ColLabWoW: development of a decision support add-on for World of Warcraft

Virginia Grande
Abstract

ColLabWoW: development of a decision support add-on for World of Warcraft

Virginia Grande

Social online games can be studied to find a relationship between these virtual worlds and civic activism, in order to find what game attributes can lead to which social and civic outcomes. Social interaction and power structure, among many other characteristics of digital media, are limited by the creators of each virtual world.

A change to the power structure may affect the thinking and behavior of the players involved, in and out of the game. To study this phenomenon, the collaborative decision-making tool ColLab has been integrated in a Massive Multiplayer Online Game, World of Warcraft. The process of the design, implementation and evaluation of the first iteration of this integration is described here.
Acknowledgments

I would like to gratefully acknowledge the enthusiastic and attentive supervision of Mikael Laaksoharju during this project. His suggestions, corrections and stimulating discussions greatly improved my work and broadened my horizons.

Patrick Prax contributed very interesting points of view without which this project would have not been complete. I would also like to mention Roland Bol’s valuable guidance and assistance and Lars Oestreicher’s feedback.

The World of Warcraft add-on developing community have constantly helped and given advice. A special mention to Treeston for not only guiding me from the beginning but also providing lots of new ideas and analyses of alternative approaches for the implementation.

I would also like to thank my family: my parents, Ricardo, Óscar and Marta, who have always encouraged me to pursue my goals and made it possible.

My thanks and appreciation also go to my friends, a fantastic group of people who have aided me from short and long distances. Special thanks to Daniel S. McCain, a continuous support who has helped both my professional and personal growth by, among many other examples, giving me that push to go abroad and boosting for my English level (proof-readings included).
Contents

1 Introduction 4

2 Background and motivation 6
   2.1 Relevant theory ........................................... 6
   2.2 Motivation ................................................... 9
       2.2.1 The tool: model and integration ...................... 10
       2.2.2 The game .................................................. 12
       2.2.3 Goals ....................................................... 13

3 Description of the work performed 14
   3.1 Requirements .............................................. 14
       3.1.1 Characteristics of ColLabWoW ......................... 14
       3.1.2 User studies ............................................. 15
       3.1.3 Add-on use ............................................... 18
   3.2 Design ....................................................... 20
   3.3 Implementation ............................................. 20
       3.3.1 Tools used ............................................... 21
       3.3.2 User Interface .......................................... 22
       3.3.3 Communication protocol ............................... 25
   3.4 Evaluation: testing of the prototype .................... 29

4 Analysis of the solution 31
   4.1 The approach for the design ............................. 31
   4.2 The overall process ....................................... 32
   4.3 Editing stage ............................................... 32
   4.4 Voting stage ................................................. 33
   4.5 Summary stage ............................................... 34
   4.6 Number of simultaneous discussions .................... 35
### Future work

5.1 New design

- 5.1.1 New settings stage
- 5.1.2 New first stage: editing

5.2 New functionality

5.3 Localization/Internationalization

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Future work</td>
<td>36</td>
</tr>
<tr>
<td>New design</td>
<td>36</td>
</tr>
<tr>
<td>New settings stage</td>
<td>36</td>
</tr>
<tr>
<td>New first stage: editing</td>
<td>36</td>
</tr>
<tr>
<td>New functionality</td>
<td>37</td>
</tr>
<tr>
<td>Localization/Internationalization</td>
<td>38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibliography</td>
<td>39</td>
</tr>
</tbody>
</table>
Introduction

Social interaction and power structure, among many other characteristics of digital media, are limited by the creators of each virtual world (VW) [1]. Prax and Laaksoharju [2] argue that, given a particular VW, an attempt to push these limitations would affect its properties. Their research focuses on analyzing how modifying the power structure and the decision-making process in a Massively Multiplayer Online Game (MMOG) would lead to more meaningful interactions, where meaningful refers not only to the enjoyment of the game but also to the social and cognitive skills that may be developed in a VW, in their function as educational tools ([3], [4], [5] and [6]).

MMOGs are, as defined by Steinkuehler, “highly graphical 2- or 3-D video-games played online, allowing individuals, through their self-created digital characters or “avatars,” to interact not only with the gaming software [...] but with other players’ avatars as well”[7].

One of the most popular MMOGs is World of Warcraft (WoW) [8], which currently has millions of users and has served as a model for similar games. Prax and Laaksoharju want to facilitate a more democratic power structure in WoW by integrating in the game a collaborative decision-making tool, ColLab\(^1\), as a platform to share a particular group of players’ opinion on a certain conflict that should be studied by the leader of this group. Prax and Laaksoharju argue that this will provide players with a way to consider issues of power and influence not only in but also outside of the game.

This report explains in detail the first iteration of this integration: which characteristics of ColLab were possible to keep and which had to be dismissed, to what extent it was possible to introduce the tool in the game and the conclusions from the testing stage. The results will serve as a basis for the

---

\(^1\)ColLab can be found and used here: http://interact.it.uu.se/collab
second iteration of the development, where more functionality will be added to the tool. After the integration is done, it will be possible to collect data related to what the discussions are about and how they are being conducted. The data, complemented by interviews and questionnaires, will be analyzed to draw conclusions about the possible effects of the game’s power structure in the players. This stage of the project is, however, beyond of the scope of the present document.
2. Background and motivation

2.1 Relevant theory

The background of this project is mainly based on the work of Prax and Laaksoharju [2], who have studied the decision-making process in MMOGs. These games give players access to worlds that appear endless, a feeling that may be further supported by the release of expansion packs for the original launches. These packs add new content to the existing one, such as extended storylines or new objects, characters and regions that were previously inexplicable or unknown of. In contrast to linear games, where the path a player must follow is already defined, the structure of a MMOG generally allows its players to not limit their gameplay: rather, they are encouraged to spend time on exploring and experiencing the setup of that particular VW.

The implementation of a VW, in a programming language, creates limitations on freedom. Users cannot do something that was not foreseen by the designers of the world (unless there is a bug). These limitations can be studied by comparing them with the constraints imposed by other structures that also influence culture, society and social interaction. As an example, the following allegory illustrates Lessig’s comparison of VW programming code and law [1]:

Alice is drowning in a lake when Bob happens to walk nearby. Between Alice and Bob there is a large patch of grass with a sign on it, which reads “Do not walk on the grass (see figure 2.1)”. Depending on in which world the action is set up, two different situations may arise:

- In real life, Bob may attempt to aid Alice. He will choose whether or not he wants to face the consequences of breaking the law, that is, stepping on the grass. If Bob chooses to disregard the sign, he would
necessarily have to obey certain laws, such as the law of gravity. Bob cannot jump the required distance without stepping on the grass.

- In a video game, the allowed actions a user may take are limited - intentionally or unintentionally - by what the designers thought. Thus, if helping Alice was not part of the design, there will be no possible way to come in her assistance: some possibilities would be that Bob cannot access the grass due to an electrified fence, that he cannot see the lake at all and thus cannot think of helping Alice or that he sees a cliff instead of grass, being limited by the laws of nature in the game.

In this example, the two main differences that Lessig stated may be observed: legitimization of the rules to follow and availability of options. The first one looks at how the rules are set and followed. Laws are decided by representatives chosen by -ideally- democratic means and thus the group being governed agrees to follow them, as they have indirectly chosen them. VW code, on
the other hand, is created by the designers of the VW and its conditions are imposed to the players, who in turn can only decide whether to participate in the VW or not. A second difference becomes apparent as well. For citizens to follow the rules expressed in law, one source of motivation is the threat of punishment for those that do not obey the norms. However, this is not the case with VW code. Lessig states that players have to comply to the conditions that the creators of that code have set for the game, which are imposed through a kind of physics, e.g. the electrified fence mentioned before. These conditions limit the interaction and social architecture (e.g. there are only particular types of groups that players can form). Thus, law motivates to follow its rules, whereas code enforces players to obey them. Lessig’s work may be interpreted as that there is no way of bending the rules imposed by the VW code, a fact especially interesting for this project when it comes to the power structure.

Prax and Laaksoharju argue that these differences are even greater: while law is external to the medium, code is not, as it is what builds that VW. “Code sets all the possibilities for action and interaction” [2]. Consequently, the process of reflection on the structure becomes harder. In the example above, in order to request a change to the structure:

- In real life, Bob sees the sign and has the possibility of asking for an explanation of the rule to the corresponding authority, requesting a change to this law and, in the end, deciding whether or not to follow it.

- In a VW, where stepping on the grass was not a possibility in the mind of the designer, Bob would not be able to walk on it. He would have to consider not only whether or not to he wants to step on the grass but also, and previously, the possibility of having a way to access the lake.

In the second case, Bob may not know what possibilities he has been excluded from: there are not only physical, as Lessig states, but also mental barriers imposed by code [2].

Jean Piaget concluded that the way children view game rules develops to the point where they realize that these rules are set to enjoy the game [9] and, consequently, should be malleable. The same applies to adults, as shown by Kavathatzopoulos and Rigas [10]. Lessig encourages everyone to reflect on this fact, regarding VW, and proclaims that “we should expect — and demand — that it [technology] can be made to reflect any set of values that we think important. The burden should be on the technologists to show us why that demand can’t be met” [1].
The structure of a MMOG often forces players to cooperate (and thus, socialize) in order to complete many quests of the game. There is a relationship between social online games and civic activism that could be studied in order to find what game attributes can lead to which social and civic outcomes [4]. In the particular case of WoW, a player creates a group and automatically becomes the leader of it. Leading involves a huge amount of power, including the possibility to add and expel members, and the only merit to obtain such power is simply being the one that started this group. Even though the general structure of the game is set in this way, players may attempt to bend and modify it by using different strategies. An example is the council model, where there is a group who share the power while the official leader, a position that is imposed by the game, is a voluntarily unused character. These attempts indicate that some players would like to change the power structure of the game and reflect on ways of doing so.

These are approaches to bending the rules of the game towards a more democratic model. As democracy is a term that should be defined due to its multiple (and mostly, vague) definitions, in the present document, democracy is understood in the same view as Laaksoharju and Kavathatzopoulos: “a dialectic process to reach collective decisions that take into account, but not necessarily reproduce, all of the citizens’ interests” [11]. Thus, it is not a system where the majority decides but rather one where all interests are considered for a final decision, made by a leader who represents the group and who could be held responsible for that choice.

2.2 Motivation

As a conclusion of all these arguments, a deliberate modification of the model of the VW would lead to potentially different and more meaningful social interactions. This new democratic system would be an alternative, so it is worth pointing out that choosing whether to use it or not would be another available decision to make.

Prax and Laaksoharju want to study how the behaviour of players may or may not change if a decision-making tool was integrated in the game. This tool would facilitate a more democratic power structure by providing, on the one hand, players with a platform to reflect on different conflictive situations and vote for different courses of actions and, on the other hand, the leader of the group with a summary of the group’s interests. According to the definition of democracy previously given, this summary would only be a source of information for the leader to consider, rather than an automatic execution of the voted options. In other words, the final decision would be
taken and executed by the leader of the group, not by the tool itself.

2.2.1 The tool: model and integration

For the development of this tool, it seems reasonable to consider some existing ones in the field of decision-making. ColLab [12] was chosen as a basis for the development of the project due mainly to one of its characteristics: it shows the user what to consider for the deliberation process but does not make any claims about which decisions are correct. This will allow Prax and Laaksoharju to analyze how the decisions are made rather than how they conform to a specific set of norms.

ColLab has been developed as an online platform for collaborative and democratic decision-making [13]. It is based on the idea that it is important to reflect on who is affected by a decision, or the “stakeholders”, and how the different solutions to a particular conflict may affect these stakeholders’ interests [14]. Participants in a discussion come to the conversation biased, as the outcome may conflict with one or more of their interests. However, these biases could be countered by showing how a decision would affect not only that participant but also all the other stakeholders [15].

ColLab’s main feature for a particular discussion consists of a matrix showing the stakeholders, each of their interests and different courses of actions proposed to solve the conflict, also referred to as “options” or “decisions”. The cells of the matrix are empty and expected to be filled by the participants in the discussion. Each of these cells will be an answer to the question: “How does this decision affect this interest of this stakeholder?” Each option is divided into two columns: one for positive arguments and one for negative. As a distributed multi-user system, ColLab allows participants to edit these arguments and see the changes in real time. A log of these is kept for transparency and prevention of sabotage.

To introduce a similar system in the game, it is necessary to develop a simplified version of ColLab that would be used not by accessing it with a web browser but rather as an integration, accessed in-game. This is because the direct use of ColLab would alter the game experience too much, as it would be necessary to go out of the game to access it and the interaction with the tool would not be fast enough, thus ruining the game immersion. For these reasons, the adaptation should incorporate the characteristics of ColLab related to the decision-making process but adapting them for in-game use.

Some changes to ColLab’s characteristics are necessary in order to integrate the tool in WoW. A first sketch of this adapted design was created by Prax and Laaksoharju (see figure 2.2). These modifications of the original tool
include:

- scaling down the visualization so that it fits the game’s screen in one window rather than in the original setup of tabs. This would allow players to interact with all the information at the same time and thus the setup process will be faster, a quality crucial for in-game use.

- providing a number of preset discussion setups, i.e. preset values for the default stakeholders, interests, arguments and options, depending on the type of discussion. These values would act as a template that participants may be able to edit and are based on Prax’s personal experience from playing WoW.

- encouraging players to vote for one decision rather than just debate over all of them, as in the original version of the tool. The motivation
of this change is, again, to speed up the process for in-game use.

- not storing any information about all the editions done in the cells. Although it would be desirable to have it as a history to restore a previous version of the matrix, it would also make the integration too heavy.

2.2.2 The game

As it is one of the currently most used MMOGs, WoW was chosen for the project. The story is set in a fantasy environment where players complete missions or “quests”. Even though the interaction with other players is optional, it greatly increases the chances of success in the game: the higher the level of a quest, the more probable it is to need to be in a group of players to complete it.

There are mainly three types of groups in WoW [16]:

- Party: up to 5 members, as a temporary alliance.
- Raid: usually 10 or 25 members, as a temporary alliance.
- Guild: a persistent group of characters who regularly play together and who generally prefer a similar gaming style.

Each player has a list of characters and may choose one for each game. This character may join only one guild, which makes the choice for this permanent group something to reflect on. For the same reason, the new guilds may put special effort in attracting potential members, due to the high existing competition in recruiting.

Another important reason to choose WoW was the easiness with which it is possible to add content to the game: in this case, the tool for the integration. These additions to the game are possible thanks to WoW’s Application Programming Interface (API) [17]. This collection of libraries allows developers and third parties to modify the game experience, mainly by changing the User Interface (UI). Each of these packages of modifications are call mods or add-ons, and their use is widespread in WoW. Some of them even become first-party elements of the standard interface, due to the fact that Blizzard has seen the popularity of such utilities [18]. Add-ons “reduce effort, make visible invisible parts of the game, aid players in coordinating with one another, and capture important aspects of a player’s history of play” [19].
2.2.3 Goals

The add-on for this project, temporarily named ColLabWoW, will serve as a tool to introduce democracy in WoW. It will allow players not only to apply social pressure to the leader of the group but also to reflect on the power structure of the game in an easier way: why it is like that and if it may be altered. In the example of the previous section, Bob may still not be able to step on the grass but now he can at least demand a motivation for that restriction and ask if it could be changed.

The usage-related data of this add-on, added to interviews and questionnaires results, will provide Prax and Laaksoharju with information relevant to their research, such as number of conflicts resolved, how long it took to resolve them, which templates were used and which arguments were important for the players. This will be their basis to study what kind of effect game power structures have on players: they argue that, if VWs are seen as learning environments, the possibility of changing the power structure that ColLab-WoW allows may even help players to reflect on real-world structures and to consider how they think power should be distributed in a society. Should the use of ColLabWoW spread, its features could be taken into account for future games’ design, especially if the current one comes from creators simply following the traditional design for MMOGs.
Chapter 3

Description of the work performed

A first version of ColLabWoW was designed and developed, as a basis for the next iteration of the project. The following sections analyze in more detail the different aspects that needed to be considered and show the decision making process regarding the design and implementation. Future iterations of this project will result in adding more functionality and improving the UI, among other changes (see Future work).

3.1 Requirements

Before starting with the implementation of the tool, the requirements and expectations for the integration were collected and studied. In this section some aspects are explained in more detail, such as characteristics that ColLabWoW should have, what kind of game experience WoW provides and when and how use the add-on.

3.1.1 Characteristics of ColLabWoW

ColLab, the tool on which ColLabWoW is based, is currently in a non-final stage: its design is susceptible to change and, thus, the design of ColLabWoW may be adapted to these changes should they suit the needs of the project. At the time of the beginning of this thesis, the following was expected from the add-on:

- Possibility of providing participants with preset values for the different elements of the discussion.
- Possibility for players to modify the preset values as well as the values provided by other players.
• Collection of data related to the use of the add-on.

• Upload of the data mentioned in the previous point to a server run by Prax and Laaksoharju and in a format adequate for its analysis.

As for what to do with the results of the final stage of the usage of ColLabWoW, that is, voting, Prax and Laaksoharju see three possibilities [2]:

1. “The result is sent to the leader who can then decide to follow it or not.”

2. “The result is broadcasted, and the leader can then decide to follow it or to publicly disregard the group’s opinion.”

3. “The add-on enforces the decision of the group; the add-on on the group leader’s computer will do what the outcome of the vote is, in effect disempowering the leader.”

3.1.2 User studies

In order to study how the game was experienced by different types of players, two user studies were performed. The first one was a test of WoW from the perspective of someone new to the MMOG, whereas the second one consisted of an observation of an experienced player.

New player

Blizzard offers new players the opportunity to try the game with no cost by using a free version. This was the version used in this study, so that the limitations imposed by it (e.g. it is not possible to join a guild) were taken into account when developing the project.

This study also helped understand which kinds of players are expected in WoW. It would seem that the small tutorial at the beginning of the game assumes that the players already know what keys are bound to which actions (e.g. “a”, “w”, “s”, “d” to move left, forwards, back and right, respectively). This, among other facts, may indicate an aim for experienced players who already have assumptions about how computer video games, and MMOGs in particular, are played.

Although the main objective in WoW is the completion of quests, it was noticed that there are certain locations in the game at which a player may stop to recover, meet other players or pause for a while. This feature was perfect to allow the use of ColLabWoW, as reflect on a conflict and the arguments for each of its proposed solutions requires some time. The fact
that players may pause completing their quests to use the add-on without having to fear for an attack or any form of punishment greatly supports that ColLabWoW may be part of a regular game.

**Experienced player**

In the second user study it was observed how an experienced player used different game modes, such as completion of quests with other players, known as “raiding”, or battles between teams of players, known as Player VS Player or PVP. The player started more than one game, so it was possible to see the dissimilarities when playing with different characters.

During this session it was possible to see that the interface may depend on the character selected: other than the default interface, the player had different ones depending on if he was playing as a healer 3.1 or a hunter 3.2. Both types of interfaces were customized by the addition of add-ons: the default interface had its action bars customized to allow a quicker access to some actions, whereas the interface for the healer included different mouse buttons set for different spells.

![Figure 3.1: Example of the UI for a healer, including the health status of the group](image)

There is a chat system, where players may type to communicate with each other or see information related to the game’s story. Some channels have default uses (e.g. channel number 2 is for trading). The most common uses of
this system are searching for new members for a group, discussing strategies and starting private conversations and small talk. If a group of players follows a relatively complex strategy, this chat system may be substituted or complemented by a Voice over IP (VoIP) method for faster communication.

It was also noticeable that the use of add-ons that not only display data but also involve interaction with the player (like ColLabWoW) was very fast: the main goal of players is to play, and they see these interactions as something that needs to be solved fast to go back to playing as soon as possible. This is important for the development of ColLabWoW, as we have to study how to convince players to spend time in participating in the deliberation process and to not see it as a waste of time. It also influences how ColLab would be integrated in the game, as the original tool is intended for audiences not aiming for such fast interactions.

Add-ons were launched in two different ways: by using the graphical interface (e.g. small icons attached to the minimap) or by the use of slash commands, that is, special words preceded by a ‘/’ that are typed in the chat box, which indicate a command. These two ways are not incompatible, as they can both be used at the same time.
3.1.3 Add-on use

When to use the add-on

Based on the work of Prax and Laaksoharju and the user studies described above, it was possible to create a list of different situations where players could benefit of the use of ColLabWoW:

- Scheduling raidings within a guild: different members will prefer different times (e.g. parents will want to start when the kids have gone to bed whereas early workers would prefer to begin earlier).

- Raiding strategy discussion, e.g. choosing the best way to complete a quest in group.

- Conflict resolution during raiding: during the user study with an experienced player, another player asked the group to wait for him but he was ignored since the raid was too easy and the group wanted to move forward. This player may have wanted to start a discussion with the group. This particular case may be challenging to fit in ColLabWoW’s settings, as the group would probably not want to stop in the middle of a raid to reflect on that conflict. It could, however, be started after finishing that raid.

- Loot distribution: players may be rewarded for some actions, such as killing a particular monster. Disputes about who gets which reward in the group may arise, as more than one player may apply for the same item. Players may use arguments such as “That object is better for my character’s class” or “I contributed to the group more than her”.

- Performance discussion: situations in which the performance of a player seems to be poor but it may not be, e.g. a healer helps prevent damage rather than healing after the damage is done. The performance of healers is measured by the healing done statistics and preventing damage is not included there, so it may look as the healer’s performance was bad.

- Conflicts in the guild: Alice complains about Bob harassing her. There is a conflict in the group, as Bob is a very good player and expelling him will decrease the performance of the group.

Add-on usage sequence

When one of the situations previously mentioned or a similar one occurs, a player may want to start a discussion. The steps that would be taken
from the creation of the debate until some action is carried through is the following:

- Alice wants to open a new discussion. Therefore, she uses the proper slash command.

- Alice selects the group that will participate in the discussion (party, raid or guild).

- Bob gets a message announcing that Alice wants to open a discussion. Bob may decide whether or not to join:
  - If Bob wants to join, he uses the slash command to join.
  - If Bob ignores the invitation, regular play resumes.

- Alice and Bob see a matrix with preset values for the description of the discussion, stakeholders, interests, options and arguments. All the players involved in the discussion are able to edit these values and add new ones. These data is shared, so
  
  - Alice’s editions are seen by Bob and viceversa.

- Alice is finished editing. She shares her intention of going to the next stage and waits for other players to finish the current one.

- All the players are finished editing. The voting stage is loaded and all the players now see the different arguments for which they can vote.

- Alice and Bob proceed to voting: they reflect on the arguments and select the ones with which they agree (this change to the original design is motivated in the following section).

- Bob finishes voting. He shares his intention of going to the next stage and waits for the rest of the players to finish voting as well.

- All the players are finished voting. The summary stage is loaded and all the players now see the results from the voting.

- Carol, the leader of the group, reflects on the results of the voting and uses them as a source of information about the group’s interests. Then she decides what to do.

- The discussion is closed.
3.2 Design

The design of the integration was based on the sketch by Prax and Laaksoharju (see figure 2.2 on page 11). However, some changes were made, of which the most remarkable one is the voting possibilities. The original idea was to let players vote for a particular course of action, thus selecting the solution to the conflict with which they agreed more. But this would provide the leader of the group with limited information. Why have the players chosen one option and not the others? In order to increase the richness of the results, especially for the one that would take the final decision, the design was altered: by having the possibility of voting for each argument separately, players are allowed to share their opinions in more detail and the leader has a greater source of information on which to make the decision.

Another important change affects the addition of elements to the matrix. While having them added dynamically would be desirable, as proposed in the sketch, this was set aside till the next iteration due to time constraints. Another approach, faster to implement, was used: a concrete number of cells would be always generated empty. Players would be allowed to add their arguments by filling in these empty cells.

For the display of the voting results, or “summary stage”, the second option proposed by Prax and Laaksoharju was chosen, as it was the one who best matched the definition of democracy given in the introduction: a process of dialog where representatives of the group (the leader) will consider the group’s interests (expressed in votes) and will not automatically execute that (by the add-on) but rather reflect on them to take the final decision.

For the graphical aspect of the add-on, the aim was to integrate ColLab in the game as another tool in it, so elements of the general UI of WoW should be considered for ColLabWoW’s visual part.

3.3 Implementation

The technical issues of the project are explained in this section: which tools were used, how the UI was created and the motivation for the choice of the communication protocol. The goal of this text is to motivate the choice or disregard of different approaches. Thus, very technical details not involved in this issue, such as those related directly to the code, are not included here.
3.3.1 Tools used

Community resources

Add-ons are usually created by players that would like to have some particular feature which is not already included in the game. In order to develop this addition to the game experience, these players look for documentation about how to access and use the API and, sometimes, share their piece of software with the rest of the WoW community.

The official documentation of the API [17] explains in detail how to use the functionality it provides. However, this technical approach may discourage potential add-on developers with no background in programming. Probably motivated by this fact, the community of WoW add-on developers has created plenty of sites dedicated to the topic, such as wowinterface.com and wowace.com [20], in which the language used is more adequate to all levels. Aside from these sources, a book [21] complemented by a website [22], is also in high consideration among the members of the community. Its wiki is referenced many times in the community forums, of which famous ones are http://www.mmo-champion.com and the official forum provided by Blizzard [23]. In many of these sites, more experienced add-on developers help the newer ones.

All of these sources of unofficial documentation are in constant change and expansion to update the information for functions and constants of which use is currently unknown by the community. Aside from this material, developers can read, use and learn from the code of other add-ons, easily found throughout the web. One of the most known sites to check for add-ons is www.curse.com [24].

Technology

Add-ons can be developed by using the Lua programming language [25] and WoW’s API. In particular, the version of Lua used has some modifications, such as naming alterations and library additions (e.g. “Wow includes an extra option for string.format() [...] This option is not included in standard Lua 5.1” [21]). However, the most remarkable change is the elimination of important features, such as the In/Out library, which prevents developers from either reading or writing from files in the client’s hard drive. Blizzard only allows to store information between sessions as the “saved variables”: the content of some specified variables is saved in a special file when the game is closed and read when it is started again [21].

The code of add-ons is organized in different files according to a structure dictated by Blizzard: a table of contents, Lua files and XML files, the latter
being optional. The table of contents lets WoW know which files the game needs to load for the add-on to work; the Lua files implement the logic of the add-on and may include the visual structure as well; the XML files define the visual structure of the add-on.

As stated above, the use of XML is optional. It is also discouraged by a great number of members of the community: some of their reasons are that they consider XML to be hard to read and that they find it uncomfortable to “jump” from Lua to XML files instead of simply reading everything in the same file. As the project aims to encourage players to use the add-on and provide feedback on it, it seems reasonable to follow the most common development practices in the community and avoid the use of XML. This avoidance refers to writing XML code: the use of XML templates of the API, such as UIPanelButtonTemplate, is highly recommended and has indeed been used in the development of CoLabWoW.

To aid in the implementation process several add-ons are used, of which the most important are !BugGrabber [26] in combination with BugSack [27] for debugging.

It is worth mentioning that the API is susceptible to change and has, in fact, been modified before: since 2006 some actions cannot be included in the add-ons to be performed automatically, as Blizzard believed this was making the game too easy [19]. Fortunately, these restrictions have not affected the development of the project so far. They would have had if, instead of choosing to have the summary of vote results simply broadcasted, additionally the add-on was to automatically execute the option chose. However, as mentioned before, this approach was disregarded.

### 3.3.2 User Interface

The original goal of the project was to focus on the logic of the procedure rather than on the UI itself. Thus, it seemed reasonable to first look for libraries implemented by other developers that would already provide tools to build a matrix. After a fruitless search, the graphical interface was developed from scratch.

**Using a library**

LibSpreadSheet [28] was recommended by the community for its editable cells and simplicity of use, which seemed to suit the project’s needs. Figure 3.3 shows the result of using LibSpreadSheet to create a very simple matrix representing a discussion. Participants could edit the contents and then vote or, rather, fill in the separated cells that represented votes.
However, this simplified representation created several issues. The most important was that the original structure of the matrix in ColLab had to be adapted to the one LibSpreadSheet allows, what forced into the design an additional column to represent the votes (named “Agree”). Ironically, trying to simplify the interface led to one more complex and harder to use than the original intent, as having the user known what to do and how was not as clear, e.g. as edition and voting were not separated in the process, it was not clearly stated when users were supposed to stop editing and start voting.

Thus, this approach was disregarded and it was decided to develop the UI without using libraries for table or matrix construction.

**New interface**

The UI for the add-on has been developed from scratch, except from some templates provided by the API. To solve the issue of potential simultaneous editing and voting, the discussion process is now divided in three stages: editing, voting and summary. For the initiator of the discussion there is an additional stage where the options are set.

When a player wants to create a new discussion, initial values have to be set. As long as the user is concerned, the only thing to choose is the scope
of the conflict, that is, which group will participate in the discussion (party, raid or guild). Figure 3.4 illustrates how the player is prompted to make this selection.

![Figure 3.4: Settings stage](image)

After the scope is set, a new window is displayed to the creator of the discussion and each of the participants that join the conversation afterwards. This window contains data that serves as a basis for the editing stage. This stage (see figure 3.5) consists of reading the preset text included in the text boxes and, should the participants consider it necessary, edit any of these values. When a player believes there is no more edition to be done, he or she may click the “Proceed to voting” button. This will indicate that this stage is done for that player and is waiting for the other players to be finished. When all the participants are done with the editing, the voting stage will be loaded.

The voting stage looks different than the editing one, as now the interaction differs (see figure 3.6). In this step of the deliberation process, each argument is the text for a button. Participants may click on any button to vote for the argument it contains (or click again to cancel that vote), as shown in figure 3.7. Special attention was put on making these buttons look like the regular ones in the game, as players are already familiar with these elements.
In a similar process as in the previous stage, when the player is done voting a click on the button below (“Send votes”) will take him or her to the next and last stage.

The summary stage’s visualization, shown in figure 3.8, is a modification of the previous one. The arguments that were not chosen are displayed as disabled buttons, as players already recognize these as parts of the UI where no interaction is possible. The arguments that were chosen are again displayed as regular buttons; the difference is that the text now also shows how many votes that particular argument got.

At any moment of all the deliberation process, a window anchored to the discussion frame may be displayed containing a list of the participants. The purpose of this window, included in figures 3.5, 3.6, 3.7 and 3.8, is to provide players with a way to know who has joined the deliberation group. This makes participants’ trust in the system to increase, as intruders should not be able to enter the discussion without being noticed.

### 3.3.3 Communication protocol

As the different participants in a discussion share data (e.g. proposed solutions, names of the stakeholders, etc.), a communication system is needed in
order to correctly synchronize this information between all the participant clients. This is especially important for the editing stage, as the different editions of the arguments may lead to synchronization errors in some situations.

There was an evolution in the selection of a communication protocol. The use of a private channel was studied as a first option. In WoW it is possible to create in the chat private channels protected by password in order to have private message exchange with a selected group of players. This allows developers to create a communication protocol for communication among members of a closed group. For this kind of task, the community suggests developers to use a protocol that follows some guidelines which have been specified by this community [29]. The aim of this suggestion is to encourage the creation of a standard. However, this method would imply using one of the available channels and there is a limited number of them. It will also lead to a channel full with messages that are not meaningful for the player. Although this could be solved, it would be a fairly complex task that seems unnecessary when there are easier approaches to the situation covering the same needs.

To be able to send messages via the chat system but without disturbing users with information irrelevant for them (i.e. showing synchronization data in
the chat window), the API provides some functions that already implement the communication protocol on which the community agrees. These functions are used in the project to send synchronization data, as well as data itself. Once the communication channel was chosen, a second issue was considered: which kind of protocol to use, the main options being distributed and centralized. Distributed involved a high load of message exchange, which could lead to expulsion from the channel (as a high amount of messages may be considered as spam) and difficulties in the synchronization of data, as it would be hard to know which client has the most up to date copy. In contrast, a centralized system allows an easy method to synchronized the data. Although there are some drawbacks, such as potential data loss, the advantages of the use of a centralized protocol led to choosing this type for the communication over a distributed one.

For each discussion, there has to be a designated player whose client will have the master copy of the data. This client will be referred to as the “master” of the discussion and it is the client of the player that starts the debate. In other words, if a player uses the slash command assigned to start a discussion, this action will set that client as the master of that discussion. A prefix is created, based on that player’s character’s name and a random suffix, which is then registered: the client will be listening to messages sent
Figure 3.8: Summary stage

to a channel which has that prefix as ID. At this point, the master sends an invitation to all the other players in the group defined by the scope of the discussion (party, raid or guild): the prefix is broadcasted so that these other clients know that they have to register that prefix in order to join the discussion and also that the sender is the master of the discussion. After the communication is settled, members of the group may choose whether to join the discussion. The clients of the ones that do will ask the master for a first copy of the discussion data. During the rest of the discussion, participants will send their updates to the master, who will in turn broadcast them to the whole group in order to provide all participants with the last confirmed version of the data.

The above protocol is one of the simplest options considered for the project, as it does not take into account the possibility of the master leaving (intentionally or unintentionally). To make the protocol more complete, an initial idea was to save the timestamps of the moments when the players joined the discussion, so that the second one who joined the game would be the master in case of having to name a new one. This is consistent with the selection of the original master, who the first who joined the discussion, that is, the player who started it.

Other options for a protocol were studied but dismissed for their -so far-
unnecessary complexity. As an example, the following one was suggested by Treeston, one active member of the community of add-on developers, as a leader election protocol:

1. When starting/connecting, the add-on broadcasts “Is there a master?”, then waits a defined timeout for any responses.

2. If the add-on that is currently master sees this, it responds. If any add-on that is not currently the master sees this, it responds too.

3. Depending on response received or not:
   (a) Response received:
      i. If a response from a master add-on is received, communicate with the master add-on to get up-to-date info.
      ii. If a response from a non-master add-on is received,
         A. but no response from a master add-on is received, the addon broadcasts “We need to pick a new master”.
         B. The add-ons that already have data now wait a random amount of time, then broadcast “I am now master”. All other add-ons seeing this broadcast communicate with the new master to synchronize data.
   (b) If no response is received, the add-on assumes that it is the only one currently active, broadcasts “I am now master” and starts with fresh data.

The above protocol provides more stable communication than in the chosen protocol, as it provides a more satisfactory way of selecting the master of the discussion, both at the beginning of the discussion and at a potential loss of master. However, it adds too much complexity and the add-on development aims for simple and light software. Nevertheless, it may be worth considering it, at least as a basis, for future iterations of the project.

3.4 Evaluation: testing of the prototype

After the first prototype was implemented and tested, four different users tried it:

• A usability expert

• A high school student
• A gamer, with no experience in WoW
• A gamer, very experienced in WoW

The variety in the testers lead to rich feedback that has been used for the analysis of this iteration. This feedback included suggestions to change both some aspects of the deliberation process itself (setting, editing, voting, summary) and the UI. The latter was slightly confusing to those who were not familiar with WoW and they were not sure of what was expected from them at some points of the discussion. In contrast, the experienced player believed that more information related to the discussion itself, rather than to the use of the tool, should be added. This feedback is contemplated in the following section.
Analysis of the solution

Some aspects of the project could be improved and others should be taken into account as good practices when developing a new version of the tool. The former issues are addressed in this section, as well as the desirable characteristics already included in ColLabWoW.

4.1 The approach for the design

The basis for the design of the tool was the design of ColLab. However, the users of ColLab and of ColLabWoW are different groups: the add-on will be used only by WoW players, whereas ColLab has to be designed for a broader target audience. Also, the research questions that ColLab may help answer are not as specific as the research questions of this project. This fact also involves that the analysis of the data will differ between the two tools. Taken these facts into account, the design of ColLabWoW should be limited by:

- What players consider to be a meaningful discussion
- What research states as a meaningful discussion
- What is possible to implement (technical restrictions)
- What is possible to analyze (data analysis restrictions)

These points have been considered to conduct the analysis of this first iteration of the project.
4.2 The overall process

The add-on was launched by using a slash command. This approach may be comfortable for experienced users, and maybe some of the new, but it also requires players to remember the command. Launch the add-on graphically, e.g. by clicking on an icon, may be considered to solve these issues and also provide users with more than one way to perform the same action, a feature specially useful when one of the two is not working, e.g. another add-on captures the mouse.

As for the discussion itself, its protocol is directing the way to discuss. However, the existence of this protocol is necessary in order to curb rhetoric. It does not benefit certain values or opinions and it provides a structured means to discuss.

4.3 Editing stage

In this stage, participants share their thoughts by filling in the empty boxes of the displayed matrix. Some of these players may provide lengthy arguments that may deter other players from participating in the discussion. To avoid this situation, a limit of 245 has been set to the amount of characters for each text box. As a basis for this limit, the documentation for the sending of messages via the add-on channel has been taken into account: “The combined length of message and prefix can be at most 254 characters (255th is the tab character used to delimit the two). [...] Length above 254 will disconnect you. [30]”. This limit helps preventing that the arguments of more literate players prevail at the same time that makes the implementation lighter. This stage gives participants plenty of freedom, as the arguments do not have preset values in this first version: the players involved in the discussion are encouraged to fill in these fields themselves. This system works with proactive participants that have some previous experience or a natural ability of finding good arguments. However, if the players invited to the conflict resolution cannot think of any content for these empty boxes, ColLabWoW fails in helping the group solve the issue. This fact suggests serves as evidence of the need of the preset values included as an option in the first idea of the design. The inclusion of this type of aid is studied in the Future Work section. All the editions that players may make to the content of the matrix, if performed simultaneously by two or more players at the same cell, would lead to issues with data copies and which one is the most up to date. A system of locks was designed, so when a player started editing a cell, that cell was blocked for the participants not editing it yet. However, the analysis of this system
showed that there was a possibility that this way of editing could lead to making participants impatient and frustrated, as they would expect to be able to edit faster. Note that communicating to all the clients that one of them wants to edit a cell would take some time, because of the messaging involved in the process, and it should be remembered that the users of the add-on are WoW players expecting a quick interaction to be able to go back to play. This showed that a new approach should be studied; it can be read in more detail in the Future Work section. As mentioned before, not having the new elements added dynamically is not the best approach, as participants may want to use less or more than the number provided. This fact should be studied in future iterations of the project.

4.4 Voting stage

As seen in the sketch from the background section (figure 2.2), the original idea was to have users vote for the proposed solutions, each one as a whole. However, this has been changed for ColLabWoW: rather than choosing one option, thus implying agreement with all of the arguments in it, players are now allowed to vote for each argument individually. This feature provides the leader of the group with a much richer input, as it allows to see the shades of meaning of each vote.

As each of the proposed solutions to the conflict are divided in two (positive and negative arguments), it could be discussed whether to allow participants to either select an argument on the positive column or on the negative one but not both, as these pairs of arguments are theoretically the opposite of each other. However, we believe that the less restrictions imposed to the players, the better the results will be, so in the current version of ColLabWoW it is possible to vote for both arguments of each pair. This way, participants that agree with both opinions may not be adding substantial weight to the final voting in terms of number of votes but are indeed provided with a way to show what their opinion is, an opinion that should be of interest for the leader of the group.

At this voting stage, anonymity should be guaranteed. However, this is currently not possible. Although no information about who voted for which argument is displayed and only the number of votes per argument is shown, there is always the possibility that some players may alter the code of their particular copy of ColLabWoW to have that confidential information available. This is possible because, as players have access to the source code of their copies of the add-on and votes are sent via a chat channel, it is always possible to modify the code in order to capture the sender of a particular mes-
sage. This situation suggests that players should not be tricked into believing that their votes are confidential. On the other hand, the better solution would be to find a scheme to guarantee anonymity. Several alternatives have been considered, though currently none of these fulfilled the requirements for a 100% anonymity:

- Alice sends her votes to a random player, Bob, who forwards this again and again until it reaches the master. This method may involve a lot of messaging, which could lead to an expulsion of the channel and a considerable decrease of the speed of the interaction with the add-on while all these messages are processed. Besides, the master needs to know somehow who sends the votes, as it has to check that everyone has voted. If not, a player could vote more than once with the intent of making those votes weigh more (a way of achieving this in a more subtle way would be to send a set of votes slightly different each time instead of just the same set).

- The master generates a set of tokens. Participants are to received them as an alias for their particular votes. The master sends all the tokens to a random player, Alice (the master does not need to keep one for him/herself); Alice keeps one that is assigned to her client and sends the rest to another random player, Bob. This process is repeated until everyone has received a token. As the master does not know who has what token, anonymity is closer than before. However, as it is possible again to alter the code, players could store more information about the tokens than what they are supposed to.

However, it should be studied to what extent it is feasible to guarantee anonymity and if, for the scope of this project, it is worth to add complex algorithms to the code rather than keeping the simplicity needed for a light and fast to use add-on.

4.5 Summary stage

The summary stage is still under design. So far, the UI includes the minimum information that the researchers participating in the project believe necessary. Some more information may be added in the future, though. The summary has two goals that depend on who is viewing the information: the leader of the group or the rest of the participants. For the latter, the summary is a way of being informed of what the group thinks; for the former, besides from that, it is a source of information to weigh and reflect on in order
to make a final decision. In other words, the leader has the responsibility of choosing one of the proposed solutions on the discussion’s matrix and, whatever this choice is, the opinions shared in ColLabWoW should be taken into account. As the whole group has the summary information, it is easy to see the attitude of the leader towards the group’s wishes: leaders still have the power to do whatever they prefer to do (ColLabWoW cannot and does not modify this power) but the rest of the group has now more resources to demand the reasons for a specific choice.

The current version only displays the number of votes per argument. This is a simple visualization but, precisely for that reason, it may be a complex tool to use: it puts a lot of weight on the leader’s shoulders. Whether more information should be provided, as a way of lessen this weight, is being studied. As an example, sums of votes in each positive/negative column were proposed but temporarily dismissed, as a sum could be interpreted as a highlight of the total number of votes. This would be a point of view that conflicts with the original idea that each of the votes should be something that the participants should reflect on.

4.6 Number of simultaneous discussions

ColLab, the online tool, provides the resources to create more than one discussion. On its origins, ColLabWoW was designed following the same idea. However, it was decided that, as an adaption of ColLab to the format of a MMOG, it was better to have only one discussion at a time.

During a game, players focus on playing. ColLabWoW would be mainly used during non-raiding times, moments where players are not completing quests. In these situations, the environment is very appropriate to encourage total focus on the discussion, which is the ideal situation. However, providing an easy way to manage more than one discussion at a time may result in some players not focusing in any of them and/or taking too long to participate in one -or more- of them. This may lead to not desirable waiting times for the rest of the participants. Thus, ColLabWoW only displays information about one conflict at a time.
Chapter 5

Future work

In this section, the issues that should be addressed for future iterations of the design and implementation are explained and these choices motivated. The issues mentioned in previous sections which have not been studied in more depth yet are not mentioned again here.

5.1 New design

As stated in the analysis of the first version, a new design of the add-on is needed. This change is specially important in the editing stage, so that the interaction is faster and more comfortable. At the same time, it should have reliable data, that is, no synchronization issues should be left unsolved.

5.1.1 New settings stage

To prepare the data for the discussion, the player that has started it will be prompted to set options. After choosing the scope of the discussion, as in the previous version, a set of different types of discussions will be available to choose from (e.g. conflict between two members of the group, set time for a raid) and the player will be able to select a template for that type or an empty matrix. When this process is finished, ColLabWoW will have all the information needed to create preset values, if needed, for the matrix representing the conflict.

5.1.2 New first stage: editing

The creator of the discussion edits the matrix, if wished. When finished, an invitation is sent to the other players in the group. Notice how this is
changed from the previous version, where all the players received an invitation automatically when the discussion was started. Letting the creators postpone this message to the other participants allows them to set the environment as they want to present it to the rest of the players and then ask them to join.

To provide participants with a basis to work on, if a template has been chosen, the preset values generated in the previous stage will be displayed from the beginning of the deliberation. However, these preset values will not be editable; though there will be an option to dynamically add a new argument. Each of the participants will be able to add only two arguments per interest and option: one for the positive column and one for the negative. These two types of arguments, preset and added, will be displayed differently so that participants may distinguish them easily. As for synchronization issues, they are not possible in this version, as there is no simultaneous editing. Players may add arguments if they consider that the ones already in the matrix are not enough. If Alice adds an argument almost at the same time as Bob and she sees that the two new arguments are the same, then Alice may remove her argument. In other words, adding and deleting the arguments a player owns are the only ways of editing the arguments for the discussion. To provide participants with more feedback, the visualization will include information about which parts of the matrix are being altered edited (by adding an argument) by the players.

As adding is possible, it may seem reasonable to have its opposite option, deleting, not only the player’s arguments but any of the arguments provided. However, it was decided that this will be forbidden (or, rather, not shown as an option). Having a player deleting what a different player previously added conflicts with the philosophy of ColLabWoW. Instead of eliminating an argument that a player do not agree with, this player may just not vote for it. It could be proposed to save a history of the editions, to allow undoing. However, this would make that version of the add-on heavier, a fact that is not desirable for this design.

5.2 New functionality

New actions will be added for the master to help keep the discussion at an interesting pace: if the waiting time for a particular (or more than one) player to finish a stage is too long, maybe due to the temporary absence of this player, the leader will have the option to force the whole group to go to the next stage. This will avoid the unnecessary block of the process if one player abandons the discussion but forgets to “officially” do so.
An action that allows to go in the opposite direction should be added as well: if Alice is waiting for the rest to finish editing and then another change is made that triggers a desire in Alice of going back to editing and “reply” to this new proposal, then she should be able to cancel her waiting stage. That is, the possibility of undoing should always be possible.

Also, to provide players with more feedback, the list of participants will have an additional element: when a player is finished editing or voting, an X (or a similar sign, like a tick) will appear next to that name on the participants list, so that the rest of the participants know who has finished already and who is still reflecting on the discussion.

5.3 Localization/Internationalization

The current version of ColLabWoW is in English. Making it available for almost any language would be an easy adaptation of the code. A good practice in add-on development is to encourage users to contribute to the internationalization of the mod, so that they volunteer to translate it to their own languages.

It is worth considering, though, that the discussion would usually be kept in English, as members of groups in WoW are usually from different countries. Having the add-on locally translated may induce players to believe that they can participate in the discussion in their own language as well. To avoid having a Tower of Babel as a discussion, this localization, if made, should be carried with precaution.
Bibliography


