Player Modeling for Role-Playing Games

Improving Bethesda’s Radiant AI

Henrique Moisés de Carvalho Louro Fernandes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Pedro Alexandre Simões dos Santos

Examination Committee
Chairperson: Prof. Rui Filipe Fernandes Prada
Advisor: Prof. Pedro Alexandre Simões dos Santos
Members of the Committee: Prof. João Miguel De Sousa de Assis Dias

May 2018
Acknowledgments

I would like to express my deep gratitude to Professor Pedro Alexandre Simões dos Santos, my thesis supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. The door to his office was always open whenever I ran into a trouble spot or had a question about my research or writing. As my supervisor, he has helped me more than I could ever give him credit for here.

I am also extremely grateful for those who participated in the training data set tests that led to the implemented classifier and those who were part of the mod validation testing group. The time and feedback they gave to this project was invaluable. I would also show my appreciation for all those who filled the demographic and mod validation surveys, thus providing the needed information for the thesis results.

A special "Thank you" to all the thousands of people who downloaded, tested, provided valuable feedback and participated in active discussions related with the "Your Own Skyrim" mod - the result of this thesis - and the overall concept, increasing the popularity of the YOS mod - which was higher than I could ever expect. This also applies to the good folks at the Skyrim mods subreddit that motivated me with their supportive words, excitement in the topic and active discussions.

I would also like to thank my family members, friends and girlfriend, that provided the emotional support to carry on in more exhausting times and did not let me lose my focus, heard me rambling about the project and topics of this thesis and provided me with some writing advice.

Finally, due to the hundreds of hours I spent adventuring in the virtual world of Skyrim and using the Creation Kit, I would also like to thank Bethesda Softworks for creating my favorite video game of all time - TESV:Skyrim - and providing the modding community the tools to easily and freely create modifications to the base game, thus creating one of the bigger modding communities in the video games scene.
Abstract

Most modern Role-playing video games (RPGs) include an extensive virtual world and interesting narratives that the player can immerse himself in. It is easy to lose track of time when exploring the virtual worlds and helping the various characters that inhabit it. A captivating and interactive narrative experience is essential in a successful RPG. However, when looking at the options that the player can take to change the virtual environment or the general direction of the narrative, those often feel limited or inconsequential. Furthermore, some content can feel generic and created to please all players, as it opposed to a more personalized experience where the player actions help shape the story.

The AAA video game industry have often disregarded this fact and developed very basic Artificial Intelligent and Player Modeling systems that often do not meet the required expectations, despite the fact that these have proven - in academic research - to enhance player interest and expected enjoyment. We do believe that a custom experience, different across player with different traits, motivations and preferences can add more replay value, amusement and better storytelling to any RPG.

In this work, we present a Player Modeling architecture that uses a Machine Learning instance that analyses player actions and interactions with the virtual world and associates them with a player profile, in order to create a tailored experience that should provide better enjoyment and immersion for the player. This system was implemented in the popular RPG title The Elder Scrolls V: Skyrim and released as a game modification (mod), which was met with extremely positive feedback by the player community.

Keywords: Player Modeling, Interactive Storytelling, Role-playing Games, Decision Tree, Player Models, Machine Learning
Resumo

A maioria dos videogames de role-playing (RPGs) mais modernos inclui um extenso mundo virtual e histórias interessantes onde o jogador se pode imergir. É fácil perder-se a noção do tempo quando se está a explorar o mundo virtual e a ajudar os variados personagens que o habitam. Uma experiência narrativa interativa e cativante é essencial para um RPG de sucesso. Contudo, olhando para as opções que o jogador tem para mudar o ambiente virtual ou mudar a direcção da narrativa, estas podem ser limitadas ou inconsequentes. Para além disso, muito do conteúdo pode parecer genérico e criado para agradar a todos os jogadores, ao contrário de uma experiência mais personalizada, onde as ações do jogador ajudam a desenvolver a história.

A grande indústria dos videogames tem frequentemente ignorado este facto e desenvolvido sistemas de Inteligência Artificial e de Modelagem de Jogador bastante básicos, que não correspondem as expectativas, apesar de já se ter provado em estudos académicos que estes melhoram o interesse e diversão do jogador. Acreditamos que uma experiência mais customizada e diferente entre jogadores com características, motivações e preferências diferentes, pode adicionar mais valor ao jogo, assim como proporcionar mais diversão e uma melhor narrativa interativa a qualquer RPG.

Neste trabalho apresentamos uma arquitectura de Modelagem de Jogador que usa uma instância de Aprendizagem Máquina que analisa as ações e interacções do jogador com o mundo virtual e associa-as a um perfil de jogador, de maneira a criar uma experiência mais personalizada que deverá proporcionar mais diversão e imersão para o jogador. Este sistema foi implementado no popular videogame The Elder Scrolls V: Skyrim e publicado como uma modificação ao jogo original (mod), o qual foi recebido com críticas muito positivas pela comunidade de jogadores.

Keywords: Modelagem de Jogador, Narrativa Interativa, Videogames de Role-playing, Árvore de Decisão, Modelos de Jogador, Aprendizagem Máquina
# Contents

<table>
<thead>
<tr>
<th>List of Tables</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>Acronyms</td>
<td>xiii</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Context ................................................. 3
1.2 Motivation .............................................. 4
1.3 Objectives ............................................. 5

## 2 Literature Review

2.1 Interactive Storytelling .................................. 7
2.2 Drama Managers .......................................... 8
    2.2.1 Centralized Drama Manager .......................... 8
        Mimesis Virtual Aquarium ............................ 9
    2.2.2 Distributed Drama Manager ......................... 9
        Virtual Dinner Party ................................ 9
    2.2.3 Mixed Drama Manager ............................... 9
        Façade .............................................. 10
        Radiant AI ........................................ 10
    2.2.4 Other Related Work on Drama Managers ............... 11
        Player Modeling and Drama Management Modules ..... 11
2.3 Player Modeling .......................................... 12
    2.3.1 Gathering Player Data .............................. 13
    2.3.2 Objective Input .................................... 13
    2.3.3 Player Profile Input ............................... 13
    2.3.4 Player Modeling Applications ..................... 13
2.4 Player Type Classifications ................................ 14
    2.4.1 History ........................................... 14
    2.4.2 Aspects of Adventure Gaming ....................... 14
    2.4.3 Players Who Suit MUDs ............................ 15
    2.4.4 Robin’s Laws of Good Game Mastering .............. 16
    2.4.5 BrainHex ........................................ 16
    2.4.6 Other Related Work ............................... 17
2.5 Quest Types ............................................. 17
    2.5.1 Time, Place and Objective Oriented ................ 17
    2.5.2 NPC Motivations ................................... 18
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Drama Manager systems and some of their characteristics, strong and weak points. Adapted from [1].</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>Attributes for the Decision Tree Classifier</td>
<td>36</td>
</tr>
<tr>
<td>5.2</td>
<td>Classification trees performance comparison overview.</td>
<td>40</td>
</tr>
<tr>
<td>5.3</td>
<td>Quest type preference per player type. The number of plus signs represent the level of interest in a quest type from a particular player type.</td>
<td>45</td>
</tr>
<tr>
<td>5.4</td>
<td>Created quest structures.</td>
<td>47</td>
</tr>
<tr>
<td>6.1</td>
<td>Comparison of the questionnaire answers per Game Experience Questionnaire (GEQ) area.</td>
<td>54</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of a Branching Story Graph.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Mimesis screenshot from [2].</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Virtual Dinner Party screenshot from [3].</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Façade screenshot from [4].</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>Player Modeling (adapted from [5]).</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Bartle player taxonomy chart.</td>
<td>15</td>
</tr>
<tr>
<td>2.7</td>
<td>BrainHex Seeker class description.</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>Quest proposition in Fallout 4.</td>
<td>18</td>
</tr>
<tr>
<td>2.9</td>
<td>Persistent agent on SecondLife (adapted from [6]).</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>The Elder Scrolls V: Skyrim gameplay.</td>
<td>23</td>
</tr>
<tr>
<td>3.2</td>
<td>The Radiant Story system causing an argument between several Non-player characters (NPCs).</td>
<td>25</td>
</tr>
<tr>
<td>4.1</td>
<td>Solution general concept.</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Proposed player modeling solution architecture.</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>Class selection in TESIV: Oblivion.</td>
<td>30</td>
</tr>
<tr>
<td>5.1</td>
<td>Implemented solution in TESV: Skyrim.</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Training Data Set used.</td>
<td>37</td>
</tr>
<tr>
<td>5.3</td>
<td>Example of Weka 3 usage.</td>
<td>38</td>
</tr>
<tr>
<td>5.4</td>
<td>Decision Tree generated by the J48 algorithm execution over a Robin Laws classification.</td>
<td>39</td>
</tr>
<tr>
<td>5.5</td>
<td>Decision Tree generated by the J48 algorithm execution over a Richard Bartle classification.</td>
<td>39</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison between performance measures from the two created classifiers.</td>
<td>41</td>
</tr>
<tr>
<td>5.7</td>
<td>Section of the Decision Tree implementation in Papyrus.</td>
<td>43</td>
</tr>
<tr>
<td>5.8</td>
<td>Example of a delivery quest letter.</td>
<td>46</td>
</tr>
<tr>
<td>5.9</td>
<td>Example of a bounty quest letter.</td>
<td>46</td>
</tr>
<tr>
<td>5.10</td>
<td>Quest creation window in Creation Kit.</td>
<td>47</td>
</tr>
<tr>
<td>5.11</td>
<td>Example of a quest narrative.</td>
<td>48</td>
</tr>
<tr>
<td>5.12</td>
<td>Other example of a quest narrative.</td>
<td>48</td>
</tr>
<tr>
<td>6.1</td>
<td>Player modeling question in the demographic survey.</td>
<td>52</td>
</tr>
<tr>
<td>6.2</td>
<td>GEQ areas score comparison.</td>
<td>55</td>
</tr>
<tr>
<td>6.3</td>
<td>&quot;Your Own Skyrim&quot; in the top weekly and monthly mods from the TESV:Skyrim Steam Workshop.</td>
<td>57</td>
</tr>
<tr>
<td>6.4</td>
<td>Online survey on YOS mod summary.</td>
<td>58</td>
</tr>
<tr>
<td>6.5</td>
<td>Online survey PM techniques opinion.</td>
<td>58</td>
</tr>
<tr>
<td>6.6</td>
<td>Summary of some answers from the mod evaluation online survey.</td>
<td>59</td>
</tr>
</tbody>
</table>
Acronyms

AI  Artificial Intelligence. 3–5, 19, 24, 48, 61

CSV  Comma-separated values. 38

DM  Drama Manager. 3, 8–10, 12, 25

DT  Decision Tree. 34, 36, 38, 40–42, 52, 53, 58

FRP  Fantasy Role-playing. 14

GEQ  Game Experience Questionnaire. xi, xii, 53–56

GM  Game Master. 14

IS  Interactive Storytelling. 3–5, 7, 12, 51, 61

MATLAB  MATrix LABoratory. 37

MDL  Minimum Description Length. 38

ML  Machine Learning. 19–21, 27, 29–31, 62

MLR  Motivated Reinforcement Learning. 20

MMORPG  Massively Multiplayer Online Role-playing Game. 14, 17, 19, 29, 60

MUD  Multi-user Dungeon. 15, 17

NPC  Non-player character. xi, 3, 11, 15, 18–20, 23, 26, 29–31, 35, 45, 48, 62

PACE  Player Appraisal Controlling Emotions. 11

PC  Personal Computer. 51

PM  Player Modeling. 3–5, 12, 17, 27, 51, 56–58, 60–62

PMM  Player Modeling Model. 11, 12

RPG  Role-Playing Game. 3, 5, 7, 14, 16, 18, 23, 27, 29, 44, 51, 56–58, 60, 62

SM  Story Manager. 25

TESV:Skyrim  The Elder Scrolls V: Skyrim. 4, 5, 10, 18, 23–26, 33, 35, 36, 43, 44, 51, 54, 56, 57
UI User Interface. 26

VR Virtual Reality. 23 24

Weka Waikato Environment for Knowledge Analysis. 33 37
Chapter 1

Introduction

1.1 Context

The concept of Interactive Storytelling (IS) defines the capacity that a player has in actively taking part in a narrative, molding it with his actions or making it progress through his decisions. In an interactive storytelling experience, the story is not as well structured as it is in traditional storytelling. Instead of a linear narrative where the author tries to present the user with a predetermined and sequential set of plot events, in IS the player keeps some of his agency, making the story more unique, engaging and personal [7].

More sophisticated systems have been object of academic research and make use of an Artificial Intelligence (AI) that manages the narrative and tries to keep the plot coherent or more interesting for the player. This AI entity is often called Drama Manager (DM) [8] and is a background agent that can manipulate the fictional world and other actors in order to gently guide the player to a certain plot point or make the story progress. Many of these DMs try to create a player profile, analyzing his behavior in an attempt to understand and map certain key properties, such as preferences and play style so that a more personal experience can be created for him [7, 9].

The creation and use of player profiles in an interactive experience is called Player Modeling (PM) [10]. These profiles can be created through several methods and can have a large variety of applications in video games, such as adapting the game difficulty, creating personalized content and believable agents, play testing analysis and monetization of free-to-play games [11]. The use of player modeling in video games makes for a more rich and diversified experience as much of the game content, sometimes also including the story, is created according to player preferences.

In the scope of modern video games, especially in Role-Playing Games (RPGs), player modeling is an essential feature. The creation of more immersive and engaging experiences can be achieved by making an analysis of the user and tracing his interactions with the virtual world. This analysis can then produce a player model to be used by a DM system to change some aspects of the game such as story, game content, characters actions and objectives. The use of player modeling and interactive storytelling techniques can be very beneficial to modern video games because they have the potential to craft much more player-focused experiences, improving player expected enjoyment and engagement, as well as adding replay value to the game.

Today there are some commercial systems that make use of IS techniques, such as the Radiant AI system in Bethesda’s The Elder Scrolls V: Skyrim (Bethesda 2011). This system allows the game engine to create dynamic reactions both from the virtual world and the NPCs to the player actions. Its

---

1Player agency defines the capacity that a user has in being able to make meaningful decisions about their actions, with regards to the game world.
objective is to create a more immersive and non-linear narrative and thus a more personal, believable and unpredictable experience for the player. This system keeps track of the player actions and decisions, as well as the attributes and achievements of the character controlled by the player - player character - in order to properly adjust the game reactions to him.

1.2 Motivation

The video game industry is growing more and more each day and with that, new IS techniques and enjoyable stories are needed more than ever in order to differentiate video games and make them unique. Many techniques have been proposed and implemented on academic environments but not on many commercial systems. The player community is constantly growing and getting more diversified and with that the need to make each experience unique and satisfying is essential. Interactive storytelling and player modeling can do a great deal in that regard, tailoring a narrative experience where the player is the central focus and the story is built around him.

Taking a look at the commercial Radiant AI system present in The Elder Scrolls V: Skyrim (TESV:Skyrim), it makes the virtual world more alive and believable and allows part of the game content and dynamic quests to be generated using some attributes from the player character. However, a deeper analysis of this system reveals that the player modeling techniques used are very basic and there is a lot of room for improvement. A major part of the narrative in TESV:Skyrim continues to be linear, predetermined and does not take into account the player preferences, convictions and play style. Even though the game possesses numerous quests and plots, all of them - with the exception of those dynamically created by the Radiant AI system - are started and develop in exactly the same way across playthroughs. Because of this narrative linear structure, the liberty of choice provided to the player is reduced. As a consequence of this, the game experience can be repetitive and monotonous after a few playthroughs and the immersion and overall satisfaction given by the gaming experience can be reduced. If the player actions are not meaningful, the player takes the role of a spectator of the action, instead of giving him the main role on the narrative, as it is supposed in IS.

Despite the academic research that has been done in the areas of Interactive Storytelling and Player Modeling which prove that an investment in these can be fruitful [12, 13, 14, 15], modern video games still use some very basic implementations of these systems. In order to empower the player (i.e. create the feeling that he is indeed actively participating in the story) and make him feel that his interactions with the virtual world and other characters truly affect the overall narrative, the game should provide the player the maximum liberty to choose and explore all the possible options and adapt itself to it. If the game does not react accordingly to the choices of the player, the interactive experience can be seen as a sequence of paths between plot points. More precisely, the virtual world should feel alive and be tailored with the player as the main focus, i.e. dynamically mutate to please the player tastes. The narrative should also still feel enjoyable, independently of player choices and likes/dislikes and it should adapt itself to the player and not the opposite.

In summary, this work addresses the following problem: Should the use of sophisticated AI techniques that use player preferences to change some aspects of the game, i.e. PM, be more often implemented in commercial role-playing games? And if so, do they improve the narrative interaction, overall game quality and satisfaction from the player perspective?
1.3 Objectives

The main objectives of this work is to propose and validate a player modeling approach to be applied in commercial RPGs in order to create a larger diversity in the form of game side-stories and content. With that, we hope to create a more diversified, dynamic, personal and overall better experience for the player. The improvements are focused on a new player modeling component that aims to present the player custom content that fits a player model chosen by a classifier (e.g. a decision tree). While the game engine is responsible for monitoring the player behavior and actions and record them, the classifier acts in the background while the player is playing to classify the player in one of several player archetypes. This archetype will then be used as input to another system that creates specific quests that should better fit the player profile and maximize his enjoyment. For instance, if the player has done a lot of exploring, the game will present the player some quests that will offer the player the possibility to explore new areas of the virtual world. This player archetype is not set in stone for the first time the decision tree chose it. Instead, the classifier continuously run to scan the player actions and periodically checks if the player play style has changed, updating the player assigned archetype if needed.

To create a larger diversity in the game side stories and objectives, this new system is able to generate dynamic stories in runtime, parallel to the main story, based on the most usual actions of the player and his in-game attributes, e.g. the player character level and attributes skill level. These changes aim to create personalized, captivating, interesting and rewarding adventures, so that the player does not feel like he is being “forced” to follow a certain path or completing the same generic objectives across playthroughs.

The proposed solution was implemented in the commercial video game TESV:Skyrim. Overall, TESV:Skyrim video game offers a large diversity of missions and stories for the player to participate. However, like in many RPGs, the objectives are very bland and generic, created to please all the players. Not all players play the same way and in all video games there are things that they like more and others that they like less. With the player modeling component developed, we want to offer the appropriate content to the appropriate players.

The result of the modifications and extension to the previously existing AI system in TESV:Skyrim were released to the public as a mod (game modification), to be focus of testing and discussion by the online video game community and an offline testing group. The results of those tests and the given feedback used to evaluate the overall quality of the implemented solution and to give substantial evidence that PM systems can indeed have a large number of applications and be beneficial to RPGs when properly integrated in a video game.

In short, the objectives of this work are the following:

1. Develop a general architecture for role-playing video games that creates tailored content according to the preferences of the player, including,
   - a player modeling module with a classifier that analyzes player behavior during playtime;
   - a mapping from the player model to sets of types of quests and missions;
   - a library of quests and mission templates to be offered to the player;
   - a system to deliver and integrate those missions within the general framework of the game.

2. Implement the aforementioned system in a commercial video game and validate it with the player community.

3. Determine whether PM systems like the one presented enhance player enjoyment, narrative interest and immersion in commercial RPGs.
Chapter 2

Literature Review

2.1 Interactive Storytelling

Interactive Storytelling is a concept that has been widely explored, researched and developed since the 1970s. In this digital entertainment style, the storyline is not predetermined and the narrative and its evolution can be influenced and changed by the user interactions with the experience [8].

The most difficult aspect of Interactive Storytelling from an author’s perspective is the ability of the game system to build a coherent narrative that preserves the creativity of the author and at the same time gives the player enough autonomy to influence the plot with his actions and decisions [16]. The way that the player interacts with the game is either by assigning him the role of a character in the virtual world or making him the director of the action [17].

In the first scenario, the player interacts with other computer-controlled characters or objects in a virtual world, completing objectives, making decisions and taking actions so that the story progresses through a certain path. In the second scenario, the player assumes the role of the director of the narrative and can directly manipulate the virtual world state and the actors. In either case, the main objective of IS is to create interesting stories where the player can actively participate, changing the course of the narrative.

In order to create an interactive narrative experience some techniques can be used. The simplest and oldest one is the one present in choose-your-own-adventure books, where the reader is the protagonist of the story and is, in certain parts of the story, presented with a set of options (i.e. decisions) that he has to choose from in order for the story to advance in a certain direction [18]. Following a certain path will then lead to another set of options or to one of several endings. Many video games use this simple technique to create a interactive experience. However, this interaction is very limited as the player is just choosing an option when a set of these are presented to him.

Interactive Storytelling in RPGs can be achieved through several approaches. A simple yet largely used technique, specially in commercial systems, is to build a direct acyclic graph, also called branching story graph where each node contains a point in the story where the player has to make a decision (e.g. plot point) and the arcs represent the paths that the player can follow in the story (Figure 2.1). More sophisticated implementations try to maximize the expected enjoyment of the player through several methods, such as creating player profiles [19] and trying to match the player with a predefined profile, using previous players feedback to create more appealing stories [20] or using probabilities that predict which story branch the player is more likely to follow [9].
2.2 Drama Managers

In order to keep the narrative coherent and to apply the changes made by the user to the story, a Drama Manager is often used. A Drama Manager (DM) is a background agent that monitors an interactive experience and intervenes in order to shape the global experience so that it reflects the user’s actions or choices, keeping the expressive goals by the author at the same time.

Drama Manager systems can be analyzed and categorized according to the level of autonomy that each implementation gives to the computer-controlled agents in order for the story to be created solely on the interactions between the player and these agents. These systems have been divided into three types - Centralized Drama Management, Distributed Drama Management and Mixed Drama Management.

2.2.1 Centralized Drama Manager

Centralized Drama Management systems have a central entity that reasons and chooses actions for all the characters and objects in the story. These systems are plot-driven and do not give neither the player nor the agents much autonomy to influence the story. A central entity represented as a DM has full control of the actions performed by the characters and tries to react to the interactions and choices that the player takes in the virtual world, intervening and adapting the story in order to keep a coherent narrative. In this category the system tries to craft a story for the player, adapting and changing the plot when the player interacts in a certain way with the virtual world or the characters in it.

Instead of a solid, structured and predefined narrative, the system smoothly guides the player to certain plot point in order to create an interesting story. The choice of a path for the player to follow depends on the internal architecture of the DM, either by creating a player model, maximizing an experience-quality function or keeping the player on an emotional trajectory. What all this architectures have in common is that the narrative is shaped and determined by the manipulation of the virtual world by the player, be his interaction with objects or other characters of the story.

This story is constructed by letting the DM choose and guide the player to a particular plot point from a set of possible transitions between plot points. This makes it so that the player always follow a structured plot and a consistent story as desired by the narrative author.
Mimesis Virtual Aquarium

An example of a centralized drama manager was implemented in Mimesis Virtual Aquarium, an educational game created by Young et al.[2] in association with a game development studio. In this scenario, the DM generates plans for a customized guided tours of the Monterey Bay Aquarium, giving information about the environment that the player looked at and were based on the state of the virtual environment which included the position of the tour guide and the inhabitants of the habitats.

2.2.2 Distributed Drama Manager

Distributed Drama Management systems are based on completely autonomous agents that have the capacity to reason and choose their own actions and interactions between them and other agents or the virtual world. In this category the stories that are created completely emerge from the actions taken by the individual agents and the interactions between them, avoiding a predefined plot or a set of possible plots. However, the computation of these individual agents can quickly become too heavy, specially in modern complex virtual worlds, since each agent has some information regarding the world and story states.

These systems give agents total autonomy to deliberate and choose their objectives and interactions with the world, other agents and the player. The plot emerges based only in these interactions. This type of narrative is designated Emergent Narrative and its main features are the character driven narrative and the variety and unpredictability of the story development, which can follow any path within the world story.

Virtual Dinner Party

An example of a distributed drama manager is present in the interactive drama Virtual Dinner Party [3]. This interactive experience allows multiple users to interact with several characters in a client-server architecture. Each character receives a role with a predefined set of attributes such as gender and status and a limited set of interaction options (e.g. couple in love, jealous ex-partner, etc). The story then emerges from the actions that the agents and users choose for their characters.

2.2.3 Mixed Drama Manager

Mixed Drama Management systems are the most complex and mix the best of the two previous categories. The system gives enough autonomy to the agents and the player, while trying to keep them in the right direction if they stray too far away from the plot.
The characters have a degree of autonomy to reason and interact with the virtual world and other characters in the scope of the current plot point, creating a plot line that a central DM tries to keep intact and fluid. If either the player or the characters seem to stray away from this predicted path, the DM is responsible for hinting both the characters or the player to a path that makes sense in the context of the story. There is always a trade-off between how much autonomy can be given to individual characters and how much the story can be controlled in order to keep the global narrative and managing story goals. The existence of trade-off exists is often referred to as Narrative Paradox.

In this type of systems, the agents are autonomous enough to reason and perform several interactions with the virtual world and the player that are appropriate in the context of the story. However, a DM supervises the global plot and can order the agents to follow a certain direction in the narrative or hint the player in order to keep the development of the story, preventing the player from being stuck in a certain plot point or fall into a certain plot point that contradicts the global story. This system makes sure that certain events happen while maintaining the story structured.

**Façade**

Matheas and Stern [4] created what is known as the first truly playable interactive drama - Façade. It is an interactive drama about human relationships. In this interactive drama, the user assumes the role of protagonist and interacts with the characters of the drama, communicating with them through text, in order to build a believable and coherent story. The main feature of this experience is that the user has to deal with the emotional aspect of two other characters, creating a plot around those emotions and dialogues. Contrarily to the classic dramatic methodology where the user plays the role of observer of the story, in this environment the user actively and directly takes part in the plot, making the narrative component the main focus of this experience.

This project is considered by many the first real Interactive Drama since the player actions have a significant influence in what events occur, those that are left out and how the drama ends. The drama takes place in a small 3D virtual world, the apartment of a married couple in which the player and the couple, Grace and Trip, can continuously move anywhere, interact with the world objects and speak. Grace and Trip try to engage the player in conversations and are designed to respond robustly to a variety of open-ended dialog from the player, including questions and provocations. These actions then affect the relation between the characters and the player, changing the tension level and therefore enacting a drama. A Mixed Drama Manager gives the agents have a certain degree of autonomy in their actions, conversations and relationship with the player and incorporates them into a larger scale of the story structure, guiding the characters to certain plot points in order to make the story progress.

Façade is considered to have a Mixed Drama Manager because the character agents select their own local behavior from the current plot point that fits their goals and situation, while a DM chooses and connects the beats. Indeed, in Façade, the agent controlled characters can reason about what actions they will perform, according to the events that they have witnessed and their adopted goals. The player has the autonomy to steer from the story as much as he wants and the systems will adapt the story and characters behaviors in order to keep the plot coherent and fluid. If necessary, the DM will also directly change the goals or reasoning of the agents if the player gets stuck in a plot point or if the story will follow a path that contradicts itself or is too strayed from the path that the DM predicted.

**Radiant AI**

The video game TESV:Skyrim makes use of a DM to make the virtual world more alive and the agents more believable. This system, called Radiant AI, is capable of creating dynamic reactions to the changes
that the player makes in the environment. More so, the agents can quickly adapt their objectives and actions in response to the player. Overall, the system can be considered a mixed drama manager, since the NPCs actions are determined by this system when a change occurs in the game. For the most part, the NPCs have their own reasoning and follow their own objectives, like daily tasks, routines, etc. However, when certain actions are performed by the player, an event is triggered that changes the reasoning of the NPCs and makes them adapt to the new state of the world. A more detailed analysis of this system is performed on Chapter 3.

2.2.4 Other Related Work on Drama Managers

An approach on Drama Managers was postulated by Nelson and Mateas [19], where they argued that optimizing an experience-quality function does not destroy interactive agency and proposed a DM based on an offline cached search as done by targeted trajectory distribution Markov decision processes (TTD-MDPs). This algorithm tries to predict the next player decision or action, based on the player previous actions and tries to guide the player through a certain direction. This systems tries to avoid low-rated experiences and increase the probability of highly-rated ones according to a function. The TTD-MDP operates offline, sampling several possible trajectories through the story world and builds them into a tree, solving an optimization problem at each node of the tree, which represent plot points.

Hernandez et al [7] argued that, in order to maximize the player enjoyment, the game should guide him through a series of emotional states, using an AI Experience Manager that targets an emotional trajectory and tries to keep the player on an emotional track, i.e. in a sequence of narrative segments consistent with the player's in-game action. In this work, they implement a system named Player Appraisal Controlling Emotions (PACE), which is an AI Experience Manager that allows the author to specify a large variety of emotions and recognizes that a narrative event can provoke different emotional responses from different types of players. PACE is a centralized drama management system as it has full control over the entire narrative, including all NPCs.

Table 2.1: Drama Manager systems and some of their characteristics, strong and weak points. Adapted from [1].

<table>
<thead>
<tr>
<th>Category</th>
<th>Plot Driven</th>
<th>Agent Autonomy</th>
<th>Story Variation</th>
<th>Player Autonomy</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Distributed</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Mixed</td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>0</td>
</tr>
</tbody>
</table>

Player Modeling and Drama Management Modules

In 2007, Sharma et al [20] argued that a majority of story-based game experiences followed fixed scripts predefined by the author and that previous drama management components ignored the personal preferences of the players. They then proposed a drama manager that tries to predict an interestingness value for a particular plot point based on player interactions with the story. This drama manager works with an expectimax algorithm and combines the author specified aesthetic with the player model and is composed of 3 components: a game engine, responsible for running and game and interacting with the player, a player modeling module, that analyzes the actions of the current player and develops a player model, and a drama management module, that influences the development of the game, making it more appealing to the player.

The Player Modeling Model (PMM) builds a player model using feedback provided at the end of each game and traces the player actions during the playthrough. This feedback is composed of the most and
least interesting moments during the game, from the perspective of the user. On a particular playthrough, the PMM compares the player interactions with those of previous players and tries to associate the prayer profile with a player model. The feedback from the previous players is then used by the PMM to predict the stories that the player is more likely to enjoy, according to the player model that the user fits the best. This information is then used by the drama manager in order to change the plot, using drama manager actions to shape the narrative. These actions are divided into two categories: Causers and Deniers. Causers hint or guide the player through a certain direction in the story, while Deniers hint or prevent the player from doing so. The goal of the DM in this particular architecture is to use drama manager actions in order to design a storyline that the player is expected to have more interest in, according to the player model.

2.3 Player Modeling

A brief explanation on a PM system was introduced in Subsection 2.2.4. In this section we take a closed inspection on the PM concept from a general perspective and over some IS instances.

In order for a system to adapt the game to user preferences, changing the narrative and other content, some information has to be retrieved from him. This information is obtained through an analysis of players cognitive, affective and behavioral patterns in order to create a model that expresses their personality, intentions and characteristics [22]. Player Modeling, from a general perspective, is the study of this data and it makes possible to perceive, record and analyze the player actions, detecting certain patterns in his interactions with the virtual world, to create a statistical model. This model is then used to create an experience that the player is more likely to enjoy and be interested in.

Since there is no optimal experience that fits all players, the game designers have to create several fragments of game content or, at least, some alternatives for a common task that are likely to please a certain groups of players. These fragments are then put together when the game is being played and are selected according to the preferences of who is playing it. This allows game designers to showcase their talent and customized detail while at the same time pleasing a larger variety of individuals. Instead of a linear experience that could be played always the same way and that could include some parts that may not be so appealing to certain players, a player model is used online so that certain parts of game content are assembled together to form a coherent experience that features personalized content for the player living the experience.

Figure 2.5: Player Modeling (adapted from [5]).
2.3.1 Gathering Player Data

The input for these player models can be captured through different ways. Gameplay input is generated by the player in-game interactions. This data corresponds game metrics such as the time the player spent in a certain task or the choices the player made until a certain point in the game. This metrics are usually used to make the correspondence between interactions and mental states, such as attention, frustration, challenge and engagement.

The choices or paths in the story took by the player can also provide some relevant information when understanding his motivations and preferences and what could be more or less interesting for him. Of course, in order to create player models with this type of input it is assumed that player actions and preferences are directly linked. However, the game experience may affect the cognitive processing patterns and focus of the player.

2.3.2 Objective Input

Objective input is the physiological feedback that is unconsciously originated by the user when certain events occur in the game. Some systems monitor player posture, facial expressions, heart rate, respiration and others in order to better comprehend the effect that certain game changes provoked on him. These measurements, specially when combined with other input can be of great use when creating player models.

Although studies prove that these measures can effectively capture the players’ response to certain game stimuli, the gadgets used to obtain this data are sometimes seen as intrusive by him and this can influence the overall experience and the input obtained [23]. Some developments have been done in this area so that the equipment used is made less intrusive and is still capable of capturing important involuntary data from him, e.g., mouse incorporate sensors and eye tracking devices [24].

2.3.3 Player Profile Input

Player Profile Input is all the static data related to the player and that is not necessarily related to gameplay. Player profile input mainly include information on player personality, background, culture related characteristics, age and sex. Of course every player is unique in certain ways and have different preferences and play styles that the game should take into account. However, the user personality and behavior doesn’t necessarily corresponds to the real-life counterpart and this has to be taken into account when trying to create authentic player models [25].

2.3.4 Player Modeling Applications

Player Modeling has a large variety of applications. The first and more important one is to adapt the game experience to the player. This includes the narrative, game content, mechanics, hints, levels, difficulty, etc. A good difficulty balance can be a crucial feature when adapting the game to the player. For some players, a good challenge is essential to keep them interested in the game, while for other players a higher difficulty can make them frustrated, often leading to disinterest. The system captures and analyses some input given directly or indirectly by the player to decided what should be best for the player.

For example, if the player is struggling or keeps getting stuck in a certain part of a level or story, the system can assume that the difficulty is too high for him and it then proceeds to lower the difficulty of the game, giving him some hints or creating an alternative way to successfully beat said part. On the other
hand, if the player is having no difficulty at all, the system may try to throw him some new challenges in order to keep the experience more exciting and engaging.

Other application is the creation of personalized game content. Modern days sometimes use Procedural content generation (PCG) that automatically generates a lot of content with limited memory resources [26]. This content can also include story segments or NPCs behavior and is sometimes the result of search based algorithms that try to adjust accordingly to player preferences or expected enjoyment.

The creation of believable agents is another application of player modeling [26]. Some systems make use of human user models to change the behavior of the agents present in the game to make them feel more "natural" or less robotic. This application can clearly be seen in some competitive video games, such as racing and shooting games, where the agents competing with the player, often called bots, simulate human behavior. These systems are capable of simulating human behavior to a certain degree, most of the times even including mistakes. Many more applications could be listed and explained here, but those presented were the most interesting to study and describe.

2.4 Player Type Classifications

2.4.1 History

Over the course of the years, some effort was made into classifying different types of role-playing players into categories, creating many different categorizations. First, these were created to help Game Masters (GMs) to create narrative structure and more interesting tabletop role-playing adventures (e.g. Dungeons & Dragons, Tunnels & Trolls, RuneQuest) and MUDs (Multi User Arena) and were later adapted to the video game scene. Specifically in adventure gaming, also known as Fantasy Role-playing (FRP) these classifications would help create tailored adventures and narratives for the participants.

Even though players follow identical sets of rules, they do not necessarily use them in the same way or for the same purpose, i.e. different types of players may have a different point of view, motivations, interests and dislikes on the same set of rules. The adaptation of these player models to video games is performed in a natural fashion, as both RPGs and Massively Multiplayer Online Role-playing Games (MMORPGs) genres stem from adventure gaming and MUDs [27].

Furthermore, some video games studios have even explicitly included player models in their games (e.g. Kingdom of Loathing [28]), crediting part of the game success to the player classification.

2.4.2 Aspects of Adventure Gaming

One of early documented division of player types was formulated by Glenn Blacow and published in Different Worlds #10 (October 1980) [29]. The author considered that there were four aspects of adventure gaming - Power Gaming, Role-Playing, Wargaming and Story Telling - and that the interaction of those four elements created the feeling of any given adventure. Under these aspects, the GM would have to adapt the adventure accordingly to better please the participants.

The Power Gaming element represents the thirst for power, levels, magic, special abilities, etc. Players from this category often think that the sources of individual strength are what matters. The Role-Playing element focus on the "lives" of the created characters and players who belong in this category are more interested in developing the story and create emotional attachment to their characters.

---

1 Person who acts as an organizer, officiant for questions regarding rules, arbitrator, and moderator for a multi-player role-playing game.
Wargaming represent the players that put more emphasis on the tactical abilities of their characters and the game mechanics. Competition and difficulty are keywords that represent this category.

Storytelling focuses on story, immersion and believable fictional world. History is a continuing and developing process, with the actions of both player and non-player characters affecting the course of events. The player characters are not on the center of the stage, nor the element about which events revolve. The player characters can only act within the tale, and their freedom is somewhat limited.

2.4.3 Players Who Suit MUDs

Richard Bartle published an article [30] where he postulated four different player archetypes for Multi-user Dungeons (MUDs). This classification, which was designed with a multi-player structure in mind, can also be applied to single-player games and separates players according to their preferred actions within the game. This taxonomy is based on a character theory and is the result of a 30 questions test (also known as "Bartle Test of Gamer Psychology") whose answers in percentage values allows the classification of individuals. The model subdivides its classes according to the player interactions and acting on the virtual world or other players and NPCs (Figure 2.6).

The archetypes are Explorers, Killers, Achievers and Socializers. Each archetype behaves differently according to its motivation and is associated with a card suit, in a rather interesting analogy.

Explorers are interested in exploring the imaginary world and game mechanics. They like to give background stories to their characters and interact with the environment they are putted in. Combat and earning experience points or levels are not as important as the feel of discovering something new.

The Killers interest resides in acting on other players or NPCs, usually to demonstrate their power or have the feel of competition. They are not as interested in the stories as they are in action and destruction.

For the Achievers, the interest lies in accumulating riches, maximizing points or having a perfect score. They often collect everything that they can and totally complete the game.

Finally, the preferences of the Socializers lie in interacting with other players and characters. The game is often use as an abstraction to an experience that they can then share with other players. Games where the interactions with other NPCs are relevant and an important part of the gameplay often please this type of player.
2.4.4 Robin’s Laws of Good Game Mastering

Another RPG player type classification was proposed by Robin D. Laws [31] that divided players into seven archetypes: **Power Gamer, Butt-Kicker, Tactician, Specialist, Method Actor, Storyteller and Casual Gamer**.

*Power Gamers* usually search for ways to exploit the game in order to maximize their gameplay, creating their characters with a *min-max* mindset and the story is not relevant for the gaming experience.

*Butt-Kickers* are more combat-focused and see the game as an abstraction to a playground where they can show their superiority.

*Tacticians* like realistic problems and challenging but logical situations where the outcome is delivered from the quality of their actions, i.e. they like to be tested for their problem-solving abilities.

The *Specialists* usually play the same character type in every playthrough. They are usually pleased with situations where their favorite character class shines.

A *Method Actor* would be someone who likes to role-play, creating individual stories for each of his characters and behaving accordingly to his character personality and traits.

Similarly, a *Storyteller* would also be more interested in the role-playing component of the experience, but favor interesting narratives and plots over experience points and numbers, as if the game was a movie or a book.

*Casual Gamers* are usually inexperienced players who see the game as an opportunity to interact with other players. They are not usually interested in having a deep understanding of the game rules and story, but rather want to pass some time and a form of enjoyment.

2.4.5 BrainHex

A more recent player classification, that covers all game genres and has some psychological and physiological foundations, have been postulated by video game industry consultant *International Hobo*. *Brain-Hex* [32] is a player classification based on behavior from seven key elements in the human nervous systems, creating the following seven classes: Seeker, Survivor, Daredevil, Mastermind, Conqueror, Socializer and Achiever.

*Seekers* often find joy in discovering new things, exploring virtual worlds and the mechanics behind the game. *Survivors* like to experience exciting moments, where they are have to take risks and escape from scary threats. *Daredevils* represent players that like action and speed, taking risks and feeling the thrill in a video game experience. *Masterminds* often find enjoyment in solving problems and puzzles in a logical way. They usually like games and moments where they have to perform important decisions. *Conquerors* are pleased with overcoming difficult challenges and defeating hard foes. Both *Socializers* and *Achievers* are essentially the same as described by Richard Bartle. *Socializers* like to beneficially interact with other players, while *Achievers* prefer to complete and explore all the components from a game, often aiming for a 100% game completion rate.

An example of a BrainHex class representation is displayed in Figure [2.7](#).
2.4.6 Other Related Work

Many other models have been postulated. Some of them are very specific for a particular game, e.g. *Dungeons & Dragons* eight player types [33], while others propose an increment or reformulation to the aforementioned models, such as Andrzej User Types [34].

Since RPGs in general provide much more diverse alternative gameplay and customization options, it is only natural that there are so many attempts to describe the psychological traits and preferences from a similar group of individuals. To this day, there is no consensus on the more accurate model, as some models work best on specific game genres or situations and others are too generic.

2.5 Quest Types

With constant game design changes over the years, no consensus was reached regarding a general or genre-specific quest classification. There has been an increasingly interest in classifying RPG quests and missions according to their content, structure or quest-giver motivations. This is specially helpful when trying to build quest generation frameworks, procedural content generation and PM implementations.

Some researchers have published work analyzing quest structure and psychological aspects of a specific video game while others have tried to build a common objective classification for every video game genre or for a specific one.

2.5.1 Time, Place and Objective Oriented

One video game classification consisting on three basic quest types was proposed by Aarseth [35]. The author postulates that video game quests can be divided into three main groups: *time-oriented*, *place-oriented* and *objective-oriented*.

In *time-oriented* quests, the player have to perform a task in a limited set of time; *Place-oriented* quests require the player to find his way to an objective, i.e. go from point A to point B; and *objective-oriented* quests puts the player in a situation where he have to complete one or several interactions with the virtual world (e.g. collect and deliver an item) in order to complete the quest.

---

17

---

Footnote: Quests or missions in RPGs, MMORPGs and MUDs are a set of objectives that a player or a group of players (i.e. party) may complete in order to earn a reward - be it items, money or experience - or to advance a narrative.
This classification was developed thinking in all video game genres at the time, so it is too generic for [RPGs] and [MMORPGs] because narration and quest diversification are two important selling points of this genres. Furthermore, in [TESV:Skyrim] there are no quests with a limited objective completion time and almost all quests have both the place and objective components, so this classification would prove useless when trying to correlate each quest type to a player model.

2.5.2 NPC Motivations

Doran et al. [36] provide an extensive work on common [RPG] quests structures. In their work they make an analysis on the most common objectives in [RPGs] and [MMORPGs] and propose a quest classification model based on seven NPCs motivations: Knowledge, Spirit, Comfort, Reputation, Serenity, Protection, Conquest, Wealth, Potential, Ability and Equipment.

To each of these motivations corresponds a quest structure with at least some specific objectives, which they called "strategies". For instance, a quest given by an NPC with a reputation motive, have "Obtaining rare items", "Killing enemies" and "Visiting a dangerous place" as strategies, each one have a correspondent sequence of actions (e.g. Obtain Item Nx -> Give item Nx to NPC) that form the quest objectives.

The authors later proposed a prototype quest generator based on a structural analysis from four [MMORPGs] [37]. This classification is indeed valid for [TESV:Skyrim], but for the purpose of this work the objective was to relate quest types to player types, therefore the NPC motivations that originated the quest were not as relevant as the quest structure diversity itself.

2.5.3 Structure-Based

Another great analysis on [MMORPG] quest types is provided by Dickey et al. from the player actions perspective [38]. He used World of Warcraft, one of the more popular [MMORPGs] ever, and ToonTown to build a detailed mission classification according to the actions that the player character must perform in order to complete the quest or advance a small narrative, i.e. other than the main plot.

While the classification was built taking into account multi-player video games, it can be adjusted to the single-player component as well. The authors divided the quests into six groups: Bounty, FedEx, Messenger, Collection, Escort and Goodwill.
**Bounty**

In *Bounty* quests, the player character is asked to find and eliminate a specific enemy or group of enemies. The place where the enemies reside is usually in different regions than the one the player character is in, forcing the player to new environments and threats. Particularly in MMORPGs, the bounty character(s) is usually from an higher level than the player character, requiring several players to cooperate in order to complete the objective.

**FedEx**

The *FedEx* quests usually require the player to deliver/pick up an object or delivery from point A to point B. The starting and destination locations are located in different regions, making the player explore the virtual world.

**Messenger**

The *Messenger* quest types are very similar to the *FedEx* quests but they are usually more focused on the interaction between the player with NPCs. They require the player to locate and talk to a NPC to obtain key advice, information or directions.

**Collection**

In *Collection* quests, the player is asked to collect a number of items or ingredients usually earned by killing a creature or enemy. The purpose of these quests is to provide an easy way for the player character to “level-up” by killing several enemies before more difficult quests.

**Escort**

*Escort* quests are quests which requires the player character to escort a NPC from one location to another. Despite being easy in concept, the escorted character is usually fragile and the player must protect it at all costs against enemies from wild encounters. This requires timing and strategy as the player can not just rush to the objective location, but must keep a steady pace for the NPC to keep up with him.

**Goodwill**

*Goodwill* quests are exclusive to multi-player video games as they require higher-level players to help lower-level players. This quest type is particularly useful to new players since it helps them getting used to the video-game environment.

### 2.6 Machine Learning in Video Games

Since machine learning and the AI concept became popular, there have been a number of applications in the video game scene. The more common examples throughout the years are path-finding and decision-making by NPCs. In an effort to develop more better games, the video game industry have invested in more complex AIs and machine learning techniques. The objective varies, ranging from creating more challenging or real-like adversaries, more believable and interesting NPCs a better overall or personalized experience from the player point-of-view and so on.

Some modern game engines development teams, such as the Unity3D development team, seized this opportunity and have integrated Machine Learning (ML) agent systems, tools and resources in the game engine for game developers to use.
2.6.1 Machine Learning in MMORPGs

Classifying Players

The use of clustering of behavior data collected during game play in online multi-player games have also been performed to model players, offering different types of players, adjusted game difficulty. Anagnostou et al. [40] have implemented a CURE data clustering algorithm [41] to classify players into two main groups - the action player and the tactical player - according to their play style. This separation allows the separation of hardcore and casual players, offering action players a more challenging experience and casual gamers a more fun, less challenging experience. The authors argue that the needs of the two types of players are rather contrasting: while action gamers enjoy the challenge and sense of power over the other players, casual gamers look for small doses of entertainment and constant failure and punishment may cause them to abandon the game.

Game Persistent Agents

Merrick et al. [6] presented Motivated Reinforcement Learning (MLR) agents that can explore the virtual world and evolve in response to interesting experiences in massively multi-player online video games. In their work, the authors have implemented a MLR model in Second Life virtual world that allows NPCs to develop new skills according to its environment, motivations and reward signals. The learning process uses Q-learning reinforcement strategy to maximize the expected value of the total reward return over all successive steps [42].

2.6.2 ML Application and Methodology in a First-Person Shooter AI

Some researchers have successfully implemented ML algorithms into a large variety of video game genres. Geisler have used the first-person shooter game Soldier of Fortune 2 to develop an agent that mimics combat behaviors from an experienced player. The author have experimented over several ML techniques including decision trees, Naive Bayes classifiers and neural networks and performed a comparison analysis over them. In his work, the author have specified the features that the algorithms can output in the form of an action such as acceleration, direction of movement, direction of facing...
and jumping. The author then proceeds to describe as he have divided the portion of the game world surrounding the agent into four cone sectors, i.e. north, south, east and west and the chosen attributes, which describe in-game relevant information for the survival and success of the agent, such as player health, number of enemies in a specific sector, distance to an objective, and so on.

The training set for the ML is generated with the game-play of an experienced player. Every couple of milliseconds, the modified version of the game collected the relevant data and the action that was being performed at that time. This methodology created a training set where each entry formed an association between the game environment variables - attributes - and an action - label.

Several decision trees were implemented and other ML techniques and the agent was put in control of the player character and its performance was subject of analysis. In this scenario, the agent controlled the player character playing against other computer-controlled enemies, just like in an usual game-play situation.

During gameplay, the agent collected game environment data and selected an action, according to the decision model that was governing the agent. The author concludes that an artificial neural network would be preferable for an a priori created agent action decision model and a Naive Bayes or ID3 implementation if the agent learning is performed during game-play.

### 2.6.3 Deep Learning

A comprehensive review on Deep Learning methods applied to game playing in video games was performed by Justesen et al. [43]. Deep Learning is a subclass of ML that operates over supervised, semi-supervised or unsupervised data and that is loosely related to information processing and communication patterns in a biological nervous system, i.e. artificial neural networks. In deep learning, a computer model learns to perform classification tasks directly from an hierarchy of previous experiences, receiving as input images, text, or sound [44].

The advantages over classical ML are that Deep Learning does not need human intervention to operate (e.g. feature selection) and Deep Learning scales well with data, i.e. it often continues to improve as the size of the data increases, instead of shallowing [45].

In Justesen et al. article, the authors describe how Deep Learning have been applied to play different video games genres (e.g. arcade games, real-time strategy games, text-based games and open-world games) and what are the requisites for each genre.

Specifically in Text-based video games, a network architecture called LSTM-DQN, that converts text from the world state to a representation and uses Q-values estimates to select the next course of action. This system successfully completed between 96% and 100% of the quests on average in two different text-based games [46].
Chapter 3

Environment

3.1 The Elder Scrolls V: Skyrim Overview

The chosen environment for the implementation and validation of the solution was a single-player open world action role-playing video game developed by Bethesda Game Studios and published by Bethesda Softworks, selling over 23 million copies across six platforms, including two Virtual Reality instances, and receiving several “Game of the Year” and “RPG of the Year” awards.

The player can explore a large open-world region of Skyrim, that is populated with many animals and creatures, inhabiting dungeons, cities, towns, fortresses and villages. There is an immense number of quests that the player can participate, each one with its plot and characters. After an introductory tutorial - that is also the start of the main quest - the main quest can be completed or ignored at the player's preference. Some quests can also be dynamically created through procedural narrative generation by the Radiant AI system, allowing for infinite, if somewhat repetitive, content. The NPCs that inhabit the region may be interacted with by the player in a number of ways and are frequent quest givers, i.e. may create new side-plots in the form of quests.

Being a RPG there is a large customization component present throughout the game. At the be-
ginning of the game, the player creates his character, customizing several traits such as gender, race and appearance. Over the course of the game, the skills that are more frequently used improve, i.e. the more the player uses a certain skill, the more that skill will develop. In the proposed solution, this feature will be explored to select game content that should maximize the player’s expected enjoyment. When the player character skills have improved enough to meet an experience threshold, the player character levels up, allowing the player to select specific skill perks that better customizes the avatar. There are also some factions that have unique stories and quests that the player character may join to further customize the character.

TESV:Skyrim was built upon an engine - Creation Engine - created specifically for it. The Creation Kit is a free development tool also made available by Bethesda that allows users to access all the Creation Engine content, from the sounds and textures to the scripts that implement the game logic and quests. The game engine contains a specific language called Papyrus, which is an object-oriented scripting language used to provide many functionalities to the game, such as dealing with events received from the game, setting variables, magic effects, quest background "logic" and interactions between game components. Due to the engine design, there are many things that are implemented as "quests", such as all the game dialogue, NPC scripted behavior and actual game missions. Of course, these are not visible inside the game but rather serve as a sequence of states containing scripts and behavior, operating exactly the same way the game quests do.

3.1.1 Mods and Modding Community

Although published in 2011, TESV:Skyrim is a video game that still has a large player and fan community that keeps creating game modifications and improvements. It has been released in 6 different platforms, with the latest one being a VR version from April 2018 for the PC. There is a huge interest in game modifications and custom content for TESV:Skyrim. This is possible because of a modding kit released to the public by Bethesda in 2012. This freeware, called Creation Kit gives the user full access to the game engine and systems, allowing the creation or modification of game content in a simple way, with the help of a graphic interface and scripting language, Papyrus. The modified content can then be published to the community free of charge. The potential to explore and modify the game systems and content with this tool was one of the main motivations to use TESV:Skyrim as platform for this work.

3.2 Radiant AI and Radiant Story Systems

The Creation Engine contains a powerful technology that allows for dynamic interactions and reactions from the virtual world and NPCs as well as the creation of quest content and objectives during runtime. This feature is divided into two distinct systems: Radiant AI and Radiant Story systems.

3.2.1 Radiant AI

The module that allows the dynamic quests and other story content to be created is called Radiant AI. This system is a quest generator that creates in-game quests whose location, actors and items are selected in runtime, with no performance loss, instead of being completely predetermined. Most of the game quests are not generated this way. However, many side quests (e.g., bounties, rescue missions, hired muscle) that do not affect the main game narrative are generated using the radiant system.

This AI analyses, for each quest, the game current difficulty settings and the player character attributes and global statistics, creating the quest accordingly. These quests are not very complex, usually

1 Skills are numerical representations of the character ability in 18 areas.
requiring the player to go to a certain location, do something there and then come back for the reward. When selecting the location of the quest, the system also selects one that the player has not visited yet. If the player already visited all the possible locations, then the system will select a random one.

Using this technique, the game can create theoretically infinite quests, since the contents of each of these quests are randomly selected from a large set of options. Even characters and locations added by any game modification can be chosen to be part of the created quest, if they fit certain parameters.

3.2.2 Radiant Story

The Radiant Story system acts as a DM in TESV: Skyrim. Since many of the interactions that the player can perform in the game world are associated with an event node, every time the player performs a certain action, a corresponding event node may be activated. These Event Nodes allow the Radiant Story system to manage the game interactions with the player, NPCs and the virtual world, giving specific instructions to the agents or changing the current game state. Everything that is created dynamically in the game is done through a sequence of states started as a consequence of a player interactions. These quests are a set of states that give specific instructions to the NPCs or change the current game state in some way.

The described event nodes are called Story Manager (SM) Event Nodes. To each one of these nodes is associated a list of possible instructions that can be initialized or updated if certain conditions are met. Let us suppose that there is a specific sword that, once picked up by the player character, will attract a group of enemies. In this scenario, when the player picks up the sword, the Player Add Item SM node is activated and will thoroughly check every possible sequence of actions that can take place according to the picked-up item. To one of those sequence of actions, it is verified if the picked-up item was the specific sword, immediately activating a set of states, i.e. summoning the group of enemies to the player.

In Figure 3.2 there is an example of the Radiant Story system in action. As the player character dropped a valuable item on the ground, the Player Remove Item node was activated, checking the possible actions that the game engine could perform. A validation was performed over the item and it was classified as valuable, initiating a sequence of actions (which is interpreted by the system as a quest) that caused the NPCs to start arguing, eventually leading to a fight. It is important to state that
the NPCs involved are not predetermined and are selected by the Radiant AI system when this “quest” is started.

3.3 Quests creation in the Creation Engine

There is a great analogy that is often made when explaining quests creation in Creation Kit. Designing a quest in Creation Kit is like preparing a theater scene. One have to define and describe all the characters, dialogue scripts, narrative and props. Generally speaking, a quest in TESV:Skyrim is composed of five interlinked components: Stages, Dialogue, Aliases, Objectives and Logic.

The stages are the moments or subdivisions of the whole quest. They function like states in a state machine. The progression of a quest in TESV:Skyrim is nothing more than a transition between stages when certain conditions are met (e.g. talking to an NPC or completing an objective); The dialogue contain all the lines that the NPCs can deliver in the scope of a quest. Usually, the dialogue lines are attached to an alias representing one or several NPCs (e.g. quest giver). Some dialogue options are only accessible when certain conditions are met or if the quest is in a certain stage.

The aliases contain all the references for the items, locations and NPCs used by the quest. Some of these can be filled by the Radiant AI system when the quest is initialized (e.g. quest giver NPC, objectives location). They function like the variables of the quest logic.

The objectives contain the tasks that the player has to fulfill in order to advance in the quest, i.e. advance the quest stage. The aliases can be used as objectives and a marker can be visible in the game map and User Interface (UI) for a specific objective when certain conditions are fulfilled (e.g. when the quest is in a certain stage).

The logic contain all the Papyrus scripts and instructions that control the quest flow. It represents the logic by which all the other parts connect, the pieces of code that run when a certain in-game event is triggered, global variables update, the functionality of certain aliases and behavior of NPCs.
Chapter 4

Solution Architecture

In this chapter we take a look at the proposed solution architecture, describing each one of its components and the execution flow during gameplay. The purpose of the proposed system is to examine the player actions during a predefined period of time, assigning him a player type from a player classification model according to the data gathered, selecting appropriate game content and presenting it to the player in an immersive way. The way that the system is designed makes it possible to be used in any RPG if certain conditions are met. Some of these conditions are delivered by some independent and external components that vary across different video games. These will be explored in this chapter, as well as those from the presented PM system.

4.1 Overview

The proposed solution consists of several components which sequentially interact with each other in a series of cycles. The main components proposed are a Player Classifier Module, a Class Manager, a Content Manager and a Player Lure Module. These components interact with each other and other game elements to function properly.

The Player Classifier Module is the unit responsible to examine the data provided by the game engine containing informations about the actions and interactions performed by the player character during a specified period of time. This component have or interacts with an ML algorithm that performs a search with the data inputed by the game engine to select one of several player types, given a predefined model. The component then outputs the selected player type to the Content Manager component.

The purpose of the Class Manager is to choose a character class to be inputed in the Content Manager, along with the player type. It operates by receiving the skill levels of the player character, analyzing them and selecting the appropriate character class from a predefined set of classes that will

![Figure 4.1: Solution general concept.](image-url)
be used by the Content Manager to select the appropriate content.

The Content Manager receives both the player type and character class to select suitable game content to present to the player. It does so by interacting with a resource pool that contains all the game resources that can be used as personalized content. This component outputs the selected content to the Player Lure Module.

The Player Lure Module is a component that integrates with the game engine to deliver the selected content by the Content Manager to the player in a subtle, immersive way. Its purpose is to provide an abstraction layer between all the background work performed by the other systems and delivers the game engine instructions on how and when to present the content to the player.

Once the Player Lure Module finishes acting on the game engine, a new cycle starts and the player actions, interactions and choices start to be recorded again, thus concluding a full cycle of the presented solution flow.

4.2 Independent and External Components

4.2.1 Player

The player is an essential part of the system. Without player interaction there would be no game and no classification to be made. Despite seeming obvious, the player is the one that generates the data to the Player Classifier Module and the Class Manager, using his avatar to interact with the virtual world.

It is also the player who validates the personalized content presented to him, by either accepting or rejecting it. Every action that the player performs with his character in the virtual world is a potential source of data to be used by the system. In a sense, the video game and the player character are a mere abstraction through which the player communicates with the game engine and expresses his thoughts, psychological traits, game style preference, intentions, likes and dislikes [52].
4.2.2 Game Engine

The game engine functions as the heart and the brain of a video game and is the entity that makes it work. It receives the player physical input and orchestrates all other game components to deliver the player some sort of feedback, be it visual, auditory or physical.

The game engine is, in reality, the component that the player is interacting with through an abstraction such as the video game itself and the character or characters controlled by the player. More importantly, it is the game engine that contains, for a particular point in time, all the information regarding the current state of the game. Because of that, the game engine is the component that, in this solution, will be queried about the player interactions and the skill levels of the character controlled by the player.

Furthermore, this component is the one that ultimately will present the player the content selected by the player classification system.

4.2.3 Character Skills Set

Character skills are a base premise of the RPG genre since they serve as a means for character personalization. Generally speaking, the different types of skill progression in RPGs can be divided into three types: player-defined, game-defined and mixed.

On the first instance, once the player character has reached an experience threshold, i.e. evolved a level, the game will ask the player to assign skill and perk points to different character skills or across a skill tree; in the second instance, the game engine is in full control of the skills and perks developed by the player character and will assign them according to a certain criteria, for instance the most used skills by the character. On the mixed type, the game engine will either assign the skills or perks automatically and will let the player choose the remaining one.

The character skills set is extremely important for the proposed system because they can help define a character class and provide a second layer of personalization over the chosen content, e.g. class-related quest and dialogue. A large majority of RPGs have character classes that are defined by some attributes or use them as guidelines for a number of features, including experience boost if the character used the class specific skills and penalties if they use others, class-limited content, NPC dialogue, etc.

For instance, a "thief" class will have a large focus on skills that will help him be more dexterous and stay unnoticed, while a "mage" class will be more focused on spell skills and so on. Most RPGs will force the player to choose a class during character creation or throughout the game, offering the possibility of changing them or not, in a similar way to the one presented in Figure 4.3.

4.2.4 Player Classification

Player classification is one of the foundations of this system, since a player type is what the ML algorithm will try to obtain to select the personalized content by the Content Manager.

The player classification model divides players into groups according to some psychological traits, interactions with the different game elements and so on. The player characteristics used to classify a player into in a class and the number of classes vary from model to model. The chosen model also has to take into account the game genre and number of players. A player classification model designed around a multi-player experience may not be adequate to a single-player game. Furthermore, all game genres have different characteristics that may or may not be considered by the model. Numerous player classification models for RPGs and MMORPGs were postulated in academic research and some have been described in Chapter 2.

One player type from the selected player classification will be output by the Player Classifier Module and inputted in the Content Manager, allowing it to choose from a large set of content, some that should
4.2.5 Content and Content Pool

Content in video games comes in many forms such as the virtual world itself, NPCs, game appearance style, narrative, mechanics, levels, difficulty level, items and so on [53]. Not all players seek the same experience in a video game and what is preferable to one may not be for another. It is the objective of the presented solution in this work to deliver personalized content to the player according to his preferences. In order to do that, some custom content have to be created or reused from the game to be selected by the Content Manager. It is also necessary to identify which content is suitable for which player type, e.g. a player who prefers action and a challenge may be more interested in an higher difficulty setting or a quest where he have to defeat several difficult enemies.

The Content Pool is a repository of the content that can be selected by the Content Manager. This does not mean that there must be an external database of content, but rather an identification of game features that can be used as personalized content to present the player and to what player type or types it is more suitable for.

4.3 System Components

4.3.1 Player Classifier Module

The Player Classifier Module is a subsystem that periodically queries the game engine for actions, events and interactions that involved the player and interacts with a ML instance in order to identify the player type that better fits a profile according to the data obtained from the game engine. Finally, it outputs the obtain player type from that interaction with a ML algorithm to the Content Manager.

Some of this system features can change, depending on the implementation and game. In a large open world the time interval between which the game engine is queried for player character data should be higher than in a game with a more linear structure or zone-partitioned world. This is because with a small time interval, the player may not have enough time to perform a large variety of actions and interactions and the performed classification could be incorrect. Also, depending on the game engine it may be necessary to store the data in this module instead of querying it periodically.
Machine Learning Instance

The ML instance is the entity responsible for choosing the appropriate player type, according to the inputed values by the Player Classifier Module. The inputed values correspond to the feature values being tested in the ML instance and the output is the chosen label after classification.

It is important to note that a set of relevant features must be defined a priori, when choosing the ML algorithm. The chosen algorithm can range from supervised options such as decision tree, Naive Bayes classification and neural networks, to more complex unsupervised learning algorithms such as clustering algorithms. Of course more sophisticated algorithms would need some model adjustments and the game environment itself should be able to support them. That is the reason why the machine learning instance is represented in the solution model as a separate part of the Player Classifier Module.

4.3.2 Class Manager

The Class Manager obtains the character skills set from the game engine and outputs a character to be used by the Content Manager. This module can be parallelized with the Player Classifier module if the implementation environment allows it, since it does not need the player type to operate. If the game engine already contains some built-in character class model, it can be used directly. Otherwise, a model that makes an association between skills and classes have to be built. Some subclasses can even be used to further personalize the content selection. The chosen character class is sent to the Content Manager in the end of the execution of this module.

4.3.3 Content Manager

The responsible entity for content choice is the Content Manager. This unit receives the player type output by the Player Classifier module and the character class or classes by the Class Manager module and selects fitting content to present to the player. To do this, the module queries the Content Pool on available content, using as criteria the inputed values. The delivered content can come in many forms, inclusively a sequence of instructions, e.g. change difficulty setting, send enemies to attack the player character and start quest N. Ultimately in depends only on implementation and available content. The chosen content is sent to the Player Lure Module at the end of the execution of this unit.

4.3.4 Player Lure Module

The last unit to run is the Player Lure Module. This entity receives the content selected by the Content Manager and provides instructions to the game engine on how, when and where to present the content to the player without breaking the normal game play and making it apparent that the new content came with to the original game. This can be achieved in an unlimited number of ways and the imagination and the game engine restrictions are the only limits. Let us suppose that the Content Manager selected a quest to present to the player. The Player Lure Module will instruct the game engine on how to get the player to be notified of this quest. The module could perhaps instruct the game engine to put an NPC at the end of the next dungeon that the player character enters in and engage him in conversation, giving details about the quest.

Once the player knows about the quest, the Player Lure Module has fulfilled its purpose and a new player analysis can begin, independently if the player accepted the quest or not. It is important to note that the system must not disturb the usual game flow and should be able to identify the appropriate moment to order the game engine to operate, i.e., if the player is in the middle of a main quest, it may not the best time to receive a new quest, as it would break immersion.
Chapter 5
Solution Implementation

5.1 Overview

The proposed player modeling architecture was implemented using some tools such as the Creation Kit and Waikato Environment for Knowledge Analysis (Weka). This system was developed using the available resources and considering game engine limitations.

A full cycle of a player classification is performed according to the following set of actions, performed in a sequential manner: The Player interacts with the video game by playing it. The result of that interaction is composed of a set of actions that the player performed and are recorded by the Game Engine. Every two gameplay hours, an update is performed on the Decision Tree Component (which corresponds to the Player Classifier Module from the proposed solution), which will gather the player actions data and input it to the implemented Decision Tree, which is the materialization of the Machine Learning Classifier from the generic solution.

The decision tree will choose a label that is used as a Player Type and is carried on to the Content Manager. Meanwhile, the Class Manager will also query the Game Engine on the player character’s Skill Set and will choose a Character Class based on the higher leveled skills.

Both the Player Type and the Character Class are inputed to the Content Manager, which will search the Quest Pool and choose a quest structure, while the Radiant AI system fills some dynamic quest objectives and characters. The fully structured Quest is then carried on to the Player Lure Module, which instructs the Game Engine on how to present the Quest to the player.

A full description on the used methodologies and technical details will be explored during the next sections from this chapter. A visual representation of the implementation architecture is visible in Figure 5.1.

5.2 Game Demographics and Player Categorization Questionnaire

The development of this work started with the gathering of some demographic data and playing habits on the player base of TESV:Skyrim. A survey was released both on the game-specific forum of the online discussion website Reddit - the /r/skyrim subreddit - and Facebook. The objectives of this survey were to gather and categorize information on the active player community, understand how players usually play the game and what was the general opinion and interest on player modeling. The survey was active online for one month and had the participation of 2746 subjects. It is important to understand that the majority of the respondents are experienced players that continuously seek information, news and new content in online forums, such as Reddit, where the survey was published.
5.2.1 Questionnaire Structure

The survey was divided in four parts: Demographic Info, Game-play Habits, Skill Set and Player Categorization.

The first section - Demographic Info - consists on purely demographic information, such as age and gender.

The second section - Game-play Habits - consists on a series of questions related to the participant experience level, such as the amount of hours spent in the game and preferred in-game difficulty settings.

The third section - Skill Set - is composed of questions that aim to gather information regarding the in-game skills, such as what skills were considered the most "useful", which ones were less used and what was the respondents’ opinion on skill-based content (such as quests).

The fourth section - Player Categorization - is composed of a series of questions related to the respondents’ opinions on personalized content, player categorization and likes/dislikes. This last section also includes a Robin D. Laws’ player type classification [31], which was the first type of classification considered for this work, but was later replaced by Richard Bartle’s classification [30].

5.2.2 Questionnaire Results Analysis

The survey was a success due to the amount of participants. In summary, the conclusions taken, that are directly related to this work, were that the majority of the surveyed are experienced players, with more than two-hundred hours in the game, that often play on a PC platform using mods and have the opinion that a more personalized content would be beneficial to this video game in particular, when in contrast with a more generic one. It can also be concluded that there is an even distribution of players between the different player types of Robin D. Laws’ player type classification, should it be used by the implemented Decision Tree (DT) classifier.

A deeper analysis to the survey responses and conclusions is made on Chapter 6. The questionnaire and the summary of the responses can be consulted on Appendix A. The results of this survey were the evidence needed to develop the solution of this work, since it was observed that there was interest in a
5.3 Decision Tree Classifier and C4.5 Algorithm

The C4.5 is a decision tree generation algorithm developed by Ross Quinlan [54] as an extension to its previous ID3 algorithm [55]. The C4.5 algorithm operates using a set of training data to build a decision tree (DT), grouping similar partitions of data according to a set of attributes and their information entropy \(^1\). In the context of this work, the training set used to feed the C4.5 algorithm was composed of a series of records that associate some player attributes or stats to a corresponding player type, i.e. an association between in-game actions performed by the player - attributes - and a player type - label. The purpose of the DT is to predict a player type, given a set of attributes as input.

5.3.1 Attributes

As the player navigates through the world of TESV:Skyrim and interacts with it, the game records his actions and keeps track of a collection of data and statistics on the player and the virtual world (e.g. Dungeons Cleared, Creatures Killed, Items Crafted, etc) in the form of global variables/counters that are available by the game engine and can be consulted inside the game, should the player want to. These attributes are divided in five groups: General Stats, Quest Stats, Fighting Stats, Magic Stats, Crafting Stats and Crime Stats.

Since there are 90 attributes that the engine keeps records of, some were discarded and others grouped together to serve as input for the decision tree classifier. There are a number of reasons for a attribute to be discarded: It does not necessarily represent player actions (e.g. Days passed, Hours slept), it is a very specific action that usually just happen under certain quest lines or other circumstances (e.g. Stores Invested In, Times Jailed) or it is too vague or considered not appropriate for a player profile (e.g. Most Gold Carried, Food Eaten, Diseases Contracted). Some other attributes were grouped due to their similarity (e.g. Items Pickpocketed vs Items Stolen, Undead Killed vs Creatures Killed). The reason behind this selection was to end up with a small but diverse set of actions that differ between players and directly reflect the player behavioral states and that the player meant to perform [56]. A higher number of features would not cause an over fitting \(^2\) problem for C4.5 algorithm [57] since it uses the pruning technique to eliminate nodes of the tree that provide little power to classify instances, i.e., only uses the attributes with more information gain.

A total of 16 attributes were chosen to represent the player profile, as listed in Table 5.1. Some of the attributes are directly taken from the game engine while others are the grouping of several attributes, due to their similarities. The attribute Speech Tests are the grouping of Bribes, Intimidations and Persuasion, which are dialogue options that the player can take to reach a certain objective and only appear in certain situations, mainly during quests. The attribute Main Quest Objectives Completed includes main quest and all factions objectives (e.g. Companions Faction, Civil War Faction, Daedric Quests, etc), while the Side Quests Completed attribute groups all other miscellaneous quests, [NPC] favors and minor tasks. The attribute Creatures Killed groups all other creatures that are neither people nor animal, like dragons and trolls. The Items Crafted stat contains all player-made items, which include the making and improvement of weapons and armor and brewed potions. Items Stolen correspond to both pickpocketed items and the ones that the player steal from the game world and lastly, Assault Crimes represent crimes committed by the player other than theft, i.e. assault and murder.

---

\(^1\)Information Entropy - The average amount of information produced by a stochastic source of data.

\(^2\)Over Fitting - The case where a statistical model corresponds too closely or exactly to a particular set of training data and may therefore fail to predict future observations reliably.
5.3.2 Labels

The player types represent the labels for the decision tree classifier and C4.5 algorithm. In a first instance, the player types considered were the ones by Robin D. Laws: **Power Gamer, Butt-Kicker, Tactician, Specialist, Method Actor, Storyteller and Casual Gamer**. However, it was decided that the player type "Butt-Kicker" was to be included in the "Power-Gamer" player type because of their similarities (according to their description by the author) in the context of **TESV:Skyrim**.

These classifications would later be replaced by the one made by Richard Bartle' taxonomy of player types, which divides player types in four archetypes: **Killers, Achievers, Explorers and Socializers**. The reason behind this change was the decrease in relative absolute error and, consequentially, root relative error when comparing the test results of the DTs created by the two player classification models as labels, whose conditions and methodology can be found in the Subsection 5.3.3.

The two classifiers built differ only in the labels used, i.e. the self-declared player type classification. The first classifier, Classifier A, was created using the labels given by Robin D. Laws player type classification, excluding the "butt-kicker" class. Using the same attributes, the second classifier - Classifier B - was created with the labels given by Richard Bartle player type classification.

The classifier A obtained a relative absolute error of 40.01% against 21.68% provided by classifier B when using the training set as test input for the classifier. The explanation for this phenomenon may reside in the fact that classifier B uses four different labels while classifier A uses six. This is only a problem due to the relatively reduced training set size (25 records) as it should not matter at a larger scale.

5.3.3 Building the Training Data Set

The final training data set used to create the classifier was composed of 25 records. Each record was obtained through a controlled play session where each player was asked to play freely for two hours. After the two hour long game session, it was asked the participants to self-classify themselves using both Laws and Bartle player type classifications.

The testing conditions were similar to all participants as a special save file was created specifically for this purpose. This save file contained a generic character named *Iant-Rad* who was placed in a random location in the woods, had already completed some objectives in the main story and learned some magic spells, whose inventory was composed of a broad selection of weapons, armor and crafting ingredients and had all the skills developed to level 30. The save file also had the global statistics (attributes) reseted so that only the actions performed by the player in the duration of the play session were considered. The purpose of this save file was to put the participant in a generic, unbiased situation. This way, it was possible to collect data similar to the one the player creates while playing outside testing scenarios. Since the objective of the experience was to map player actions to player types/models, it was of the utmost importance that the player played according to its player subconscious model.

After each play session, each participant was asked to self-classify himself according to Laws’ player classification using the description given by the author for each player type [31], with the exception of the
"Butt-Kicker" label. The classification resulted in the following: 5 "Casual Gamers", 4 "Method Actors", 4 "Power Gamers", 3 "Specialists", 4 "Storytellers" and 5 "Tacticians". The attributes for each participant were later extracted from the save files generated by each play session, thus creating the first training data set. The second data set, which can be seen in image 5.2, was created through the same process but using the player classification by Bartle and resulted in the following class distribution: 6 "Achievers", 8 "Explorers", 7 "Killers" and 4 "Socializers". All the details of the data sets (i.e. attributes collected and player self-classifications) can be found in Appendix C.

5.3.4 Data Mining Software Used

In order to interpret the collected data, it had to be inserted as parameters to an algorithm or software so that more useful information could be extracted. In the context of this project, the purpose of the collected data was to be inputed as training data in a decision tree generation algorithm. Two different pieces of software that provide such feature were studied, Weka and MATLAB:

- **Weka 3** is a machine learning software written in Java that contains a collection of tools and algorithms for data analysis and predictive modeling [58]. It provides an intuitive graphical user interface for data manipulation and result visualization and includes all the standard data mining tasks, i.e. data preprocessing, clustering, classification, regression, visualization, and feature selection.

- **MATLAB** is a numerical computing environment and programming language developed by MathWorks [59] that is widely used in engineering and scientific applications. It features a high level computer programming language with the same name and can perform a large number of tasks, such as computation of numeric data, creation of graphics for scientific use, modeling, simulating and analysis of data. MATLAB also contains a specific library to build classification trees (Statistics and Machine Learning Toolbox™).

After studying both systems it was decided that Weka 3 was the software to be used because of its accessibility and ease to use. MATLAB also requires the user to have a license and code the solution while Weka 3 is a free-to-use software that offers a graphical interface and browser to data entry and analysis. While MATLAB is a very powerful tool and programming language, Weka 3 provides all the necessary features to build a simple classifier in a comprehensive manner.
5.3.5 Generating the Classifier

The generation of the decision tree with Weka is a process subdivided in some minor tasks. Firstly, the training data set has to be loaded into the software. In order to do this, a Comma-separated values (CSV) file was created with the data set in Figure 5.2 and imported into the program. With this action, Weka displays an overview of the properties of the loaded data, such as the attributes and number of instances. Next, there is a pre-selection of the attributes to be used by the generation algorithm. In our case, the attribute "Player Name" was not relevant for the DT, so we discarded it. After that, comes the algorithm/function choosing. In this phase, the J48 algorithm was chosen, which is the Java implementation in Weka of the C4.5 algorithm [60].

The J48 has some meta parameters that can be adjusted, such as minimum elements in each node, pruning confidence factor, the amount of data used for reduced-error pruning and so on. These were kept on default due to the sample size. Some of the more relevant ones include: 2 minimum instances per leaf, 3 folds for reduced-error pruning, 0.25 confidence factor used for pruning, an Minimum Description Length (MDL) correction set to true and perform tree pruning set to true.

Lastly, some testing option have to be chosen. Because of the relatively small sample size, it was decided that the testing would be done using the training set as test input. The other available option were to perform cross-validation and percentage split. Those were both inviable because they need a larger set to operate correctly [62]. Using the training set as testing set, all the records will traverse the generated classifier and be classified with one of the labels.

5.3.6 Resulting Decision Tree

With all things set, the J48 algorithm ran and created the classifier, which can be seen in Figure 5.5 and a report, which contains a summary, detailed accuracy by class and confusion matrix, present in Appendix B.

---

3Oracle MDL is an algorithm that identifies the attributes that have the greatest influence on a target attribute, discards input...
Figure 5.4: Decision Tree generated by the J48 algorithm execution over a Robin Laws classification.

Figure 5.5: Decision Tree generated by the J48 algorithm execution over a Richard Bartle classification.
Table 5.2: Classification trees performance comparison overview.

<table>
<thead>
<tr>
<th></th>
<th>Correctly Classified Instances</th>
<th>Incorrectly Classified Instances</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robin Laws</td>
<td>19 (76%)</td>
<td>6 (24%)</td>
<td>0.76</td>
<td>0.05</td>
<td>0.699</td>
<td>0.76</td>
<td>0.716</td>
<td>0.687</td>
<td>0.947</td>
</tr>
<tr>
<td>Classification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Richard Bartle</td>
<td>22 (88%)</td>
<td>3 (12%)</td>
<td>0.88</td>
<td>0.039</td>
<td>0.89</td>
<td>0.88</td>
<td>0.876</td>
<td>0.846</td>
<td>0.976</td>
</tr>
<tr>
<td>Classification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The classifier is composed of six decision nodes and seven final leaves. The decision nodes represent a test that is made to a new instance when traversing the graph. These tests are based on the attributes that the J48 determined were the properties with more information gain [63]. Those with less information gain were pruned from the tree. The leaves represent the chosen classification for the new instance, according to the way it navigated the unidirectional graph. Because of the arguments passed on to the J48 algorithm, a leaf is only created if at least two instances of the training set end in that leaf. All the decisions are based on numeric tests due to the nature of all the attributes being numeric.

The generated report is created with the data that the algorithm produced while testing the decision tree with the training data set. A full content of this report can be seen in Appendix [5]. From it, we can conclude that 22 out of 25 (88%) instances were classified correctly, while 3 out of 25 (12%) instances were classified incorrectly. An analysis of the confusing matrix explains which data was correctly and incorrectly classified: all the 6 training set instances containing an the label "Achiever" were correctly classified by the DT as "Achiever" using only their attributes; all the 8 training set instances containing an the label "Explorer" were correctly classified by the DT as "Explorers"; from all the 7 training set instances labeled "Killer", 5 were correctly labeled "Killer" while 1 was labeled "Explorer" and another one gained the label "Socializer"; lastly, from all the 4 training set instances labeled "Socializer", 3 were correctly identified as "Socializer" while 1 was incorrectly labeled "Achiever".

5.3.7 Classifier Performance Analysis

The generated report shows some interesting performance indicators, making possible the comparison between the generated DTs. The considered indicators when comparing both classifiers are: Precision, Recall and F-measure. The three measures use the number of correctly and incorrectly classified instances for each label in different ways to better show the classifier performance. Precision is the number of correctly classified instances divided by the number of all positive results returned by the classifier; Recall is the number of correctly identified instances divided by the number of all the instances that should have been classified correctly; The F-measure considers both the Precision and Recall measures to describe the classifier overall accuracy. It is essentially the harmonic mean of precision and recall [64]. There are some other measures given in the report but those do not necessarily apply to a multi-label classification with attributes composed of discrete values.

From the results analysis it is clear that the classifier delivered from the Richard Bartle player classification has a substantial accuracy advantage over the classifier generated with the Robin Laws player classification. The tests run on the Richard Bartle classifier obtained a Precision value of 0.890 against 0.699, a Recall value of 0.880 vs 0.760 and a F-measure value of 0.876 vs 0.716 against the Robin Laws one. Higher MCC\(^4\) and ROC\(^5\) were also obtained by the Richard Bartle classifier.

One of the reasons behind this discrepancy could reside in the fact that the test set used was the

\(^4\)Matthews correlation coefficient takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes.

\(^5\)The Receiver operating characteristic curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
same as the training set, which was relatively small for the number of labels - 7 vs 4 - used by each classifier. The full report comparison can be found in the appendix \[B\] and an accuracy summary can be seen in the chart from the Figure \[5.6\].

This system was designed as a proof of concept. In a real development scenario, one would need to have a larger training data set and a testing set, both composed of a large diversity of individuals with different game experience levels and background.

### 5.4 Decision Tree Implementation in TESV:Skyrim

The decision tree was implemented using a *Papyrus* script. This script (*DecisionTreeQuestScript*) is attached to a sequence of states which are represented in Creation Kit as a quest (*DecisionTreeQuest*). When the initialization of this sequence of states occur, the game engine fires an event that is caught by the *OnInit* method in the script. This method displays an in-game notification stating that the DT was initialized successfully and calls the *RegisterForSingleUpdate* built-in function to mark the script for an update.

This function receives as a parameter the time interval (in seconds) in which the update will be performed. The parameter inputed corresponds to 7200 seconds (two hours), which is the periodicity that we have decided to run the classifier. It is also important to note that this value is measured in real time seconds, does not increment while there is an open menu (i.e. pause menu, magic selection menu, etc) and is stored when the game is saved. This means that the player can save his progress and close the game and the passed time will remain stored for when the save is loaded.

The *OnUpdate* function corresponds to a decision tree execution and preparing a new quest for the player. It is composed of five parts: choosing a player type (*DecisionTree* method), choosing a player class (*ChooseClass* method), initializing and choosing a quest for the player (*StartQuests and SelectQuest* methods) and register the next update (*RegisterForSingleUpdate* engine method). Each one of these components will be described in detail in the next sections.

![Classifiers Comparison](image)

Figure 5.6: Comparison between performance measures from the two created classifiers.
5.4.1 Game-Enabled System Discussion

There was a discussion on whether the decision tree component should be initialized immediately when starting a new game. The initial part of the "unmodded" version of the game is composed of a forced tutorial scenario, which is associated with the "Unbound" main story quest, to introduce new players to some game mechanics. Since the training data set data collection was performed in an unrestricted environment, there was the concern that the actions performed during the tutorial in a new game start - which are similar across every player due to the linear nature of the tutorial - could affect the player classification. To prevent that, the SM Event Nodes come into play. These are events that run when a particular action occurred in game.

In this particular case we were interested in the Change Location Event node. This node runs every time the player traverses locations and contains a list of quests that should be activated or changed in some way if certain conditions are met. The DecisionTreeQuest quest was added to this list with the validation to whether the "Unbound" quest was completed and the "DecisionTreeQuest" was already running. This guarantees that the component is initialized only once if the player already completed the "Unbound" quest. This verification covers both scenarios where the player started a new game and completed the tutorial section; and where the player loaded a preexisting save file where he already finished the tutorial.

5.4.2 Decision Tree Component

When the update event is triggered, the OnUpdate function will run. This function is the responsible for selecting a class to the player and running the classifier, assigning him a player type. Firstly, the DecisionTree function is called, gathering and copying the player attributes from the last two hours, through the difference between the attributes gathered in the last classifier execution and the current ones. If this is the first classifier execution for a particular save file, then the previous attributes are assumed to be 0. This way, it is guaranteed that there is no value accumulation between classifier executions, since all these attributes are global counters stored in the game engine.

After the attributes from the two previous hours are collected, an update is preformed, assigning the current attribute values to the previous ones. This way, every time the algorithm runs, the previous attributes gathered will be the ones used by the previous algorithm execution. For example, at the beginning of the DecisionTree function, the variable dungeonsCleared will be loaded with the content of the global game variable "Dungeons Cleared" (through the engine function Game.QueryStat) minus the prevDungeonsCleared value, which will be 0 if the classifier have not been executed yet. After that the dungeonsCleared value is ready to be used by the classifier. After that, the prevDungeonsCleared value will be update with the dungeonsCleared value, so that it can be used in the next algorithm execution, two hours from there.

An alternative way to get the current values from the previous two hours was to set the real global value to 0 after the classifier execution. However, this would break other game quests that use the real value from some of these variables.

After all the variables are collected, the tree algorithm is finally executed. The representation of the decision tree is represented in Papyrus by encapsulated if-else sequences, representing the test nodes of the tree. The leaves of the tree are represented by return values that correspond to the player type. The return values are represented as integers with the following correspondence to the player types they represent: 1-Explorer; 2-Killer; 3-Achiever; 4-Socializer. In each node, the classifier will compare a certain attribute and either perform another comparison or return the player type, if it reached a leave. The tree implementation can be seen in detail in Figure 5.7.
5.4.3 Class Manager Component

The ChooseClass method will assign a class to the player based solely on his in-game skills. For the purpose of this work, three classes were designed: the Mage, the Hunter and the Blacksmith. Each class is a representation of a group of several skills. There are 18 in-game skills in TESV:Skyrim, so the objective was to create three classes that grouped most of the skills and represented a "theme" for the quests proposed to the player. This function computes three class weights and decide the player class that corresponds to the highest class weight. The class weights are archeryLevel, blacksmithLevel and mageLevel and they are computed through the average of the correspondent skills. The skill value is obtained with the engine method GetBaseActorValue(attribute) from the player reference with the attribute being the skill in question (e.g. Game.GetPlayer().GetBaseActorValue("Marksman")).

The class weight is the average of a group of skills. These are as follow: Hunter Class Weight - Marksman, Sneak, LightArmor, Pickpocket and Lockpicking; Blacksmith Class Weight - Two-Handed, Smithing, Heavy Armor and Block; Mage Class - Destruction, Conjuration, Alteration, Illusion and Restoration. At the end of the function, an integer value is returned, representing the class that obtained higher weight. The assumed coding is the following: 1 - Hunter, 2 - Blacksmith, 3 - Mage.

5.4.4 Quests Manager Component

The personalized content presented to the player comes in the form of custom quests. These quests were created focusing on each of the available player classes and types. A total of 18 quest structures were created, six for each player class available. The six quest archetypes created were adapted from [65] and are the following:

- Escort Quest
- Bounty Quest
- Messenger Quest
Collect Peacefully Quest
Collect Aggressively Quest
FedEx Quests

Quest Structure

All quests were created in a way such that their starting point, their goal's location and their starting point are dynamically selected by the RadiantAI component. Creating such quest structure was no easy task, but this way we can guarantee that there is a larger diversity in quests and any quest can be initiated in a location that the player is near.

Each quest is unique in its own way, that they all follow a logic sequence of actions composed of three parts - meeting and accepting the quest from the quest giver, doing a set of actions in order to complete the objective proposed by the quest giver and returning to the quest giver. The Bounty Quests behave differently because, instead of a quest giver, the player receives a note of a bounty, i.e. a letter delivered to the player explaining that there is an active bounty. There is another independent part that is not directly part of the quest but is also important, which is the lure part. This part makes it so that a courier will find a player a deliver him a letter with information on a new available quest in a nearby town. This part exists to gently guide the player to a new quest in an immersive way.

Quest Initialization

After the player class and type are decided, the StartQuests function will run, resetting all the custom quests that were initialized but not started by the player or already finished. This guarantees that, even if the player rejected a previous quest proposed to him, a similar quest can still appear in a different place in the future. The instructions inside the StartQuests method reset all the custom quests, making them ready to be started again, should they be chosen by the ChooseQuest function.

The ChooseQuest function is the responsible for selecting a quest to present the player, according to his preferences. From the analysis of the most common RPG quests types and their description [65], an association was made between quest types and player types, i.e. what quest types a player from a specific type was more likely to enjoy. Since no relevant research was found that co-related these two classifications and for the scope of this work, it was decided that an association should be made, directly linking a player type to three quest types, just through an analysis of both the player and quest type definitions given by the authors.

Quest Type Preference per Player Type

The work by Dickey et al. helped create the quest classification used in this work. Because of the single-player nature of TESV:Skyrim, the Goodwill was removed, i.e. there is no other players to help in the virtual world of Skyrim and no quest already present in the game that follows the description for that particular quest type. Furthermore, the Collection was segmented into two different groups: Peacefully Collect and Aggressively Collect.

We argue that there can be a Collect quest where the player character is required to defeat enemies to acquire the required ingredients (as the original Collect Quest by Dickey et al) and other, in a more peaceful fashion, where the player does not have to necessarily get involved in combat, but can just gather the required items from the virtual world. Despite similar in content, the two types are very different and may appeal to different player types. Both of them suit particularly well in TESV:Skyrim and the original version of the game contains some of these quests.
Table 5.3: Quest type preference per player type. The number of plus signs represent the level of interest in a quest type from a particular player type.

<table>
<thead>
<tr>
<th>Quest Type/Player Type</th>
<th>Killer</th>
<th>Socializer</th>
<th>Explorer</th>
<th>Achiever</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounty</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+++</td>
</tr>
<tr>
<td>FedEx</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td></td>
</tr>
<tr>
<td>Collect Peacefully</td>
<td>+</td>
<td>+++</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Collect Aggressively</td>
<td>+++</td>
<td>+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Escort</td>
<td>+</td>
<td>+++</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messenger</td>
<td>++</td>
<td>+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the purpose of this project, the quest distribution per player type was the one assumed in Table 5.3. This means that, for each player type-class combination there are a total of three quest types to select from. The quest selection is made using as input the player class, the player type and a random factor that guarantees that the quest type is not always the same.

In Papyrus, this are a sequence of if-else instructions contemplating all the possible combinations. Since there are three quest types for each player type-class combination (as described in Table 5.3), a probabilistic rule was implemented to confer different probability of selection for a quest type. For each player type-class combination, there is a distribution of 50%, 35% and 15% between the three quest types that the player is progressively more likely to enjoy. An higher chance represent a stronger likelihood of the player enjoying that particular quest type according to their description, and are represented by a larger number of plus signs in the Table 5.3.

**Quest Selection Example**

As an example, let's consider a classifier execution that returned a "Killer" player type and a "Mage" player class. This case will only be true for the if(playerClass == 3 && playerType == 2) case in the ChooseQuest method. For this specific instance, the system will select between the "MageCollectAggressiveQuest", "MageBountyQuest" and "MageEscortQuest" quests, with 50%, 35% and 15% probabilities, respectively.

**5.4.5 City Lure Component**

Once the system selected the appropriate quest, a city lure component is executed. This component materializes itself in the method StartCityLure, which is called at the end of the Choose Quest component. The purpose of this component is to gently guide the player to the chosen quest in an immersive way. Without this component, the player would not know whether a new custom mission is available or not. By using a courier NPC, the player can be informed that someone in a certain city wants to propose a quest to the player character. This courier system is already used in the vanilla game to deliver letters and notes to the player character as part of a quest or to inform the player that a certain quest is available, so this system is used in the same way in this work to guarantee immersion and familiarity to the player. Immersion wise, it is preferable to inform the player of something that is happening in game with an NPC delivering a letter than a pop-up notification appearing on the screen.

The StartCityLure function perceives what quest was selected by the Choose Quest component and initializes the vanilla courier script, ordering the courier NPC to track the player and deliver him a letter. The contents of the letter are defined by the type of custom quest previously selected by the system and the location of the player. Letters are divided into four major groups: Delivery Letters, Collection Letters, Escort Letters and Bounty Letters. Delivery letters provide a structure for the FedEx and Messenger quest types; Collection letters provide a structure for both the Collect Peacefully and
Aggressively Collection quest types; Escort letters provide a structure for the Escort quest type; and the Bounty letters provide a structure for the Bounty quest type. The structures are divided according to the NPC who is responsible for beginning the quest. All letters excluding the bounty ones inform the player that, because of his reputation, a NPC is interested in his skills and wants to propose him a quest (Figure 5.8). The bounty letters are a particular case, since this quest type does not have a quest giver. Instead, the letter itself starts a quest, asking the player character to kill someone or something, simulating a note that is not necessarily targeted to the player, but other Skyrim residents and adventures as well (Figure 5.9).

The content of the letters is personalized at the level of the location that these refer to. When the `StartCityLure` method is called, besides choosing what type of letter will be delivered to the player, the system also identifies the player character location in the game as well. Every interior and exterior space in Skyrim is treated by the game engine as a cell. This includes all the dungeons, cities, open-world space, houses, etc. The virtual world of Skyrim is divided into nine holds. Each hold have a region capital represented by a settlement or city. While the majority of the cells have their parent region set as their hold, some do not. This prevents to system to 100% identify the correct hold where the player character is when the classifier runs.

Once the player character location and letter model are selected, the loose components (also known as Aliases in the Creation Engine) of the letter are filled by the Radiant AI system. These refer to the location where the player character is and, consequentially, where the quest giver is. Let us consider that the player character is inside a dungeon located in the Windhelm Hold when the classifier runs and that the Blacksmith Messenger quest was the one selected by the Quest Selector component. In this particular scenario, the Delivery Quest letter model is the selected one and the location alias will be filled with the WindhelmHoldLocation location, creating the letter seen in Figure 5.8.

### 5.4.6 Custom Quests

There is a large diversity of quests that can be presented to the player according to his preferences, i.e. player type and class. All the quests selected by the Quest Selector component were created specifically for the purpose of this work. There are a total of 18 custom quests dispersed between three classes. Each class has a quest for each one of the six quest types. These quests were created using the interface provided by Creation Kit and some Papyrus scripting. The way that quests relate with a class is through their plot and objectives, i.e. a quest belonging to the hunter class will have the player using his hunting skills and/or helping other hunters.

The Creation Kit tool provides a window just for the purpose of quest creation. The majority of the
Figure 5.10: Quest creation window in Creation Kit.

Table 5.4: Created quest structures.

Quest Types

Designing a quest can be a cumbersome task because of all the components involved. For the purpose of this work, 18 different quests were created, as represented in Table 5.4. Each quest has a different plot but follow the same structure for the same quest type. The plot is also themed by the player class.

In the bounty quests, the player character receives a letter indicating that there is a bounty on a creature, animal or enemy NPC, and a great rewards awaits him, should he complete the bounty. In this scenario, the player must kill the entity that have an active bounty and report to any Government Steward. The escort quests require the player character to escort an NPC to another town of the player choosing. The NPC will ask the player character company since they share the same class. In the messenger quests, an NPC asks the player character for help to deliver a message to another NPC in any part of Skyrim. These particular quest type usually involves exploration and random encounters on the road to the objective, just like the FedEx quest type. This quest type is similar to the messenger one but requires the player to deliver, fetch or trade items with the NPCs. Finally, the collect quests require the player character to collect a certain number of items. The difference between peaceful and aggressive

---

Footnote:
In the world of Skyrim, there is an NPC in each one of the nine holds, responsible for the day-to-day running of the hold. These have special dialogue options and function like a Jarl servant.
collection quests reside in the fact that the aggressive one requiring the player to kill creatures or enemy NPCs to get the items, while the items asked in a peaceful quest type can be obtained in the game world without a fight.

All the quests have some similarities between them. All of them reward the player character with money or other useful items and experience in a skill belonging to the chosen class. This way, the player is encouraged to play according to his preferences and be rewarded by doing so. Due to the City Lure Component all quests have a letter delivered to the player character, explaining that, because of his reputation in a certain skill, an NPC in a town (usually the nearest town) is interest in proposing him a challenge. After interacting with the quest giver, the player can decide if he wants to accept the quest or reject it. Another common feature is the dynamic characters and objectives location selection. We made use of the Radiant AI system to select those. In all the quests there are some components that are filled this way. This very large quantity of possible combinations causes an increase in quest diversity.

**Quest Example**

Let us take a look at the ArcheryMessengerQuest, which is one of the simplest quests and part of the hunter class quests. When this quest is selected by the Quest Selector component, it automatically sets this stage state to 0 and runs the City Lure component. The City Lure component will check the player character actual location and dispatch a courier with a letter. The quest giver for this particular quest is an innkeeper. Using the location of player character, the Radiant AI system will assign the innkeeper from the nearest town (i.e. in the same hold where the player character is) as the quest giver. The letter will state that the innkeeper from that particular hold heard about the hunting skills of the player character and might have a mission for the player character (similar to the one present in Figure 5.8).

In the dialogue section of the quest creation, some dialogue branches were defined. These have the condition to only be used by the quest giver, which was the filled "Innkeeper" alias when this quest was initialized. This makes it so that only that particular innkeeper can deliver those dialogue lines. When the player reaches the inn, the innkeeper will engage in conversation with the player, explaining that he wants the player to check on a hunter in the region and get back with the news (Figure 5.11). This behavior is caused by the ForceGreet AI package added to all the innkeepers in the game. This AI package orders the NPC to deliver a predefined dialogue line when the stage for this particular quest is set to 0. It is also important to note that the innkeepers will continue to behave like they would in an unaltered version of the game.

![Figure 5.11: Example of a quest narrative.](image1)

![Figure 5.12: Other example of a quest narrative.](image2)

If the player accepts the quest, the quest stage will be set to 10. There is an objective defined for this stage, which points to the "Hunter" alias, which was filled by the Radiant AI with a reference to a random
hunter in the region that the player character was in when the quest was initialized. At this point, the objective “Check on the hunter” is displayed inside the game quest list and as a marker in the compass and map.

When the player finds the hunter and talks to him, he tells the player that everything is fine with him and he have been busy hunting (Figure 5.12). After that, the quest stage is changed to 20. Again, all the dialogue, dialogue conditions and post-dialogue behavior was specified in the dialogue tab. A new objective is now displayed, since it is the objective associated with stage 20. This objective points to the “Innkeeper” alias, which was the quest giver. The new objective “Return to with the news to the innkeeper” is now displayed in the quest list and as a marker on the game map and compass.

When the player character returns to the innkeeper and tells him the whereabouts of the hunter, he will reward him with some gold coins and the player will gain experience in the archery skill (as this was a hunting-related quest). After the player character receives the rewards, the quest stage is set to 40, its finishing stage.
Chapter 6

Tests and Results

The results of this work were evaluated both quantitatively and qualitatively. In this chapter we make a description and analysis of the methodologies and obtained results, in order to understand how the solution performed and what was the general feedback regarding it. First, we describe and discuss the results obtained in the player demography online survey, which was conducted to perform a demographic analysis on the TESV:Skyrim player community and understand the level of interest in IS and PM systems. Secondly, we detail the experiment conducted over a testing group in order to evaluate the created player model and quality of the implemented solution. Lastly, we assess the community reception and feedback over the released game modification containing the implemented solution, performing a critical analysis over the popularity of the mod and an online questionnaire results related to the mod and PM systems in RPGs.

6.1 Player Demography Survey

In order to understand how is the active player community composed and to assess its interest in IS and PM systems, an online survey was conducted and published on the social media website facebook.com and the online discussion forum reddit.com, in a TESV:Skyrim related sub-forum /r/skyrim. This survey was composed of 26 questions distributed into four sections - Demographic Information, Gameplaying Habits, Skill Set and Player Categorization - and the summary of the results can be seen in appendix A of this document. The survey obtained 2746 answers and it is important to state that the majority of the respondents are experienced ones in TESV:Skyrim since those are the ones that actively frequent game-related forums, from where the major part of the obtained answers came.

From the demographic section of this survey, it can be concluded that, from all the players, approximately 85% are male, 66.7% are between 15 and 24 years old and 22.9% between 25 and 35 years old, while 6.9% are over 35 years old and 3.5% less than 18 years old.

In the next section - gameplaying habits - it is clear that the respondents are very experienced in the game, with 67.8% of them answering that have played more than than 200 hours, against only 0.6% who have played less than 20 hours. Furthermore, only 9.8% of the surveyed played on lower difficulties, against 42.4% playing in above normal difficulties and 47.8% on normal difficulty. The most used playing platform is the Personal Computer (PC) (61.8%) and from those, 64.8% use mods in their usual play sessions.

From the Skill Set section, 85.1% of the players surveyed answered that they consider that there are some in-game skills that are "essential" and make the game easier and 61.4% think that they felt
penalized by the game when using a particular skill. 85.9% also consider that the skills that they used the most could be used to create more personalized content, which is one of the base motivations of this work.

In the last section - player categorization - consists of questions assessing the interests of the respondents in different game features and a Robin Laws player type selection. It can be concluded that the majority of those questioned (63.4%) often role-play and give their characters a backstory and a specific set of skills to use throughout the playthrough, playing differently with each character they created (78.2%). The last question was made to assess the possibility of using Laws’ classification. From the answers, it can be concluded that this model could be used since there is an even distribution from the players in the different classifications, with the highest being "Storyteller" (19.3%) and the lowest being "Butt-Kicker" (8.7%). This questionnaire was done before collecting the training data set for the DT, which explains the classification used in the survey.

6.2 Solution Evaluation

6.2.1 Testing Group Evaluation

A test was conducted to validate the used player classification and the implemented solution. It was performed in a testing group composed 30 subjects (24 male, 6 female), where 23 of those were experience players and 7 were non-experienced, i.e. were familiar with the game but had not played it for too long. The objective of this test was to assign the players the correct and "opposed" player type during two separate gameplay instances to understand if the difference in content was noticeable by the subject, which session was preferred and how was it perceived. Before the play sessions each participant was asked to take the Richard Bartle test, which provided us the correct player class, i.e. the one that the player self-classifies himself in.

This test was done in a special environment, with prepared save files for each player type. At the beginning of each play session, the correct or incorrect save file was loaded and the subject was asked to play freely for 1 hour. After each gameplay session, the subject was asked to answer one of a two-parts survey, containing questions about how he felt during and after the playing session. To avoid biased answers, the subjects were not informed about the details of the test objective nor did they know...
which player type save file they were playing with (correct vs incorrect one) for the duration of the test. It is also important to note that the order by which the correct and incorrect versions were presented to the player was random, i.e. some test participants played the save file with the correct player classification first while others played the incorrect one. The conducted test followed the following protocol:

1. **Bartle Test** - The participant is asked to take the Bartle test. The result of this test gives us the player class that we nominate as the correct one.

2. **First Play Session** - A save file is chosen for the player. This could be the correct or incorrect version, i.e. the save file that contains a character with the same player type that the participant obtained in the Bartle test or the opposite one. The participant is then asked to play the game freely for one hour.

3. **Questionnaire Part One** - The participant is asked to fill the corresponding part of the questionnaire in relevance to how he felt during and after the gaming experience.

4. **Second Play Session** - Another save file is loaded. If the correct version was selected in the first play session, the incorrect version is now loaded and vice-versa. The subject is again asked to play for one hour.

5. **Questionnaire Part Two** - The participant is asked to fill the corresponding part of the questionnaire in relevance to how he felt during and after his last gaming experience.

**Environment**

For this test a special environment was created. Since the experience consisted of two separate gameplay sessions of one hour each, the game modification had to be slightly adjusted to behave accordingly. Since the created solution in this work takes a very long in-game time to gather player data, an alternative version was created specifically for the purpose of this test. In this modification, the game will play normally for 10 minutes for the test subject to get used to the game, i.e. controls, environment, and after that, the decision tree algorithm will initialize, selecting the appropriate content - in the form of quests - according to the player attributes.

For this testing scenario we have created four different save files, each one manipulated so that the DT system selects a different player type (i.e. Killer, Achiever, Socializer and Explorer) when it starts. In all these save files the player starts in the same in-game place - which is relatively close to a city - and is playing as a mage. All save files were also adapted in difficulty to prevent a frustrating experience by unexperienced players.

**Questionnaire**

The questionnaire presented between play sessions was one that followed the GEO framework [66]. This framework allows the capturing of game experience based on a number of items such as positive affect, competence, immersion, flow, challenge, etc. In this questionnaire, a set of questions represented by a number are presented to the respondent assessing his thoughts and feelings using a five point scale, from 0 to 4, where 0 is "Not at all" and 4 is "Extremely". Despite looking like a regular survey, its numbered structure makes it possible to identify the overall score of a participant in one of several topics (e.g. flow, annoyance, competence) by calculating the average of a set of answers. For the purpose of this work we decided to use the core questionnaire and post-game modules of the GEO questionnaire, since the social presence module was not applicable. The core questionnaire module was composed of 33 questions and the post-game module composed of 17.
In this section we will perform an analysis on the obtained results from the questionnaire performed by each test subject. This will validate the used player classification and overall experience provided by the added content to the game. The results cover all the areas assessed by the GEQ test. However, the ones that we were more interested in were Immersion, Flow and Positive Effect during gameplay and Positive Experience post-gameplay. This was because we wanted to understand how the testers perceived the experience and if, in general, they preferred the experience with the correct player assigned or if it was indifferent to them. The statistic metrics used are composed of the mean, median and standard deviation of scores obtain from the participants in the different GEQ areas. Due to the nature of the tests, it was also possible to perform a Wilcoxon signed-rank test [67] to understand the levels of significance between the results provided by the two classifications, given by a *p*-value. A side-by-side comparison allows as to draw some conclusions. A summary of the mean, median, standard deviation for each GEQ area and the significance between the results can be seen in Table 6.1.

From a general analysis, it is clear that the participants preferred the correct player classification over an incorrect one. Looking at the Table 6.1 the mean values for the Competence, Immersion, Flow, Positive Effect and Positive Experience areas for the correct player classification are higher than its counterpart. More over, the values in the areas of Annoyance, Challenge, Negative Effect and Negative experience are higher when the participants played with the incorrect classification save file.

Starting with Immersion, the players consider themselves slightly more immersed in the narrative and experience when playing with the appropriate player type assigned (Figure 6.2). This can be explained...
by the fact that, with a different classification, the content presented to the player is not as interesting
and engaging than one that fits his cognitive profile, thus creating a shallower experience that the player
do not associate with so well.

Regarding Flow, the testers also preferred the experience with the correct player type assigned.
However, as seen in Figure 6.2, the difference is not as evident as the other ones. The significance level

Figure 6.2: GEQ areas score comparison.
in Table 6.1 also reveals that this comparison is the least relevant of the four. Despite the fact that this area obtained an average score, it is not by any means an irrelevant result. One possible explanation is that the testes perceived the generated content as well integrated in the game and behaving like base game content. That being the case it is a good result, since one of the objectives of this work was to provide well integrated content that blended well with the rest of the game, behaving like it was created with the game itself.

A deeper analysis on Positive Effect reveals some interesting conclusions. Even though getting an above-average experience overall during gameplay, the testers considered that they got more enjoyment when correctly identified as their player type. Since this was one of the hypothesis from this work, it is a very significant result. Furthermore, the Wilcoxon signed-rank test performed over this area proves that there is statistically significant in the results (Table 6.1). It is safe to assume that, during gameplay, there is a higher enjoyment level if the content presented matches the preferences and play style of the player. Furthermore, it can also be concluded that the used player model is well adjusted and there is a strong link between the player model and the content that the decision tree chooses for the player.

Lastly, the Positive Experience values obtained from the GEQ corroborates the hypothesis that the model is indeed well fitted. There is an approximate 41% mean increase when comparing the obtained results between the incorrect and correct player classifications (Figure 6.2). As the questions for this particular score corresponded to the post-game module from the questionnaire, the interpretation made was that the test participants stayed with a higher positive impression after playing the game with the save file containing the correct player classification. In addition, the significance level obtained and displayed in Table 6.1 supports the fact that difference from the results obtained from the two play sessions is relevant.

### 6.2.2 Community Mod Feedback and Survey Results

To assess the TESV:Skyrim player community, the “Your Own Skyrim” game modification was published in the nexusmods.com modding content website and the digital distribution platform Steam, in the Steam Workshop. The NexusMods website allows users to upload and download modifications for computer games across five hundred video games, acting as a source of original content distribution. NexusMods is highly noted for its support of the game The Elder Scrolls V: Skyrim and is often regarded as the largest website supporting modifications for games in The Elder Scrolls series of games. Steam Workshop is a Steam account-based hosting service for video game user-created content. This Steam integrated system allows the player to browse and quickly install game specific content and modifications. As of May 2018, Nexus Mods contained over 57 thousand different uploaded items and Steam Workshop almost 29 thousand. In NexusMods alone there have been more than 1.4 billion downloads of TESV:Skyrim content. The mod description contained detailed information about the implemented solution and a request for those interested to fill a questionnaire containing questions about the mod itself and general opinion on PM systems in RPGs. A topic was also started in the Skyrim mods specific subgroup of the online discussion forum reddit.com to announce the mod launch. Dozens of people replied on the topic engaging on discussions, giving opinions and feedback and asking specific detail questions. On both NexusMods and Steam Workshop the “Your Own Skyrim” mod exceeded the expectations. The feedback and reception by the player community was very gratifying, with over 10 thousand mod downloads as of the beginning of May 2018 and a high level of interest on the issues addressed in this work.

Let us examine the NexusMods domain first. The game modification was published in the beginning of April 2018 and quickly captured the interest of many. Over the two next weeks after being published it was in the top 10 most popular Skyrim Mods of the week and top 10 trending Skyrim mods, reaching the
Figure 6.3: "Your Own Skyrim" in the top weekly and monthly mods from the TESV:Skyrim Steam Workshop.

16th place in the most popular mods of April 2018. As of May 2018, 2581 downloads were performed on "Your Own Skyrim" mod, a community member offered his help to translate the mod to Spanish and a mod developer asked permission to use mod in 3 very popular mods that he was working on. Furthermore, the discussion section of the mod page was filled with people debating the explored concepts and giving their feedback on the mod.

On Steam Workshop the results were even better. The "Your Own Skyrim" mod was in the top 10 Skyrim Mods of the week for the two following weeks after being published, reaching the first place during the second week for the top Skyrim mods of the week and top Skyrim mods of the month with a five star rating, as seen in Figure 6.3. As of May 2018 the total number of downloads for the mod from the Workshop was 7609.

A total of 447 people answered the questionnaire provided in the mod description. The questionnaire contained 9 questions that aimed to assess the opinion of those who downloaded the mod regarding the mod itself and PM systems in modern RPGs. Let us make an analysis on the obtained results. The population that answered the survey is composed of players that play an average of 2 to 10 hours per week (65,5%), with 20,4% of the surveyed playing more than 10 hours per week an 14,1% less than 2. 76,7% of the respondents have also played TESV:Skyrim form a combined total of more than 200 hours. Regarding the use of mods, 39,4% of the participants usually play with more than 100 mods installed, 36,9% with 20 to 100 mods and only 7,6% usually play with less than 6. As for the mod itself, 38,7% of the surveyed played it for more than 4 hours, 22,6% between 2 to 4 hours and 38,7% played it for 2 or less hours. When asked how many quests they have completed with the mod active, the distribution is even, with 17,8% of the survey participants answering that they have completed no quest using the mod, 15,2% completed one quest, 27,1% completed two, 11,4% three and 28,5% have completed more than three quests.

The last two sets of questionnaire questions are performed using a series of statements for the surveyed to answer in a 5 point Likert scale, with 1 being "Strongly Disagree" and 5 being "Strongly Agree". The first one assess the players opinion and experience with the mod. A summary of some of the given responses is displayed in Figure 6.4.

At the end of the questionnaire, a set of three questions rates the respondents thoughts on PM and its
applications on RPGs in general. The results show a great interest in this type of PM applications. To the statement "I think that a player modeling component (i.e. a system that adapts the game to each player personal taste) should be more frequent in RPGs", two thirds (66.6%) of the surveyed strongly agree and 27.9% agree with it, while 5.6% have a neutral opinion, disagree or strongly disagree. Finally, regarding the statement "I think that TESV:Skyrim should have personalized content for each player, according to their play styles and/or personal tastes", 62.2% of the respondents consider that they strongly agree and 30.4% agree with it, as seen in Figure 6.5.

6.3 Results Overview and Discussion

The results provided by both the testing group and online mod evaluation provide the foundation to validate the hypothesis proposed by this work. From the results obtained by the testing group experience we concluded that the Richard Bartle player classification model is adequate when used by the DT de-
Figure 6.6: Summary of some answers from the mod evaluation online survey.
developed solution. Moreover, we concluded that a suitable player model improves the player engagement and immersion in the experience by 21.5%, as well as providing a 24.2% increase in enjoyment level during and 41% post gameplay, when comparing the mean values obtained for those areas. It can also be concluded that the developed solution is well integrated in the base game structure and the new content included in the modified version of the game fits well in an unmodified game instance, since the tested system maintained the flow and narrative structure of the "vanilla" video game. This was one of the objectives of this work since we did not want the game to feel different in any way, as if the PM system was already implemented in the commercialized version of the video game.

Regarding the "Your Own Skyrim" game modification containing the developed architecture, it got extremely positive feedback by the game community. It is important to state that the respondents ranged from a large variety of players since the mod was released in two platforms - NexusMods and Steam - which cover a large spectrum of players, from the more experienced ones to the less ones. Not only did the players who downloaded and tested the mod praised it but they also demonstrated that PM systems have their place in modern and mainstream RPGs, possibly even in MMORPGs due to the similarities between the two genres. The results show that there is a high demand for this type of systems in the AAA video game industry, which was one question that this work wanted to answer.
Chapter 7

Conclusions and Future Work

7.1 Conclusions

With the constant growth of the gaming community and industry, new ways to captivate the audience have to be studied and developed. Some IS and PM techniques have proved themselves effective when creating engaging and diversified narratives. Modern interactive media, such as video games, should allow the user to be a participant in the story and not just a viewer of the action, crafting the virtual world and narration with his interactions and actions and maximizing player engagement, immersion and enjoyment.

Player Modeling techniques use psychological approaches to create player models so that the interactive media provide the user with personalized content and a tailored experience. In this work we proposed a PM architecture that is capable of examine the player actions during a specified period of time and assign him a corresponding player type through machine learning techniques, impressively presenting the player dynamically generated content that should improve his overall enjoyment and engagement.

The Elder Scrolls V: Skyrim was the chosen platform to develop a solution implementation because of its open-world role-playing nature, its unquestionable success and popularity, and the accessibility to game modification tools - the Creation Kit. The Richard Bartle player classification was the one used for this particular implementation but other player models were also considered. As for the player type assignment algorithm, a decision tree was used, which was generated using the C4.5 decision tree generation algorithm over a training set composed of 25 records. The content presented to the player according to his classification was developed as a large variety of in-game quests which have some elements, such as objectives, locations and quest givers, that are generated during runtime with a game AI system - Radiant AI.

The developed solution was released to the public as a game modification - the "Your Own Skyrim - Decision Tree Classifier" mod - which was subject to appreciation by a selected group of players that formed a testing group and the game community. From the results of the tests conducted with the testing group it was concluded that the chosen player classification model is appropriate and that there was a significant increase in player enjoyment and imaginative immersion when the players were assigned a save file with the correct player type versus an incorrect one. From the game community questionnaire it was concluded there was high approval rate, with 87.2% of those surveyed responding that they had enjoyed the mod, 74.4% agreeing that presented content was determined by their in-game actions and 71.6% concurring that it matched their preferences an play style. Furthermore, it was also concluded that there is a high demand for player modeling systems in modern role-playing video games, with an...
impressive 94.6% of the respondents agreeing that PM components should be more frequent in RPGs.

The outcome of this work and the obtained results are indeed very promising. We managed to develop a solid PM architecture based on a valid classification model and using a ML technique that can be replicated in commercial game versions. The outcome of the conducted tests on the final solution allows us to support the hypothesis that this type of systems should be invested on during game development by the AAA video game industry in order to create more immersive and believable virtual worlds, maximize enjoyment and an overall better experience for the player, thus adding engagement and replay value to the video game.

7.2 Future Work

There is always room for improvement. We consider that there are some aspects of the solution that could be improved in a future work. The first point to address is the fact that the training data set used by the tree generation algorithm is relatively small. The set was composed of 25 entries and because of that, we could suffer from over-fitting problems. We do not consider that this happened, but we understand that a larger sample would, of course, provide even better results. Because of the size of the data set, with a larger number of labels in the tree, it could be problematic. Which brings us to the next point.

Even though the Richard Bartle model served well its purpose, we felt that we could explore different player classifications should we had a larger sample. More recent player models contain more diverse player classifications. It would be interesting to see the architecture working on a different player classification.

For the purpose of this work only quests were created as customized content. Of course this is not the only possibility of game substance that can be created or selected. The possibilities are endless, from a different story line, to NPC dialogue and interactions, random wilderness encounters, and so on. Our choice is based on the fact that quests, despite having some random aspects, are easier to test and track and are much more noticeable than the aforementioned features.

Lastly, due to time limitations, it was only possible to develop the quests for three character classes - hunter, blacksmith and mage. In a future work it would be interesting to see new, more specific character classes and a larger variety of quest structures. This point was addressed several times in the discussion sections of the mod pages and online forums. Some community members inclusively offered their help with ideas for new character classes.

During the data collection phase, a research was performed as to whether there was a connection between the two player classifications and if it was possible to implicitly convert the player types from Laws’ classification into Bartle’s and vice-versa. No fruitful conclusion resulted from this study as no clear relation was found between the two classifications and there was a divergence in the selection of player types from one classification into the other, i.e. players who self-classified themselves as a certain player type in a classification model did not followed a pattern when self-classifying in the other classification model. This could be explained by the small sample size that composed the training set. With a larger set and a ML algorithm maybe some sort of relation could be established between the two classifications and it would be an interesting study.
Bibliography


(52) S. Bjork and J. Holopainen, “Patterns in game design (game development series).” 2004.


(65) M. D. Dickey, Aesthetics and design for game-based learning. Routledge, 2015.


Appendix A

The Elder Scrolls V: Skyrim Player Classification and Demography Survey and Results
Are you interested in achievements?

Do you usually make your characters join all factions/guilds or just some?

Do you think the skills that you use more regularly should be used to create a more personalized character? (e.g. If you used alchemy too much, create some alchemist-related items related to alchemy would pop-up)

Set of skills - do you try to maximize all the skills available (jacks of all trades) - or do you try to maximize only a unique build for your interests - improving only a few skills?
Would you agree that your characters decisions in-game reflect your

How interested are you in the lore of the game?

Do your characters usually collect rare/special items when playing?

Do you prefer a more stealthy or guns blazing approach?
Google Forms

What do you think is important for you when playing Skyrim?

Which of the following categories do you think describes you better?

Do you re-play when playing Skyrim if you have given your characters a background, personality, etc.?
Appendix B

J48 algorithm results comparison between Robin D. Laws and Richard Bartle player types as labels
1 - J48 algorithm results with Robin D. Laws player categorization as labels
Log:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: Decision Tree - Dataset - Sheet1-weka.filters.unsupervised.attribute.Remove-R1,6
Instances: 25
Attributes: 17
  RPG Player Type
  Locations Discovered
  Dungeons Cleared
  Chests Looted
  Books Read
  Speech Tests (*)
  Main Quests Completed (**) Side Quests Completed
  People Killed
  Animals Killed
  Other Creatures Killed (***)
  Sneak Attacks
  Magic Learned
  Items Crafted
  Ingredients Harvested
  Items Stolen
  Assault Crimes
Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

------------------
Chests Looted <= 17
| Dungeons Cleared <= 2: Method Actor (4.0/1.0)
| Dungeons Cleared > 2
| | Side Quests Completed <= 2: Tactician (5.0/1.0)
| | Side Quests Completed > 2: Power Gamer (3.0)
Chests Looted > 17
| Dungeons Cleared <= 3
| | People Killed <= 26: Casual Gamer (5.0/1.0)
| | People Killed > 26: Storyteller (5.0/1.0)
| Dungeons Cleared > 3: Method Actor (3.0/2.0)
Number of Leaves : 6
Size of the tree : 11

Time taken to build model: 0.03 seconds

=== Evaluation on training set ===
Time taken to test model on training data: 0 seconds

=== Summary ===
Correctly Classified Instances          19               76      %
Incorrectly Classified Instances         6               24      %
Kappa statistic                          0.7087
Mean absolute error                      0.1107
Root mean squared error                  0.2352
Relative absolute error                 40.0156 %
Root relative squared error             63.2848 %
Total Number of Instances               25

=== Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
<th>PRC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casual Gamer</td>
<td>0.800</td>
<td>0.050</td>
<td>0.800</td>
<td>0.800</td>
<td>0.750</td>
<td>0.950</td>
<td>0.740</td>
<td></td>
</tr>
<tr>
<td>Method Actor</td>
<td>1.000</td>
<td>0.143</td>
<td>0.571</td>
<td>1.000</td>
<td>0.727</td>
<td>0.700</td>
<td>0.958</td>
<td>0.705</td>
</tr>
<tr>
<td>Specialist</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.826</td>
<td>0.290</td>
<td></td>
</tr>
<tr>
<td>Power Gamer</td>
<td>0.750</td>
<td>0.000</td>
<td>1.000</td>
<td>0.750</td>
<td>0.857</td>
<td>0.846</td>
<td>0.976</td>
<td>0.875</td>
</tr>
<tr>
<td>Tactician</td>
<td>0.800</td>
<td>0.050</td>
<td>0.800</td>
<td>0.800</td>
<td>0.800</td>
<td>0.750</td>
<td>0.960</td>
<td>0.765</td>
</tr>
<tr>
<td>Storyteller</td>
<td>1.000</td>
<td>0.048</td>
<td>0.800</td>
<td>1.000</td>
<td>0.889</td>
<td>0.873</td>
<td>0.976</td>
<td>0.800</td>
</tr>
</tbody>
</table>

Weighted Avg.    0.760    0.050    0.699    0.760    0.716    0.687    0.947    0.717

=== Confusion Matrix ===

a b c d e f  <-- classified as
4 0 0 0 0 1 | a = Casual Gamer
0 4 0 0 0 0 | b = Method Actor
1 2 0 0 0 0 | c = Specialist
0 0 0 3 1 0 | d = Power Gamer
0 1 0 0 4 0 | e = Tactician
0 0 0 0 0 4 | f = Storyteller
2 - J48 algorithm results with Richard Bartle player categorization as labels
Logs:

=== Run information ===

Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: Decision Tree - Dataset 2 - Sheet1-weka.filters.unsupervised.attribute.Remove-R1,6
Instances: 25
Attributes: 17
  RPG Player Type
  Locations Discovered
  Dungeons Cleared
  Chests Looted
  Books Read
  Speech Tests (*)
  Main Quests Completed (**) 
  Side Quests Completed
  People Killed
  Animals Killed
  Other Creatures Killed (***)
  Sneak Attacks
  Magic Learned
  Items Crafted
  Ingredients Harvested
  Items Stolen
  Assault Crimes

Test mode: evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree

------------------
Locations Discovered <= 15
  | Items Stolen <= 13
  |  | Chests Looted <= 20
  |  |  | Assault Crimes <= 2
  |  |  |  | Assault Crimes <= 0: Explorer (4.0/1.0)
  |  |  |  | Assault Crimes > 0: Killer (2.0)
  |  |  | Assault Crimes > 2: Explorer (5.0)
  |  | Chests Looted > 20
  |  |  | Magic Learned <= 0: Killer (3.0)
  |  |  | Magic Learned > 0: Achiever (2.0/1.0)
Number of Leaves : 7

Size of the tree : 13

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          22               88      %
Incorrectly Classified Instances         3               12      %
Kappa statistic                          0.837
Mean absolute error                      0.08
Root mean squared error                  0.2
Relative absolute error                 21.6822 %
Root relative squared error             46.6168 %
Total Number of Instances               25

=== Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>MCC</th>
<th>ROC Area</th>
<th>PRC Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achiever</td>
<td>1.000</td>
<td>0.053</td>
<td>0.857</td>
<td>1.000</td>
<td>0.923</td>
<td>0.901</td>
<td>0.996</td>
<td>0.976</td>
</tr>
<tr>
<td>Explorer</td>
<td>1.000</td>
<td>0.059</td>
<td>0.889</td>
<td>1.000</td>
<td>0.941</td>
<td>0.915</td>
<td>0.989</td>
<td>0.958</td>
</tr>
<tr>
<td>Killer</td>
<td>0.714</td>
<td>0.000</td>
<td>1.000</td>
<td>0.714</td>
<td>0.833</td>
<td>0.802</td>
<td>0.952</td>
<td>0.868</td>
</tr>
<tr>
<td>Socializer</td>
<td>0.750</td>
<td>0.048</td>
<td>0.750</td>
<td>0.750</td>
<td>0.750</td>
<td>0.702</td>
<td>0.964</td>
<td>0.729</td>
</tr>
</tbody>
</table>

Weighted Avg. 0.880 0.039 0.890 0.880 0.876 0.846 0.976 0.901

== Confusion Matrix ==

a b c d  <- classified as
6 0 0 0 | a = Achiever
0 8 0 0 | b = Explorer
0 1 5 1 | c = Killer
1 0 0 3 | d = Socializer
Appendix C

Training Datasets
<table>
<thead>
<tr>
<th>Player Name</th>
<th>RPG Player Type</th>
<th>Locations Discovered</th>
<th>Dungeons Cleared</th>
<th>Chests Looted</th>
<th>Speech Tests (*)</th>
<th>Main Quests Completed (**)</th>
<th>Side Quests Completed</th>
<th>People Killed</th>
<th>Animals Killed</th>
<th>Other Creatures Killed (***</th>
<th>Sneak Attacks</th>
<th>Magic Learned</th>
<th>Items Crafted</th>
<th>Ingredients Harvested</th>
<th>Items Stolen</th>
<th>Assault Crimes</th>
<th>Theft Claims</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joaquim Quadrado</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Manuel Guimarães</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>António Machado</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Andreia Antunes</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Pedro Saldanha</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Daniel Ramalho</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Francisco Silva</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Tiago Cardoso</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Catarina Vieira</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Renan Mauch</td>
<td>Casual Gamer</td>
<td>13</td>
<td>6</td>
<td>16</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>27</td>
<td>20</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

(*) - Speech Tests include Bribes, Intimidation and Persuasion
(**) - Main Quests include the main story line as well as other factions mission (i.e. Companions, Dawnguard, Daedric, etc...)
(***) - Other Creatures cover all the creatures that are neither people nor animals (i.e. Trolls, Dragons, Undead, Dwemmer, etc...)
<table>
<thead>
<tr>
<th>Player Name</th>
<th>RPG Player Type</th>
<th>Locations Discovered</th>
<th>Dungeons Cleared</th>
<th>Chests Looted</th>
<th>Skills Increased</th>
<th>Books Read</th>
<th>Speech Tests (*)</th>
<th>Main Quests Completed (**)</th>
<th>Side Quests Completed</th>
<th>People Killed</th>
<th>Other Creatures Killed (***)</th>
<th>Sneak Attacks</th>
<th>Magic Learned</th>
<th>Items Crafted</th>
<th>Ingredients Harvested</th>
<th>Items Stolen</th>
<th>Assault Crimes</th>
<th>Murder</th>
<th>Other Crimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>João Alves</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Ana Paula</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Carolina</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Carla</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Daniel</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Diana</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Fabio</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Fernando</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Ana Isabel</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>João</td>
<td>Explorer</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

**Notes:**
- (**) - Main Quests include the main story line as well as other factions missions (i.e. Companions, Dawnguard, Daedric, etc...)
- (*) - Speech Tests include Bribes, Intimidation and Persuasion
- (***) - Other Creatures include non-people and non-animals (i.e. Trolls, Dragons, Undead, Dwemmer, etc...)