

# Game Manual Tech-Debts 4.0

## 1. Introduction

### 1.1 Objective of the Game

During the game Tech-Debts 4.0 players build software systems. The goal is to acquire the most points for your own system. Accumulating technical debt can be a strategic choice, potentially speeding up or slowing down your progress. This game seeks to raise awareness about technical debt and its impact. The yellow text boxes throughout the manual highlight real-world parallels to the concepts discussed.

The game is played 2 vs 2 to encourage discussions among team members.

The best handler of technical debts wins!

#### Technical debt

„In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible.”

### 1.2 Game Concept






You are project managers in a company. It's your job to plan and carry out software projects. Developers will implement the system and must solve its problems. Above you in the hierarchy is the IT management team that makes crucial decisions regarding deadlines and budgets. They have a lot of IT background knowledge but are usually not involved with the practical implementation. They want to be able to continuously develop the system, hence they value quality. They also make decisions that can affect all levels of development and are in contact with customers who want feature-rich software as soon as possible.

The company has just landed two new clients. Human and financial resources are limited, leaving capacity for only one new project at present. However, the management has come up with a solution. As project managers you are to implement both projects as prototypes within a limited time. Afterward, only the system that has generated more users and is of higher quality will be implemented. A good indication of quality is the extent of technical debts in a system. So, the goal is to outperform the opposing team and increase both the user count and the quality of your system.

## 2. Setup

### 2.1 Game Components

You need the following items to play the game in print version:


- 2 dice (D6) 
- One game piece per team.  
- At least 30 game pieces in red  to represent Technical Debts (TD).
- At least 9 game pieces in green  to represent task progress.
- Tech-Debt 4.0 in printed form
- Classic game pieces, tiles, or similar items from a game collection in different colors or other small identical objects can be used as game pieces.
- Timer or stopwatch (recommended), if you play the game for a set time limit (e.g., the game ends after 60 minutes).



## 2.2 Before the First Game

- All game elements must be prepared initially. Cut out all cards (🏠, ⭐, !, ? and Workstation/Remove TD). The page with overviews is cut in half, so each team has an overview. The 2 systems and the user display are not cut out.

## 2.3 Setup

- Prepare the systems by placing your systems, consisting of two pages, in front of you. Place the overview along with the workstation next to your system.
- In the middle of the table, shuffle the cards face down into separate piles.
- Place the user count next to the piles. Place the teams' game pieces on 0.

For digital play: Instead of a user display, there is a counter  for each team in the digital version. Double-clicking on the counter opens it. Add or subtract points by clicking on the plus and minus buttons.

- Also, place the game pieces for TD  and task progress  in the middle. The setup should look like Figure 1.

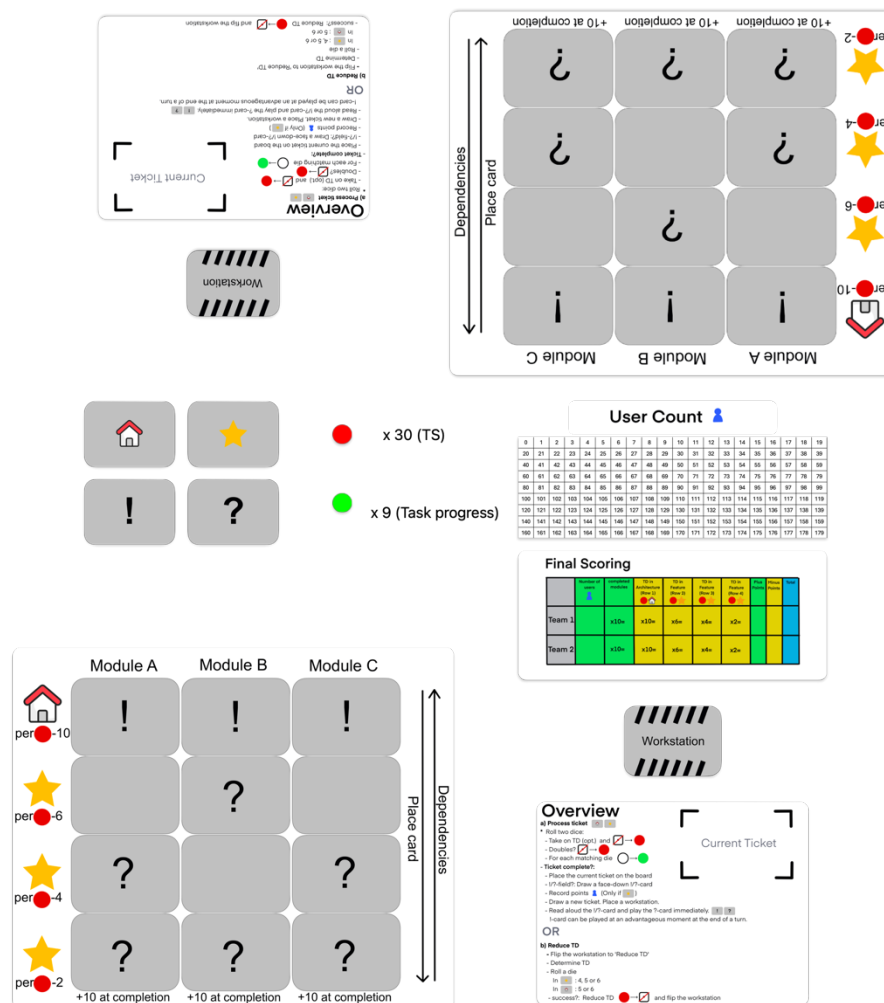




Figure 1: Game setup

Before the start of the first round, each team draws a  from the face-down stack and places it on the current ticket spot in the Overview. Then, place your workstation on any space in the first row where the is  depicted.



### 3 Game Elements

#### 3.1 Tickets

Tickets are the individual components that are assembled into a large system over the course of the game. They are placed on a field in the system after processing. Each team works on exactly one ticket at any given time, which is referred to as the current ticket. There are two types of tickets:

##### **Tickets in Software Development**

In software development the collection of all pending tasks is often divided into what are known as tickets. This could include, for example, a bug fix, a new feature, a reorganization of components (architecture) or the reduction of technical debt. Thus, they are not limited to the areas of feature and architecture. In reality the implementation of an entire feature or architecture would also be divided into many smaller tickets. Tickets therefore correspond more to smaller work units.

- **Architecture** : They improve the infrastructure and flexibility of the system, thus allowing it to expand.
- **Feature** : Features are desired functions for the system. These tickets, upon complete processing, generate as many new users as the number shown behind them.

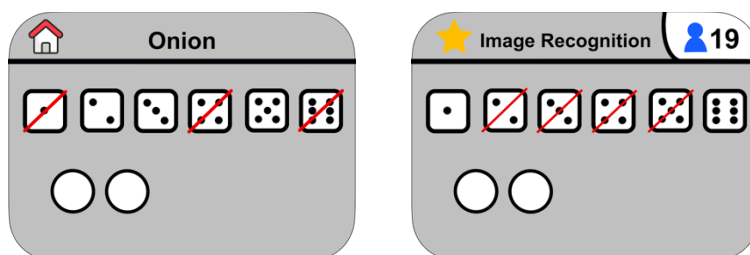











Figure 2: Architecture Ticket and Feature Ticket

**Tasks** : Tasks on the tickets are marked by empty circular fields. The number of empty fields determines how many tasks need to be processed to implement the ticket. The number of tasks can vary from 2 to 5.

**Free and Locked Dice**  : All dice faces are depicted on the tickets. They can be free  or locked . There can be 1 to 5 locked dice faces on a ticket.

## 3.2 System

More and more tickets are added to your system over the course of the game, causing your system to grow and generate users. As the game progresses, tickets are placed on various fields in the system. The symbol on the field determines what happens after a ticket is placed.

- **Empty fields:** nothing happens here.
- **Action fields** : An action card  must be drawn.
- **Event fields** : An event card  must be drawn.

## 3.3 Technical Debts (TD) ●

You can consciously take on TD ● on your current ticket. This is advantageous whenever you want to quickly add a ticket to your system. However, TD can also be received unwittingly during the game. Regardless of how TD have entered the system, they carry the risk of hindering you later in the game.

### (Un)conscious Technical Debt

In software development technical debts can be consciously incurred. This creates short-term shortcuts for development. This happens, for example, when a provisional solution is implemented under time pressure. More often technical debts enter the system unconsciously for various reasons and are only discovered later. For instance, a lack of professional competence can be such a reason. Generally, for future development, technical debts carry the potential risk of incurring additional costs.

## 3.4 Workstation / Reduce TD

Before the processing of a ticket begins, it must be decided where the ticket should be placed upon completion. The workstation is placed at this location. The workstation can be flipped to the Reduce TD side to deal with the removal of Technical Debt from the system.

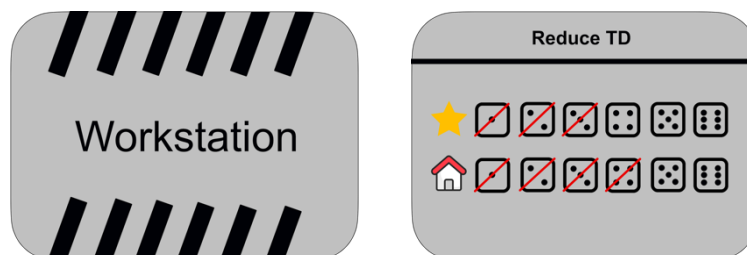




Figure 3: Workstation and Reduce TD

## 3.5 Event Cards and Action Cards

The cards all have a name, contain a description, and a bold event or an action.

- **Event cards** : represent unexpected events during development. They are marked by a letter that signifies the type of event: The orange **C** stands for the **Cause** of TD ●, they are discovered randomly and lead directly to the acquisition of TD ● in your own system. The red **C** stands for **Consequence** and can have direct negative effects depending on the amount of TD ● in your system. Therefore, the more TD ● you have in the

system, the higher the risk of consequences! The event of this card must be executed immediately!

- **Action cards** : contain measures that can be helpful for the development of the system. Here, the action may be played at a beneficial moment at the end of a turn.

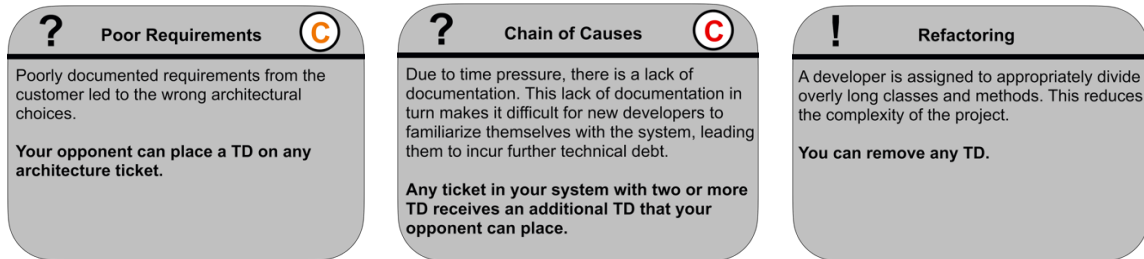


Figure 4: Event Cards (C and C) and Action Card

### 3.6 Users

During the game, you can gain **users** for your system. These will be displayed in the user count.



(added by the counter  in the digital version)

### 3.7 Overview



The overview next to your system provides an overview of the turn sequence. Here the individual steps are listed in order. There is also space to place the current ticket.

## 4 Placing Tickets and Dependencies Between Tickets

### 4.1 Placing Tickets

Tickets are placed into the system after all tasks have been processed. The system is divided into columns or modules and grows from **top to bottom**. Architecture tickets  can only be placed in the first row of the system, while Feature tickets  can only go into the subsequent rows beneath. Thus, each module first gets an Architecture and then Features can be added. This allows the modules to grow independently of each other.

The location where a ticket is to be placed is decided **before processing a ticket** by placing the workstation on the corresponding field in the system.

Figure 5 illustrates the placement options, showing possible locations for the workstation for tickets in an advanced system. In this example, an Architecture  could be placed at the blue-marked spot and a Feature  at one of the green-marked locations.

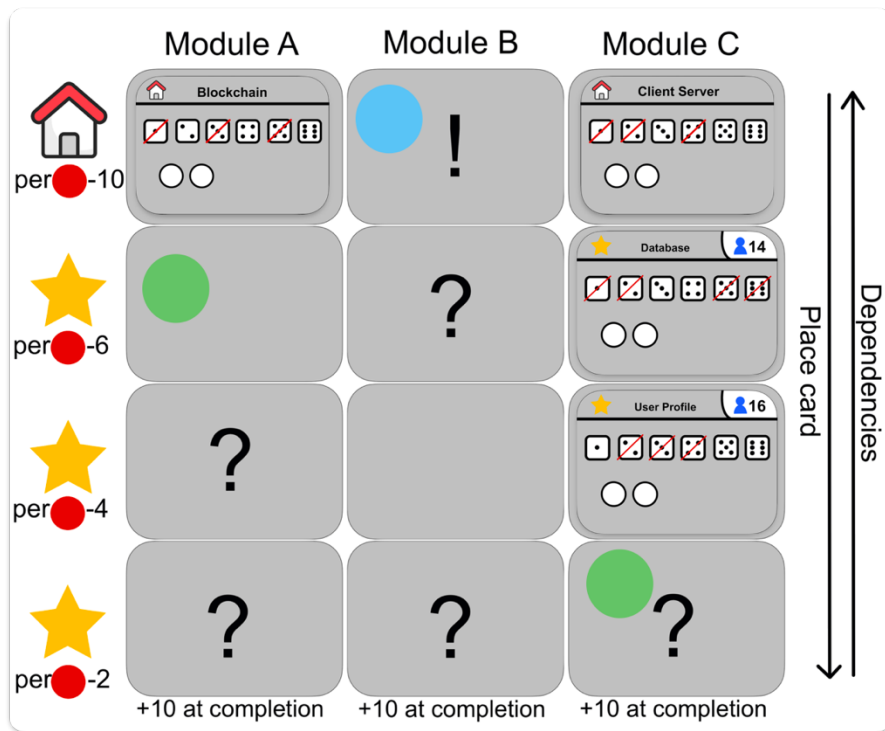


Figure 5: Placement Options for Architecture and Feature Ticket

## 4.2 Dependencies

A ticket is always dependent on all the tickets lying above it from the same module. Figures 6-8 show different examples of dependencies. The field with the workstation, and thus the current ticket, depends on the tickets above.

In Figure 6, the current ticket is to be placed in the second row of Module A, and therefore depends on the Architecture from the first row of Module A.

### Dependencies and Technical Debt

If there are technical debts in a system, they can potentially constrain further development. If a ticket A relies on another ticket B that is laden with technical debts, it might happen that the development for ticket A is limited or even impossible. This problem can affect further development tickets and cause additional technical debts. In reality dependencies are not necessarily limited to tickets but can also affect entire modules, for example.

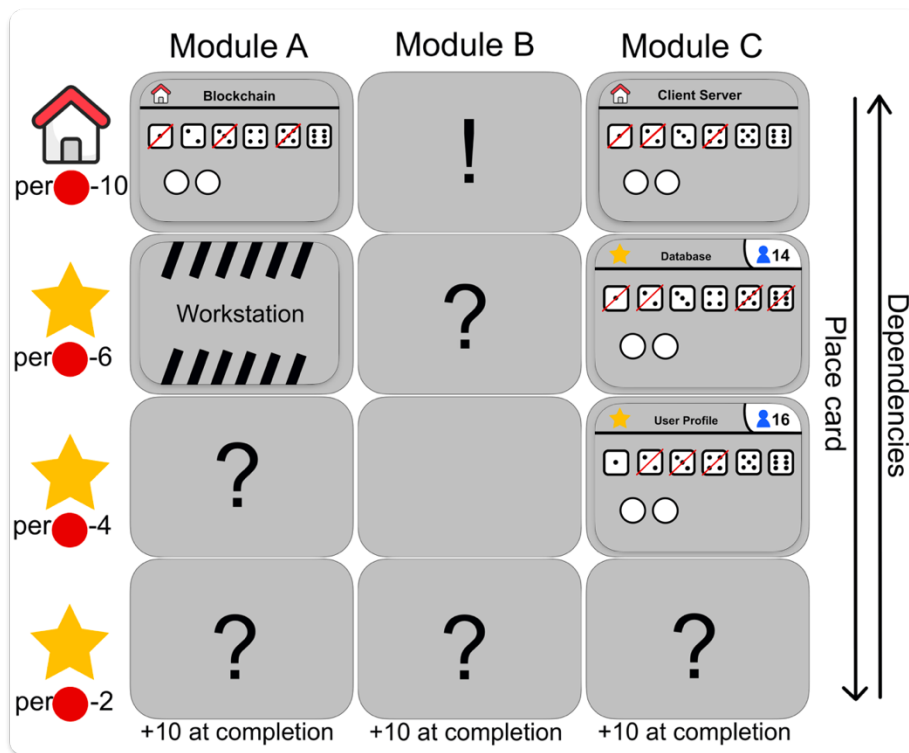


Figure 6: Dependency example 1

In Figure 7, the current ticket is to be built in the first row of Module B. This ticket is not dependent on any other ticket since no tickets can lie above it. This applies to all tickets built in the first row, thus to all Architecture tickets.

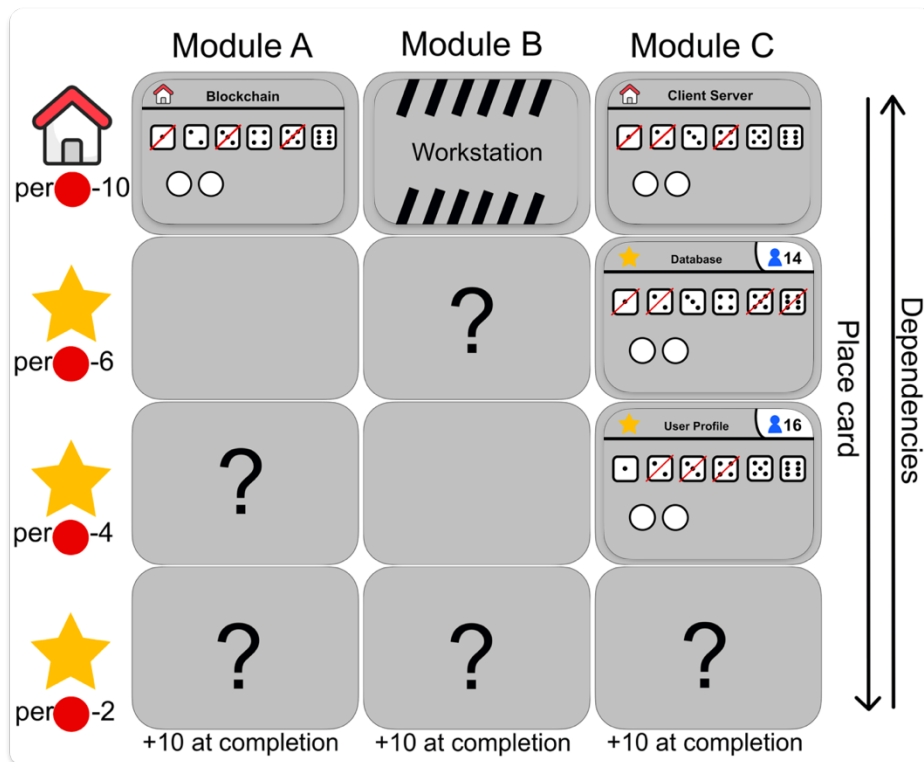


Figure 7: Dependency example 2

In Figure 8, the current ticket is to be placed in the last row of Module C. This ticket is dependent on the three tickets lying above it.

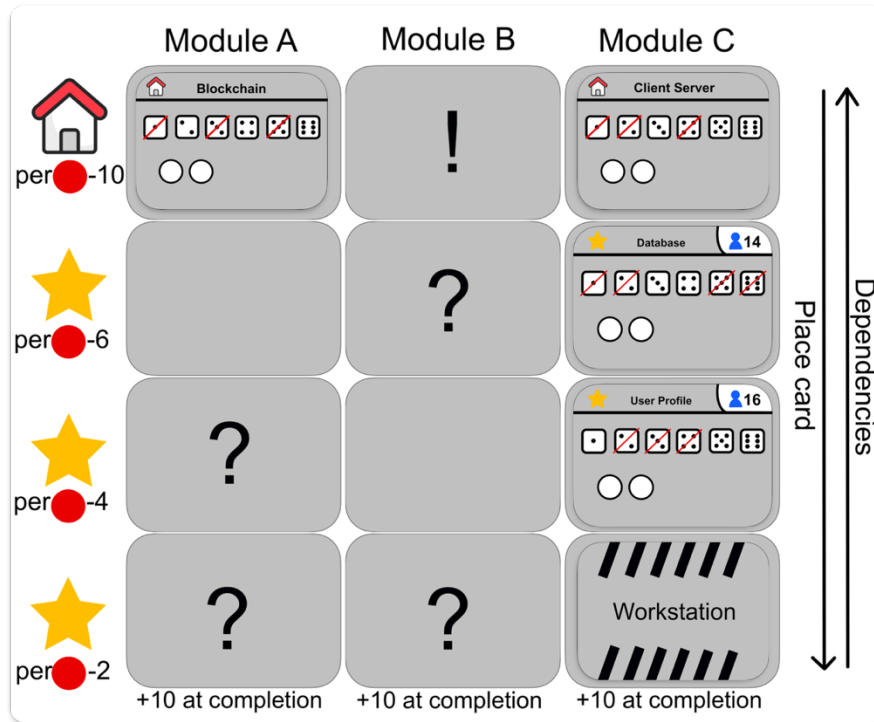



Figure 8: Dependency example 3

If the current ticket depends on another ticket, the dice numbers on which the other ticket has TD ● are locked for the processing of the current ticket. This lock cannot be bypassed by the (un)voluntary acquisition of TD ● on the corresponding dice number of the current ticket. The locked dice numbers without TD ● from already built tickets do not affect the current ticket.

Figure 9 shows an example of dependencies with TD ● in the system and the resulting free □ and locked  dice for processing the tasks. Here, the current ticket Live Chat is to be implemented into the system on the workstation field. The ticket depends on the features above: User Profile and Database, as well as the Client Server. Other tickets have no influence on the current ticket.

The locked and free dice are determined as follows:

- 1 is locked due to the TD ● on the Client Server ticket.
- 2 is locked due to the TD ● on the Database ticket.
- 3 is locked both due to the TD ● on the User Profile ticket and the locked dice number on the current ticket.
- 4 is locked both due to the TD ● on the Client Server ticket and the locked dice number on the current ticket.
- 5 is locked due to the locked dice number on the current ticket. But it can be freed by consciously taking on TD ● on the current ticket. (Because 5 is not marked with TD ● in the tickets above).
- 6 is free because of the free die on the current ticket.



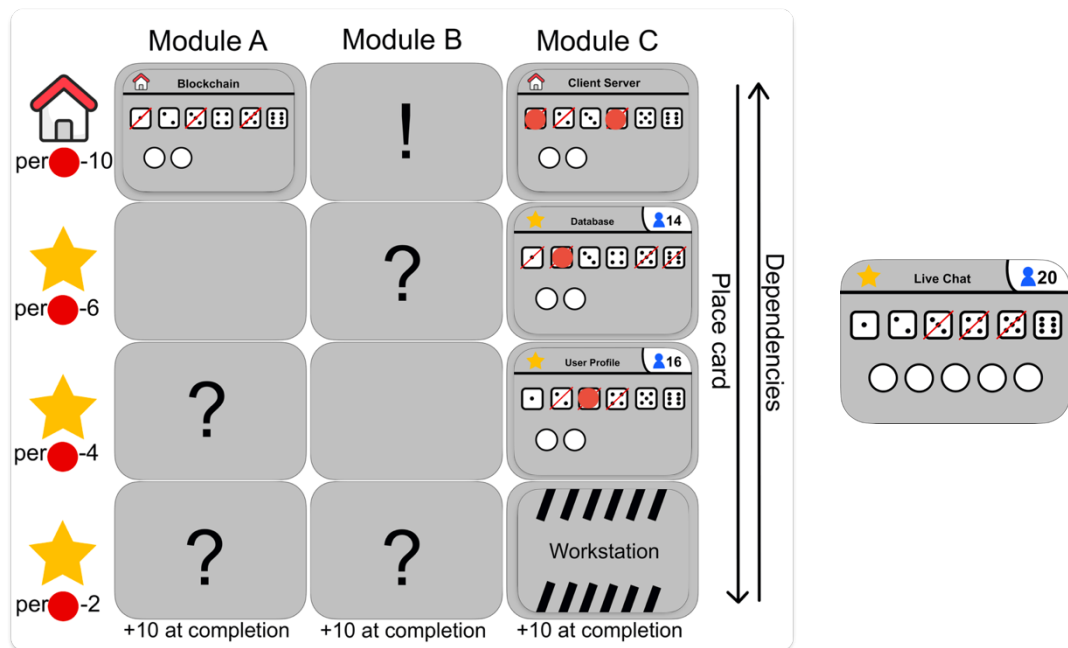


Figure 9: Dependency with TD

## 5 Game Progress

To build the software system, several rounds are played. In each round, each team takes a turn. In a turn, two optional steps are possible. You must decide whether to (continue to) work on your current ticket or to reduce TD in your system.

### a) Process the Ticket

To process your current ticket, the following steps are necessary:

#### Roll:


Now roll two dice. If you roll doubles, you automatically place a TD ● on the number rolled, provided there isn't one already there. It doesn't matter if the die on the current ticket is locked or free. If this number is not blocked by a TD ● from a dependent ticket, the rolled doubles can be used directly for processing the current ticket. If the number is blocked by a dependency, you place a TD ● on the current ticket on the number of the double rolled dice. If the current ticket already has a TD ● on that number, the rolled doubles can still be used directly for processing the current ticket.

#### Inevitable technical debt

Avoiding the accumulation of technical debt is not entirely possible because it can also be incurred unconsciously. However, even if unintentional, this can lead to shortcuts in development.

#### Take on TD (optional):


No free dice rolled? You can intentionally place any number of TD ● on your current ticket on locked dice. This is done by placing a TD ● on the desired die of a ticket. You can then use this as a free die for processing tasks until the ticket is completed, as long as it is not

blocked by a TD  from a dependent ticket. This makes implementing the current ticket significantly faster. As long as the taken TD is on your current ticket, the number remains unlocked.

#### **Taking on technical debt**











The deliberate taking on of technical debt allows for quicker procession of tickets. However, for future tickets built upon this ticket it could mean that their processing may be slower.

**But be careful, later other tickets that depend on this ticket cannot use the dice occupied with TD for processing the current ticket!**


Then check how many free dice were rolled. For each free die hit, place a game piece  on an empty task field. If the task fields are already full, the excess die is forfeited.

#### **Ticket complete?**


If all tasks are completed, the current ticket is integrated into the system.

- Remove the game pieces  for processing the tasks. **The TD  remains on the ticket!** Then exchange the workstation with the current ticket.
- If the ticket lands on an **event field**  or **action field** , take the corresponding **event card**  or **action card**  and set it aside face down, **but do not look at it yet!**
- If you have implemented a feature , record the generated user count. 
- Then choose a new current ticket from the middle (feature or architecture) and place the workstation at a valid position in your system.
- If you have drawn an event card or action card, you may now look at it. Read the card out loud.
  - The event card  must be executed immediately.
  - The action card  can be played at a beneficial moment at the end of a turn. You can play any number of previously drawn action cards at the end of your turn. Then set the card aside.

#### **b) Reduce TD:**


In this step, you can reduce the TD accumulated in your system. First, turn your workstation to the Reduce TD side. Then decide on a specific TD  to be reduced. Afterwards, roll a die.



- To reduce any TD  on a Feature , you must roll a 4, 5, or 6.
- To reduce any TD  on an Architecture , you must roll a 5 or 6.



If you roll the required number, remove the corresponding TD  from your ticket. If you do not succeed, you can decide in the next turn whether to execute step a) or b) again. Flip the Reduce TD card back to the workstation side.

For a quick reference, you can find the individual steps of a turn on the Overview.

## 6 End of the Game

The game ends either as soon as all 12 spaces for tickets have been filled in a system or after a predetermined time. In either case, the round is completed. Afterwards, the following points are accounted for in addition to the already collected number of users  for each system:

- For each completed module, +10 points
- For each TD  on Architecture, -10 points
- For each TD  on Feature in rows 2 / 3 / 4, -6 / -4 / -2 points respectively

The TD  and the number of users  from the current ticket are not included in the final scoring! It is best to record the individual items in the scoring table, which is located under the user display. You can take the total number of users directly from the user display. Figure 10 shows an example for calculating the final scoring.

The team with the most points has built the better prototype and wins the game.

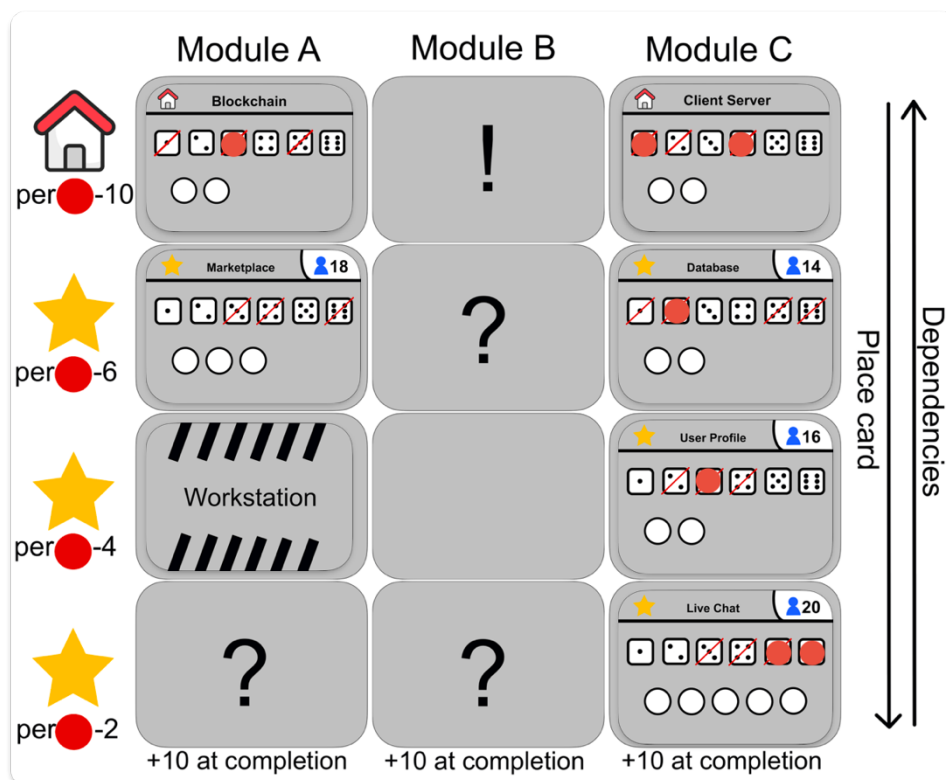



Figure 10: Example for the Final Calculation

### Example Calculation:

In this example, the team has generated 68 users. Additionally, they have completed Module C, which is worth 10 points. From these two numbers, the team has a total of 78 plus points.

Now, minus points are calculated based on the number of TDs . The team has a total of 3 TDs on Architecture tickets. These are multiplied by 10, resulting in 30 minus points for Architecture.

The TDs on Feature tickets are then broken down by the row in which the respective ticket with a TD is located. In the second row, each TD ● incurs 6 minus points. In this example, there is one TD ● on the Database ticket in Module C, thus 6 minus points. In the third row the Feature User Profile has one TD ●, resulting in 4 minus points. Finally, there are 2 minus points per TD in the fourth row. Here lies only the Feature Live Chat in Module C, which has two TD and thus generates 4 minus points.

This results in a total of 44 minus points. For the final result the minus points are deducted from the plus points. Therefore, in this example, the team has achieved a total of 34 points.