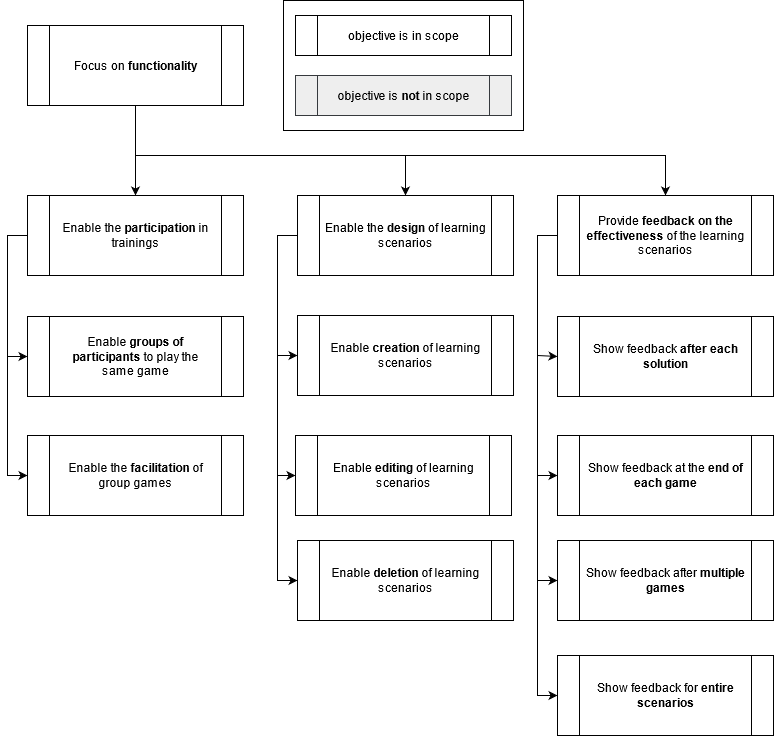


Figure 2: Hierarchy of functional objectives of the SCAP platform (source: own work).

# Definitions

To ensure the use of an ubiquitous language in formulating subsequent requirements, the central concepts of the research project are hereafter defined.

## **Scenario**

A scenario is an ordered sequence of *stories*, which are suitable for one or more *target groups*. The meta-information of a scenario includes a *title*, *description* and possible *image*.

A scenario may also have *variables*, which may be visible or not visible to a player.

A scenario can be created, modified and deleted by a *scenario designer*.

## Game

A game is an instance of a scenario that has been or can be played by one or more players. A game with only one player is a single player game, whereas a game with a group of players is a group game.

A group game is sometimes called a group discussion. A group game can be facilitated by a trainer and observed by an observer. A game has a history, which covers all solutions that have been submitted.

A game can have one of three states:

* *Open*: A trainer has chosen a scenario to play and created a lobby for this game. Participants can now join the lobby.
* *In Progress*: Participants are actively playing the game. No further participants may join now.
* *Closed*: The game has either been finished (all injects played) or aborted. No further participants may join now. Trainers cannot change any of the game variables.

## Story

Let story S be a directed graph. Let I be a set of all nodes of S and let T be the set of all edges of S.   
  
Note that the nodes of a story are hereafter referred to as injects and the edges of a story are referred to as transitions.

## Inject

An inject provides the user with information or a task, thus advancing the scenario. It can be understood as a possible state that a game may reach.

An inject may reference and be referenced by any number of transitions. An inject that only refers to one transition is an informational inject, while an inject that refers to multiple transitions is an input inject.

An inject that is not referenced by a transition is called an entry point, whereas an inject that references no further transitions is called an exit point.

An inject that neither refers to nor is referred by transitions is considered illegal. For the sake of simplicity, this case will not be considered further.

An inject can be solved by a learner, if they select a transition. The exit point cannot be solved.  
An inject may have an inject reaction, which provides immediate feedback to the learner, after they have solved the inject.

## Transition

A transition describes a labeled, directed path from one inject to another inject. If multiple transitions point connect one inject to another, they can be called choices.

## Target Group

A target group is a set of statements which describe the type of learner that is expected to profit most from this scenario.

These statements refer to the industry of the organization of the learner, the prior knowledge, the position of the learner within their organization.

# Requirements

## Methodical Approach

The above objectives must now be decomposed into concrete requirements. System verbs describe the priority of the requirement, whereas behavior verbs describe the behavior which the system must exhibit to satisfy this requirement.

The system verbs in the requirements description follow the [MoSCoW](https://en.wikipedia.org/wiki/MoSCoW_method) method:

* **MUST** denotes a requirement, without which the system cannot be used to perform core use cases.
* **SHOULD** denotes a requirement which is not needed to perform core use cases, but without which the system is strongly limited.
* **COULD** denotes a requirement which is not necessary for the system to function, but which might bring considerable additional value.
* **WILL NOT** denotes a requirement which is understood to be relevant for the system, but not currently in scope of implementation.

The behavior verbs in the requirements description describe which actions the system supports:

* **ENFORCE**: This action must be performed by the user. The system validates the action.
* **PROHIBIT**: This action must not be performed by the user.
* **ALLOW**: The user can choose to perform this action or not. The system validates the input.
* **SHOW**: The user must do nothing, there is no input.

The system verbs can be used to prioritize the requirements:

* **Essential:** This requirement must be implemented
* **Medium:** This requirement will be implemented if it fits into the implementation schedule and does not put any essential requirements at risk.
* **Low:** This requirement will be implemented if it fits into the implementation schedule and does not put any essential or medium requirements at risk.

## User Types

The requirements are structured by user. If a requirement is relevant for multiple users, it has been placed with the user who is affected most strongly.

From the previously discussed functional goals, the following user roles can be deduced:

1. **Participant.** The players of the strategic awareness game.
2. **Trainer.** The trainer who will facilitate the strategic awareness game.
3. **Scenario Designer.** The person who develops and improves the learning scenarios.

Moreover, separation of concerns dictates that another user have the ability to manage the assignment of roles to user accounts. We can therefore deduce the need for an **administrator role**.

Finally, the MITRE playbook[[1]](#footnote-1) suggests having a white team for cyber exercises, which does not actually facilitate the exercise, but monitors and provides live feedback if necessary. This may require another type of role, the **observer**.

## Participant Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority** |
| 1 | The platform MUST ALLOW participants to collaboratively[[2]](#footnote-2) play a game as part of a group. | Essential |
| 2 | The platform COULD ALLOW participants to play a game alone. | Medium |
| 3 | The platform SHOULD ALLOW participants to describe themselves using keywords.  Keywords can be used to describe:   * The industry of the participant, * The prior experience of the participant, * The position of the participant within their organization. | Medium |
| 4 | IF the participants have used keywords to describe themselves,  AND these keywords indicate that the participants are not familiar with some concepts,  THEN the platform SHOULD SHOW hints with additional information to explain these concepts to the participants. | Medium |
| 5 | The platform MUST ALLOW participants to view the variables of the game that are set to “visible”. | Essential |
| 6 | The platform MUST ALLOW players to solve injects.  An inject solution may be:   * Selecting one choice of multiple choices. * Inserting a textual value. * Advancing an informative inject. | Essential |
| 7 | The platform MUST SHOW the same sequence of injects to all participants of a game. IF participants choose different choices when solving one inject,  THEN the system MUST determine one choice which all participants follow. | Essential |
| 8 | The platform MUST ALLOW participants to access a game without having to log in. | Essential |
| 9 | IF a participant was previously disconnected from a game,  THEN the platform MUST ALLOW players to rejoin. | Essential |
| 10 | The platform SHOULD PROHIBIT participants from joining closed games. | Essential |
| 11 | The platform SHOULD SHOW participants the personal history of a previously played game. The personal history of a game consists of:   * All of the solutions for injects which this participant has submitted in the course of this game. * Timestamps for the aforementioned solutions. * Timestamps denoting the start and end time of the game. | Medium |

## Trainer Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority** |
| 12 | The platform MUST ALLOW trainers to open games. | Essential |
| 13 | The platform SHOULD ALLOW trainers to see how many participants have joined an open game. | Medium |
| 14 | The platform MUST ALLOW trainers to begin games. | Essential |
| 15 | The platform MUST ALLOW trainers to see the relevant information of a game in progress:   * Which inject participants are currently working on. * Which variables exist in the game. * What value each of the variables has. * Any other injects and stories. | Essential |
| 16 | The platform SHOULD ALLOW trainers to change variable values for games in progress. | Essential |
| 17 | The platform MUST ALLOW trainers to abort games.  Aborted games are *closed.* | Essential |
| 18 | The platform SHOULD SHOW trainers the general history of a game which they have previously closed. The general history of a game consists of:   * A count of the number of players of this game. * All of the solutions for injects which have been submitted in the course of this game. * Timestamps for the aforementioned solutions. * Timestamps denoting the start and end time of the game. | Low |

## Observer Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority** |
| 19 | The platform SHOULD ALLOW observers to see how many participants have joined an open game. | Medium |
| 20 | The platform SHOULD ALLOW observers to see the relevant information of the game in progress:   * Which inject participants are currently working on. * Which variables exist in the game. * What value each of the variables has. * Any other injects and stories. | Medium |
| 21 | IF an observer has observed a game in the past,  THEN the platform SHOULD ALLOW the observer to view the general history of this games any time in the future. | Low |

## Scenario Designer Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority** |
| 22 | The platform MUST ALLOW the creation of scenarios. | Essential |
| 23 | The platform MUST ENFORCE that a scenario has all required meta-information:   * Title, * Description, * Target Group. | Essential |
| 24 | The platform MUST ENFORCE that a scenario has one or more *stories*. | Essential |
| 25 | The platform MUST ENFORCE that the stories of a scenario are in sequential order. | Essential |
| 26 | The platform MUST ENFORCE that a story has the following properties:   * Title. | Essential |
| 27 | The platform MUST ENFORCE that a story has one or more *injects*. | Essential |
| 28 | The platform MUST ENFORCE that a story has exactly one inject as an *entry point*. | Essential |
| 29 | The platform MUST ENFORCE that an inject has the following properties:   * Title, * Textual Description. | Essential |
| 30 | The platform SHOULD ALLOW an inject to have an Image. | Medium |
| 31 | The platform COULD ALLOW an inject to have informational snippets. [[3]](#footnote-3) | Medium |
| 32 | The platform MUST ALLOW an inject to have zero or more choices.  Injects with zero choices are *Exit Points*, they conclude a story or a game.  Injects with only choice are *informative injects*.  Injects with more than one choice are *choice injects*. | Essential |
| 33 | The platform MUST ensure that all injects within a story are can be reached from at least one other inject. | Essential |
| 34 | The platform MUST ENSURE that injects to not only have circular references. | Essential |
| 35 | The platform MUST ALLOW a transition to change the variable values of the game. | Essential |
| 36 | The platform MUST ALLOW a transition to have different outcomes, IF a predefined condition is met. | Medium |
| 37 | IF an inject has more than one transition, the system MUST ENFORCE that each transition has a title. | Essential |
| 38 | The platform MUST ENFORCE that a transition condition references an alternative inject. | Essential |
| 39 | The platform MUST ALLOW the editing of scenarios. | Essential |
| 40 | IF a scenario has been edited,  THEN the platform MUST allow backwards compatibility of usage statistics.[[4]](#footnote-4) | Medium |
| 41 | The platform MUST ALLOW the deletion of scenarios.  This will also delete all games and statistics that are associated with this scenario. | Essential |
| 42 | The platform MIGHT NOT allow the selective deletion of previous versions of a scenario. | Low |

## Administrator Requirements

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **Priority** |
|  | The platform SHOULD ALLOW the creation of user accounts. | Low |
|  | The platform SHOULD ALLOW a user account to have multiple roles. | Low |
|  | The platform SHOULD ALLOW an admin to add to and remove roles from user accounts. | Low |
|  | The platform SHOULD ALLOW each user to delete their own account. | Low |
|  | The platform SHOULD ENFORCE each account to have adequate authentication measures, such as, but not limited to:   * Password strength, * 2-factor-authentication, * Biometrical authentication. | Low |

# Agile Approach

## Methodical Approach

Originally proposed by Beck[[5]](#footnote-5) in 1999 and refined by various authors, *user stories* are natural-language approach to expressing requirements from the point of view of a specific user of the system. User stories are kept in a *backlog* in order of priority.

Patton[[6]](#footnote-6) has critized that backlogs, like all requirements documents, tend to be flat lists and has instead proposed the *user story map* as a way of structuring requirements. A user story map is a hierarchical collection of requirements that are derived from *user activities.* User activities describe objectives that a user may have and can be further broken down into *user stories.*

Patton further argues that each system has a number of requirements which are so essential that not the system cannot function if they are missing – the *backbone*. Because missing just one of the requirements from the backbone renders the system unusable, these requirements cannot be further prioritized.

The backbone can be decomposed into smaller user stories however, some of which may be more critical than others. A small number of user stories is usually enough to satisfy the minimum of the backbone requirements. Because these stories in principle allow the system to function, Patton calls them the *walking skeleton* (see Figure 3)*.* This walking skeleton corresponds with the concept of a *minimum viable product*, which, while not ready for productive use, already covers the core functionality of the intended system.

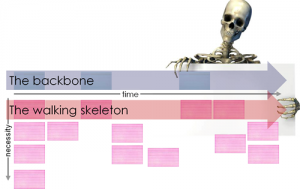


Figure 3: User Story Maps feature essential requirements in the backbone. These requirements are then decomposed into user stories. The user stories most essential for delivering the backbone are also called the “walking skeleton” (source: Jeff Patton[[7]](#footnote-7)).

Subsequent user stories will then add to the functionality in increasing levels of granularity. This creates a hierarchy of user stories, where the stories most critical for delivering the backbone must be implemented first (see Figure 4). Blue boxes represent the backbone, green boxes represent use cases that make up the walking skeleton and yellow boxes are additional use cases.

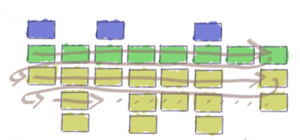


Figure 4: A user story map visualizes which user stories are absolutely critical to delivering the product, even if they are spread across multiple features. (source: Jeff Patton[[8]](#footnote-8)).

## User Story Map

Figure 5 depicts how the requirements outlined in chapter 4 might be structured in a user story map. The blue boxes describe user activities, sometimes called epics or user journeys. The yellow boxes are the backbone of the system and consist of comparatively large user stories that are necessary for using the system. The white boxes are the smaller-scale user stories that collectively form the backbone. The walking skeleton is called *MVP (Minimum Viable Product)* in this map – these are all of the user stories that are strictly required for essential use of the system. Cagan claims that because the walking skeleton is intended as more of a tool for validation and testing rather than productive use, the term *minimum viable prototype* may be more adequate.[[9]](#footnote-9)

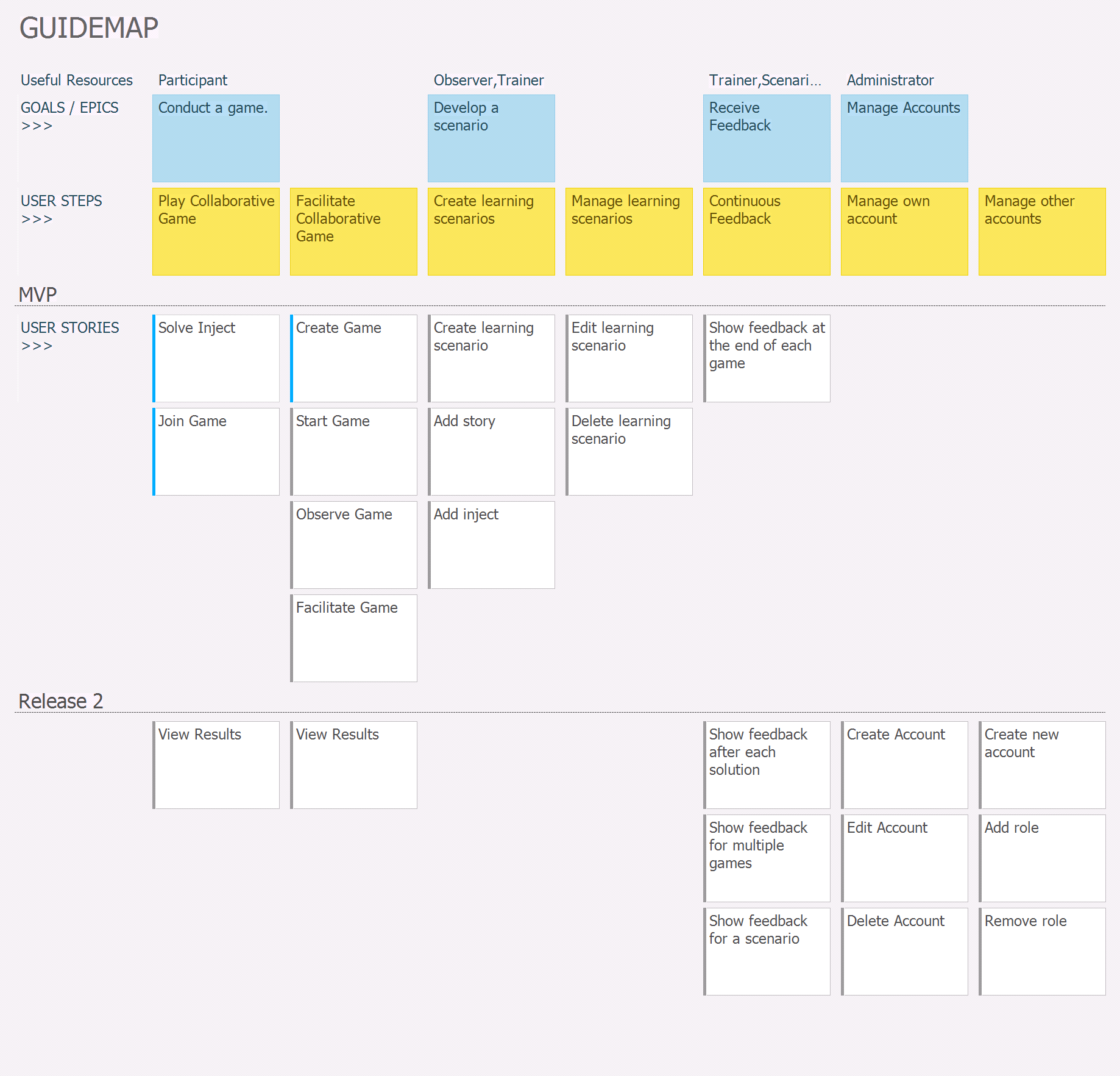


Figure 5: The user story map of the planned system (source: own work).

As is apparent from Figure 5, the user journeys “Play Collaborative Game”, “Facilitate Collaborative Game”, “Create Learning Scenarios”, “Manage Learning Scenarios” and “Continuous Feedback” must be completed to at least some extent for the release of the MVP.

The status of each of the user stories is provided by the colored line at the left. The blue line for “Solve Inject”, “Join Game” and “Create Game” indicate that these stories are currently in progress, whereas grey lines represent user stories that are still open.

## Detailed Descriptions of the User Stories

A user story map provides an excellent overview over the existing requirements and their priority with respect to the entire project. Nonetheless, more information is required to enable actual implementation and testing of these user stories. Therefore, the user stories shown in Figure 5 will be described in more detail in this chapter. The user stories follow the Connextra Template[[10]](#footnote-10).

Figure 6 describes a user story from the perspective of a game participant. It has a descriptive title (“Solve Inject”) and a short description of the requirement (“to solve an inject”). This particular story additionally comprises a definition of done (DoD), which is a checklist of tests which the system must pass for this story to be considered as “done”. This story further contains the tag “splittable”, as an indicator that it may need to be split into even smaller stories for the actual implementation.

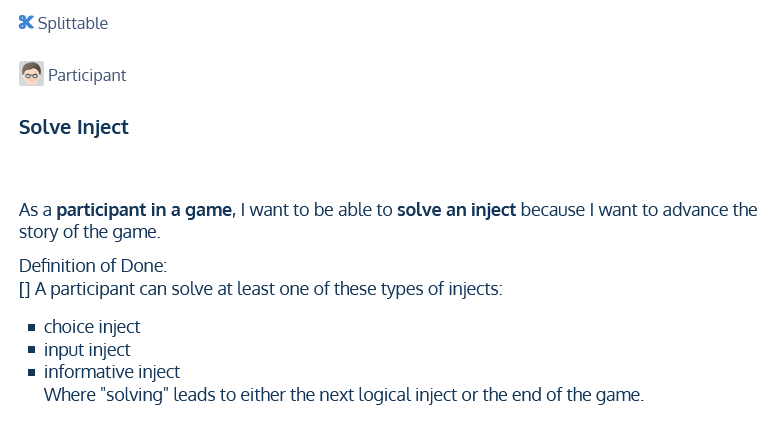
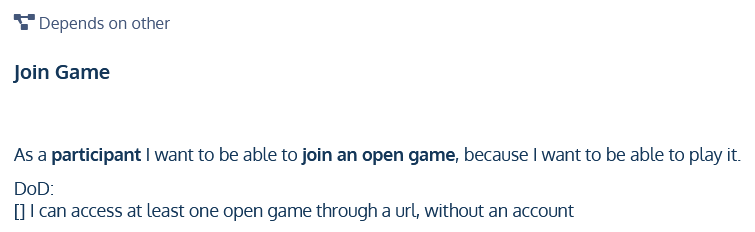


Figure 6: The user story "Solve Inject".

In addition to the obligatory title, story description and DoD, the story “Join Game” (see Figure 7) also has the tag “Depends on other”, because it has a logical dependency on the story “Create Game”.

Figure 7: The user story "Join Game".

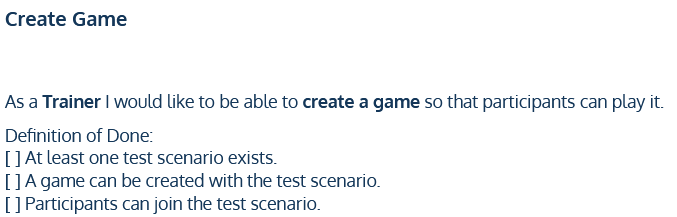


Figure 8: The user story "Create game".

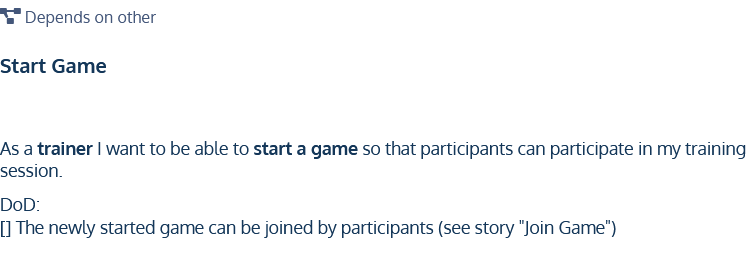


Figure 9: The user story "Start Game".

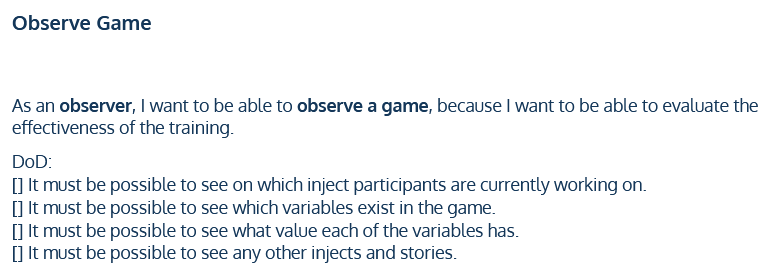


Figure 10: The user story "Observe Game".

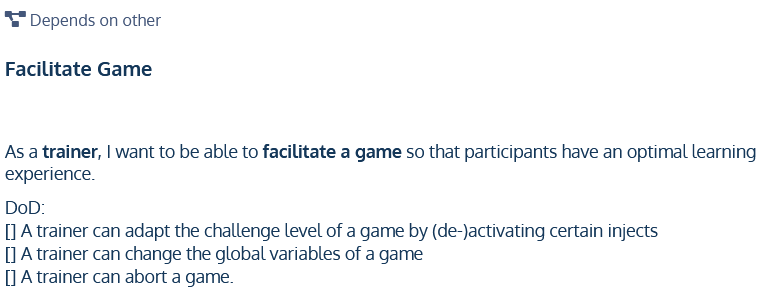


Figure 11: The user story "Facilitate Game".

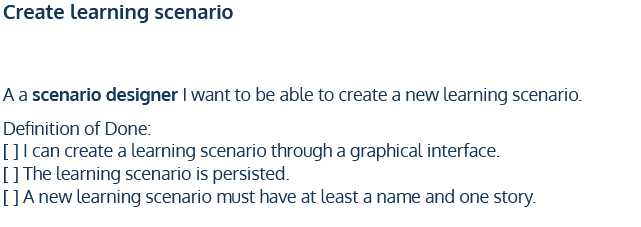


Figure 12: The user story "Create learning scenario".

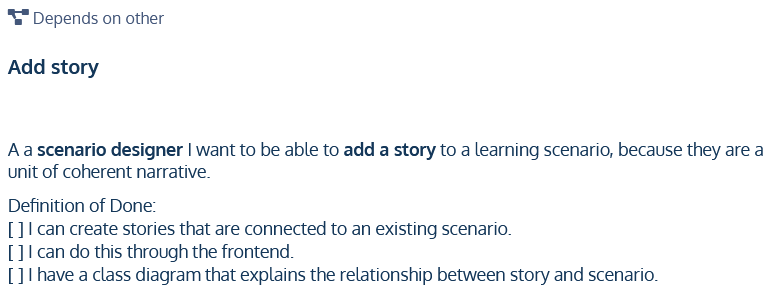


Figure 13: The user story "Add story".

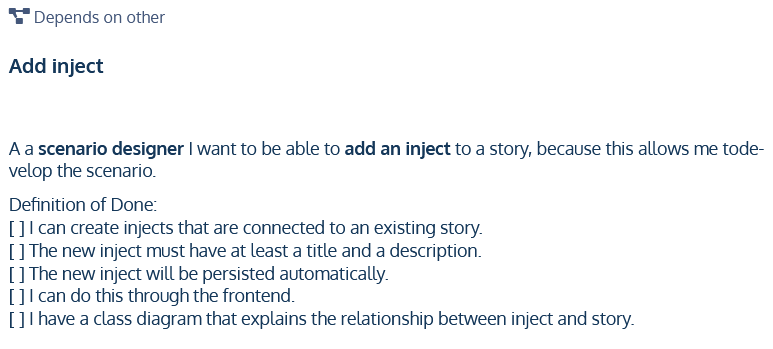


Figure 14: The user story "Add inject".

1. Kick, Jason (2014): MITRE Playbook. [↑](#footnote-ref-1)
2. Collaborative game play: All participants have the same goal. In practice, they play through exactly the same scenario. [↑](#footnote-ref-2)
3. An informational snippet is shown to a user, IF they meet certain criteria (i.e. pre-defined keywords) and can provide additional information or an additional challenge.

   An example for an additional snippet would be to show a HR-professional an explanation of why a certain attack vector can be easily exploited, but not show ing this information to an IT specialist, who is expected to know this already. [↑](#footnote-ref-3)
4. When a scenario is modified, this may impact future statistics. Therefore there must be a way to make transparent which change in statistics was brought about by which modification. [↑](#footnote-ref-4)
5. Beck, Kent (1999): Extreme Programming Explained: Embrace Change. Addison-Wesley. [↑](#footnote-ref-5)
6. Patton, Jeff (2005): It’s all in how you slice it. [↑](#footnote-ref-6)
7. Patton, Jeff (2008): The New User Story Backlog is a Map. Retrieved from <https://www.jpattonassociates.com/the-new-backlog/>. Last access on 2021-06-02. [↑](#footnote-ref-7)
8. Patton, Jeff (2008): The New User Story Backlog is a Map. Retrieved from <https://www.jpattonassociates.com/the-new-backlog/>. Last access on 2021-06-02. [↑](#footnote-ref-8)
9. Cagan, Marty (2017): Inspired. John Wiley & Sons, 2. Edition. [↑](#footnote-ref-9)
10. Agile Alliance (ed.) (2021): User Story Template. Retrieved from <https://www.agilealliance.org/glossary/user-story-template/>. Last access on 2021-06-04. [↑](#footnote-ref-10)